

Selected Topics in Interactive Computer Graphics

Habilitationsschrift

vorgelegt von
Dr.-Ing. Martin Lambers

eingereicht bei der
Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen

Siegen 2020

Contents

Introduction	3
1 Mappings between the Sphere and Planar Surfaces	4
2 Simulation of Time-Of-Flight Sensors	6
3 Complementary Work and Related Topics	9
Conclusion	11
Bibliography	11
Publications	15
Interactive Creation of Perceptually Uniform Color Maps	15
Distortion Optimized Spherical Cube Mapping for Discrete Global Grid Systems	20
Survey of Cube Mapping Methods in Interactive Computer Graphics . .	25
Fast Motion Estimation for Field Sequential Imaging: Survey and Bench- mark	34
Realistic Lens Distortion Rendering	47
Robust Range Camera Pose Estimation for Mobile Online Scene Recon- struction	53
Quantified, Interactive Simulation of AMCW ToF Camera including Mul- tipath Effects	66
Mappings between sphere, disc, and square	80
Lowering the entry barrier for students programming Virtual Reality ap- plications	101
Comparison Of Spherical Cube Map Projections Used In Planet-Sized Terrain Rendering	105
Simulation of Time-of-Flight Sensors for Evaluation of Chip Layout Variants	144
Ground Truth for Evaluating Time of Flight Imaging	152
Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion	175
User Interface for Volume Rendering in Virtual Reality Environments .	183
Ellipsoidal Cube Maps for Accurate Rendering of Planetary-Scale Ter- rain Data	194
Structural Performance Evaluation of Curvilinear Structure Detection Algorithms with Application to Retinal Vessel Segmentation . . .	200
A stereoscopic movie player with real-time content adaptation to the dis- play geometry	209

Introduction

This habilitation thesis summarises the postdoctoral scientific work of the author in the following areas:

Sec. 1: Mappings between the Sphere and Planar Surfaces

These mappings are of central importance for various applications in Computer Graphics (e.g. environment maps, shadow and lighting maps, sampling strategies for ray tracing) and in Cartography, Remote Sensing, and Astronomy (e.g. mapping of astronomical entities and mapping of surfaces of celestial bodies). Their properties, in particular areal and angular distortions and discontinuities, are analysed for the special cases of mappings to the square and the circle [8] and mappings to cube faces [3]. Cube faces are of particular importance for the visualization of data about the Earth as well as neighboring planets and moons [10, 15]. In this context, the difference between the rotational ellipsoid model and the sphere has to be taken into account [2].

Sec. 2: Simulation of Time-Of-Flight Sensors

The simulation of Time-Of-Flight sensor imagery allows evaluation of sensor design variations without producing prototypes [11] and the creation of realistic data with Ground Truth [12] for the evaluation of data processing algorithms [4]. In addition to light propagation [7], the simulation must consider appropriate models of the lens system [5] and the sensor electronics [11] to produce realistic results. The results from this work were used to support research activities in 3D reconstruction [13, 6].

Sec. 3: Complementary Work and Related Topics

Additional research work motivated by teaching activities and collaborations was carried out in related areas such as stereoscopic rendering [17], Virtual Reality [9, 14], Scientific Visualization [1], and Image Analysis [16].

The overarching topic of this work is the *interactive* aspect of Computer Graphics: the ultimate goal is to enable interactive visual analysis [1, 2, 3, 5, 8, 10, 14, 15, 16], simulate dynamic scenes in an interactive fashion so that motion is handled appropriately [4, 7, 11, 12], reconstruct 3D environments with interactive sensor handling [6, 13], or render stereoscopic contents with the goal of immersion [9, 17].

All of these use cases impose hard constraints on the computing time available to produce or to process each frame of information, and therefore require careful design of data structures and algorithms that make efficient use of dedicated computing resources such as Graphics Processing Units (GPUs).

1 Mappings between the Sphere and Planar Surfaces

Overview

Mappings between the sphere and planar surfaces have been studied for centuries, starting with world maps and celestial maps [Sny87]. Since the sphere surface is not developable, such mappings always exhibit either areal or angular distortions, and often both. Equal-area mappings preserve area ratios, at the cost of large angular distortions, and conformal mappings preserve angles locally, at the cost of large area distortions.

In the context of Computer Graphics, such mappings are central in various areas. The most common one is environment mapping, in which information about the environment is used to compute lighting for objects placed within that environment. In this case, the spherical environment around a center is mapped to one or more planar surfaces represented by two-dimensional textures. Environment mapping can be used to implement glossy or mirroring reflections inside graphics pipelines that only support local illumination, or to represent complex environmental lighting that cannot be modelled using distinct light sources, such as in sky-lit scenes. Other application areas include panoramic imaging, procedural texturing, the generation of sampling points on a disc or sphere for material/light interaction sampling, and the representation of data bound to a sphere surface in visualizations of, for example, remote sensing data for Earth.

Projections of a sphere or a hemisphere to a single planar surface (often a disc or a square) is possible (see Fig. 1 for examples), but results in strong distortions and often in singularities [Sny87]. Nevertheless, such mappings have useful applications in Computer Graphics, for example in panoramic imaging and sampling point generation [8].

The first step to reduce distortions is to subdivide the sphere into regions that are mapped to separate planar surfaces, usually the faces of a polyhedron. As the number of faces grows, distortions are reduced significantly [Sny92], at the cost of introducing interruptions at the face boundaries that often manifest themselves as C^1 discontinuities of the mapping function.

Most existing hardware and algorithms are build for handling images and similar two-dimensional sampled data as rectangular arrays with fixed sample spacings in horizontal and vertical direction. Often, especially in the context of GPUs and hierarchical methods such as quad trees or mipmaps, the most efficient representation is even more restricted, requiring square arrays with one fixed sample spacing in both directions.

This makes the cube the polyhedron of choice for many applications, especially in interactive Computer Graphics [3] where efficient use of computational resources is paramount: the number of faces is low, and each face is a square (see Fig. 1).

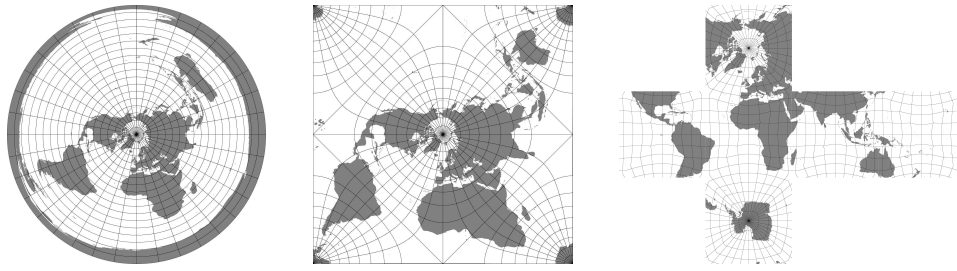


Figure 1: Mappings of a sphere surface. From left to right: an equidistant mapping to a disc, a conformal mapping to a square, and an equal-area mapping to the faces of a circumscribing cube.

Contributions

In the context of planetary-scale terrain rendering for visualization purposes, the author introduced the use of an equal-area cube mapping approach for hierarchical data management [15]. This was an improvement over previous methods with more complex map compositions [Koo+09] or other cube mapping variants [LMG10]. However, the focus on area distortions was limited. Drawbacks of the chosen equal-area mapping method include high computational costs and additional C^1 discontinuities at the diagonals of each cube face.

A subsequent survey in this application context compared many more mapping methods [10]. Each mapping was analyzed with regard to its computational costs and its area and angular distortions. For distortion analysis, two measurements for area and angular distortions were defined based on the texture sampling needs of spherical terrain rendering, in particular clip mapping.

These and other [SM01] highly specific measurements do not translate well to other application domains. For that reason, alternative measurements that are application independent were derived and first applied to mappings between sphere, disc, and square [8]. These measurements are based on the framework of Tissot's Indicatrix [Sny87], used in world map projection analysis. The idea of the Indicatrix is that any mapping method maps an infinitesimal circle on the sphere onto an infinitesimal ellipse on the planar surface. This ellipse describes the local characteristics of the map projection. Its most important properties are the lengths a of its semi-major axis and b of its semi-minor axis. Based on product and ratio of these two lengths, measurements D_A for local area distortion and D_I for local isotropy distortion, which is strongly related to local angular distortion, were defined. Although more properties of the ellipse can be analyzed, these two measurements are sufficient to characterize the map sampling performance in the context of Computer Graphics.

Later, these generalized measurements were applied to cube mapping methods used in the wider context of interactive Computer Graphics [3], not limited to planetary terrain rendering. The simplest form of cube mapping, and the one

commonly implemented in graphics hardware, is equivalent to Gnomonic mapping from the sphere center onto the faces of a circumscribing cube. This mapping results in relatively strong area and angular distortions. Over the years, many methods were developed to reduce these distortions by manipulating the cube face coordinates obtained from the (potentially hardware-based) implementation of basic Gnomonic cube mapping. The aforementioned survey of these methods revealed that a common pattern of these methods is to apply a one-dimensional transformation function to the horizontal and vertical cube face coordinates that is sigmoid in shape. A systematic search in the space of these functions revealed an algebraic sigmoid that results in a cube mapping method with better distortion properties than previously available alternatives, at reduced computational costs.

One remaining problem with cube mapping is the handling of the C^1 discontinuities at cube face borders. Graphics hardware typically has functionality builtin that enables seamless sampling across cube face boundaries (available in OpenGL via `GL_TEXTURE_CUBE_MAP_SEAMLESS`), which is sufficient for the typical use case of environment mapping. However, other application areas have different requirements. Especially in the context of numerical analysis or simulation, such discontinuities are undesirable. Depending on the application context, it is possible to move them over areas of little interest [2] by rotation.

Furthermore, if the application data is about the surface of the Earth or other celestial bodies, the underlying model is not a perfect sphere but rather a spheroid, or ellipsoid of revolution. The difference between sphere and spheroid cannot be ignored if one strives for maximum precision, especially if elevation measurements are involved [15]. The coordinate shift that was originally used to solve this problem is straightforward, but an alternative was later found that is both more elegant and more precise [2].

2 Simulation of Time-Of-Flight Sensors

Overview

Sensor simulation allows comparative analysis of sensor design variations without building prototypes [11], the analysis of sensor behaviour in application scenarios without recreating these scenarios [BDC20], and the evaluation of sensor data processing algorithms based on Ground Truth information about both scene and camera [12].

Time-Of-Flight (ToF) sensors measure per-pixel information about the distance of scene objects. This distinguishes them from other types of imaging sensors and opens a completely new range of applications. Here, we focus on the simulation of amplitude-modulated continuous-wave (AMCW) ToF sensors because of their importance in application fields and because of the unique challenges they pose.

ToF sensors measure distances based on the time t that light travels from the active light source (in close proximity to the sensor) to an object in the scene and back to the sensor: $d = \frac{1}{2} \cdot c \cdot t$, with c being the speed of light. AMCW ToF sensors

emit intensity modulated light in the near infrared range. The sensor pixels measure the correlation between the reference signal g and the light signal s reflected from the scene [Kol+10]:

$$C(\tau) = s \otimes g = \lim_{T \rightarrow \infty} \int_{-T/2}^{T/2} s(t)g(t + \tau)dt \quad (1)$$

Assuming a sinusoidal signal with a modulation frequency f_{mod} , a correlation amplitude a , a correlation bias b , and a distance-dependent phase shift $\phi = 2\pi \cdot 2d \cdot \frac{f_{\text{mod}}}{c}$, the correlation measurement is

$$C(\tau) = \frac{a}{2} \cos(f_{\text{mod}}\tau + \phi) + b \quad (2)$$

The common approach to reconstruct the phase shift ϕ for the distance computation is to use the arctangent on four samples of the correlation function $D_i = C(i \cdot \frac{\pi}{2})$, $i \in \{0, 1, 2, 3\}$. Using the common library function `atan2`, we have

$$\phi = \text{atan2}(D_3 - D_1, D_0 - D_2) \quad (3)$$

The correlation function samples D_i are called phase images and are obtained by subtracting two signals $N_{A,i}$ and $N_{B,i}$ per pixel: $D_i = N_{A,i} - N_{B,i}$. These signals result from the electrons generated in the optically active area of a sensor pixel, which are directed towards two readout circuits A and B using an electric field that is based on the reference signal g .

Sensors based on this principle suffer from various sources of systematic errors, including the following (see Fig. 2):

Distance inhomogeneity: Light paths received by one sensor pixel do not generally originate from a single surface point, but rather from a solid angle around the main light direction through the pixel center. Thus, light paths from foreground and background objects with different phase shifts mix, which leads to distance reconstruction errors. Due to the nonlinear reconstruction principle, the reconstructed distance does not necessarily lie between the foreground and background distances, resulting in the "flying pixel" effect.

Motion artefacts: In dynamic scenes, the camera pose and scene surfaces may change both during the acquisition time of a single phase image and in the time between phase image acquisitions, leading to inconsistent phase information from which the distance is reconstructed.

Multipath effects: Light does not only travel from light source to surface and directly back. This direct light path is superimposed with many indirect light paths, which deteriorates the phase sampling depending on the amplitude of the indirect light paths.

The common root cause of these error sources is that phase samples do not represent a single surface point.

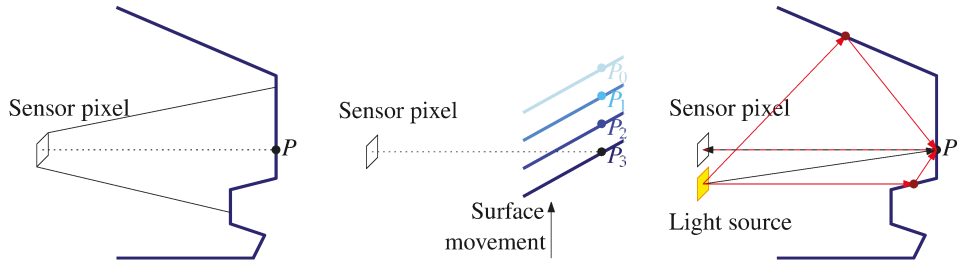


Figure 2: Error sources of AMCW ToF sensors. From left to right: distance inhomogeneity leads to flying pixel effects, surface (or camera) motion leads to motion artefacts, mixing of direct (black) and indirect (red) light paths leads to multipath artefacts.

Realistic simulation of AMCW ToF sensors must take these and other effects into account, and to be useful in the application scenarios listed above, it must be able to simulate complex and dynamic scenes. This requires an appropriate light propagation model as well as a sensor model that takes the acquisition principle and chip properties into account.

Contributions

Previous AMCW ToF simulation models already used spatial oversampling to account for distance inhomogeneity and simulated the four phase images at different points in time to account for motion artefacts [KK09]. However, modelling of physical quantities and chip properties was limited, so that the noise model was phenomenological only, and quantitative comparison with real sensor data was not yet possible.

The basic AMCW ToF simulation model introduced by the author [11] was the first that was evaluated not only qualitatively, but also quantitatively against real sensor pixels. For this purpose, its physically-based light propagation model (from light source power and emission characteristics to the energy accumulated in each sensor pixel) was combined with a sensor model that is based on the AMCW ToF acquisition principle but additionally takes chip characteristics into account, including conversion of energy to electron-hole pairs and their distribution to the A and B readout circuits, and design parameters such as pixel spacing and the layout of optically active areas on each pixel. In a collaboration with pmdtec (<https://pmdtec.com/>), this enabled the evaluation of an alternative chip design without building a prototype. Since the robustness against motion artefacts was of central importance for this evaluation, the simulation model improved their simulation by also taking motion during the acquisition of a single phase image into account.

Spatial and temporal oversampling and light propagation in complex and dynamic scenes require significant computational resources. Since the simulation can be parallelized on pixel level, the use of Graphics Processing Units (GPUs) is rea-

sonable. However, using the rasterization-based graphics pipeline accelerated by GPUs means that light propagation is initially restricted to direct illumination only, which means that multipath effects cannot be simulated.

This restriction was later lifted by extending the Reflective Shadow Map approach [DS05] with information about active AMCW virtual point lights to arrive at a single-bounce approximation of indirect lighting [7]. Reflective Shadow Maps are originally limited to a single view of the scene and therefore lack information about directly illuminated surfaces that are outside that view. This restriction can be lifted by using full environment maps such as cube maps, which benefit from the improvements described in Sec. 1.

A limitation of these sensor models is that they are either based on the basic pinhole camera projection principle [KK09] or take only thin lens vignetting effects into account [11]. For comparisons with real cameras, lens effects such as radial and tangential distortion have to be taken into account. To lift this limitation, the standard lens calibration model from widely used software packages such as OpenCV and Matlab was integrated into rendering pipelines so that lens parameters measured with standard tools can be used directly in camera simulation [5].

All of the simulation functionality described above was implemented by the author in the Open Source camera simulation software CamSim (<https://marlam.de/camsim>), which was recently described as "the most advanced ToF camera simulator" [BDC20]. It provides a rich set of Ground Truth information, including information about scene geometry (positions, surface normal vectors, and IDs of scene objects, shapes, and triangles), scene dynamics (forward and backward optical flow in 2D and 3D), camera pose and parameters, and simulated sensor raw data such as phase images. This information is useful for the evaluation of ToF data processing algorithms [12], for example in the area of motion artefact correction [4].

This kind of information is also useful for higher-level applications using per-pixel distance data, such as 3D scene reconstruction [13], a crucial part of which is the estimation of camera poses from the input images of a handheld camera [6].

3 Complementary Work and Related Topics

Additional research in related areas of interactive Computer Graphics was motivated by activities in teaching and by collaborations. This work is detailed in the following subsections.

Stereoscopic Rendering

Stereoscopic rendering provides distinct images for the left and right eye of the viewer, so that depth perception based on stereopsis becomes possible. Stereoscopic video is typically preproduced and only played back, not rendered on the fly. This is problematic since the optimal configuration of left and right view de-

depends on viewing geometry parameters such as screen size and distance. This means the same stereoscopic video material cannot target both cinema and home viewing conditions with the same level of quality.

To lift this restriction, functionality to estimate the disparity between both views and retarget them to new viewing conditions during playback was integrated into the author's GPU-based stereoscopic video player Bino (<https://bino3d.org>) in a collaboration with INRIA (France) [17].

Virtual Reality

Virtual Reality systems place especially hard time constraints on the delivery of the next rendered frame, since latency in the reaction to user input will prevent immersion and may even lead to simulation sickness.

In teaching the technically oriented Virtual Reality (VR) lecture and tutorial at the University of Siegen, it became apparent that for students implementing their own VR applications, the main entry barrier is the management of multiple displays, GPUs, and hosts in a render cluster. Student project groups supervised at the University of Siegen used abstraction libraries and frameworks such as Equalizer [EMP09] to target Virtual Reality systems, for example in the field of interactive volume analysis [14], but these frameworks are typically very complex themselves, so that a large part of the project time was spent fighting software complexities instead of focussing on the topic.

To reduce this burden and enable more student activities in the Virtual Reality teaching context, a concept was devised to hide such complexity and provide a renderer interface that is as familiar as possible to students who completed the beginner-level graphics courses, while still supporting the complete range of Virtual Reality hardware, from head-mounted displays to render clusters driving multiple displays [9].

Scientific Visualization

One of the building blocks of visualization systems is the mapping of quantities to colors using color maps. Ordered values progressing from low to high are mapped to sequential maps, and data ordered around a central value with extrema in both directions are mapped to diverging maps. For unordered data, qualitative maps that do not imply magnitude differences are used. Examples for these categories are given in Fig. 3.

The quality of color maps is determined by criteria such as discriminative power and perceptual uniformity [Buj+18]. The latter is important to avoid introducing interpretation bias. These perceptual qualities at the same time form a bridge to the larger field of Computer Graphics. Driven by the needs of visualization applications as well as teaching activities, a set of methods to allow interactive design of color maps that are perceptually uniform by construction was introduced [1]. The examples in Fig. 3 were created using these methods.

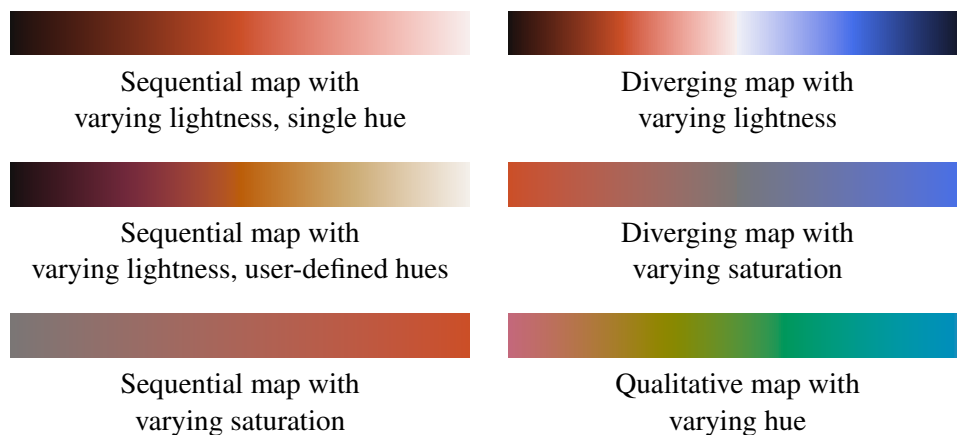


Figure 3: Examples for sequential, diverging, and qualitative color maps.

Image Analysis

The detection of curvilinear structures in images is important in medical applications, such as the analysis of blood vessel structures in the eye. To evaluate such methods, measures based on pixel differences were often applied, but these fail to capture the true performance of each method because they do not take structural aspects into account. A structure-aware evaluation scheme based on graph representation and matching was introduced [16] to alleviate this problem.

Conclusion

The main theme of the post-doctoral scientific work of the author is the *interactive* aspect of Computer Graphics and the restriction it places on computation time, necessitating the efficient use of GPUs to meet application requirements.

The work covers both basic research such as the analysis of mappings between spheres and planar surfaces (Sec. 1) and application-oriented research such as the simulation of Time-of-Flight sensors (Sec. 2), both of which provide results that are relevant to the larger field. These main areas are complemented with research work in related fields, motivated by teaching activities and collaborations (Sec. 3).

Bibliography

References

- [Sny87] J.P. Snyder. *Map projections—a working manual*. Vol. 1395. Professional Paper. US Geological Survey, 1987. DOI: [10.3133/pp1395](https://doi.org/10.3133/pp1395).
- [Sny92] J.P. Snyder. “An Equal-Area Map Projection For Polyhedral Globes.” In: *Cartographica* 29.1 (1992), pp. 10–21. DOI: [10.3138/27H7-8K88-4882-1752](https://doi.org/10.3138/27H7-8K88-4882-1752).

- [Koo+09] R. Kooima, J. Leigh, A. Johnson, D. Roberts, M. SubbaRao, and T.A. DeFanti. “Planetary-Scale Terrain Composition.” In: *IEEE Trans. Visualization and Computer Graphics* 15.5 (2009), pp. 719–733. DOI: [10.1109/TVCG.2009.43](https://doi.org/10.1109/TVCG.2009.43).
- [LMG10] R. Lerbour, J.-E. Marvie, and P. Gautron. “Adaptive Real-Time Rendering of Planetary Terrains.” In: *Full Paper Proc. Int. Conf. Computer Graphics, Visualization and Computer Vision (WSCG)*. Feb. 2010.
- [SM01] J. Snyder and D. Mitchell. *Sampling-Efficient Mapping of Spherical Images*. Tech. rep. Microsoft Research, 2001. URL: <https://www.microsoft.com/en-us/research/publication/sampling-efficient-mapping-spherical-images/>.
- [BDC20] M. Baumgart, N. Druml, and C. Consani. “Multipath Ray-Tracing-Based Modelling of Time-of-Flight Cameras.” In: *Sensor Systems Simulations: From Concept to Solution*. Ed. by W. D. van Driel, O. Pyper, and C. Schumann. 2020, pp. 93–147. ISBN: 978-3-030-16577-2. DOI: [10.1007/978-3-030-16577-2_4](https://doi.org/10.1007/978-3-030-16577-2_4).
- [Kol+10] A. Kolb, E. Barth, R. Koch, and R. Larsen. “Time-of-Flight Cameras in Computer Graphics.” In: *Computer Graphics Forum* 29.1 (2010), pp. 141–159. DOI: [10.1111/j.1467-8659.2009.01583.x](https://doi.org/10.1111/j.1467-8659.2009.01583.x).
- [KK09] M. Keller and A. Kolb. “Real-time Simulation of Time-Of-Flight Sensors.” In: *J. Simulation Practice and Theory* 17 (2009), pp. 967–978. DOI: [10.1016/j.simpat.2009.03.004](https://doi.org/10.1016/j.simpat.2009.03.004).
- [DS05] C. Dachsbacher and M. Stamminger. “Reflective Shadow Maps.” In: *Proc. Symp. Interactive 3D Graphics and Games (I3D)*. 2005, pp. 203–231. DOI: [10.1145/1053427.1053460](https://doi.org/10.1145/1053427.1053460).
- [EMP09] S. Eilemann, M. Makhinya, and R. Pajarola. “Equalizer: A Scalable Parallel Rendering Framework.” In: *IEEE Trans. Visualization and Computer Graphics* 15.3 (May 2009), pp. 436–452. DOI: [10.1109/TVCG.2008.104](https://doi.org/10.1109/TVCG.2008.104).
- [Buj+18] R. Bujack, T. L. Turton, F. Samsel, C. Ware, D. H. Rogers, and J. Ahrens. “The Good, the Bad, and the Ugly: A Theoretical Framework for the Assessment of Continuous Colormaps.” In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* 24.1 (Jan. 2018), pp. 923–933. DOI: [10.1109/TVCG.2017.2743978](https://doi.org/10.1109/TVCG.2017.2743978).

Publications

- [1] M. Lambers. “Interactive Creation of Perceptually Uniform Color Maps.” In: *Proc. EuroVis (Short Papers)*. May 2020. DOI: [10.2312/evs.20201048](https://doi.org/10.2312/evs.20201048). See page 15.

- [2] A. Dimitrijević, P. Strobl, M. Lambers, A. Milosavljević, and D. Rančić. “Distortion Optimized Spherical Cube Mapping for Discrete Global Grid Systems.” In: *Proc. Int. Conf. Information Society and Technology (ICIST)*. Mar. 2020, pp. 109–113. URL: <https://www.eventiotic.com/eventiotic/library/paper/595>. See page 20.
- [3] M. Lambers. “Survey of Cube Mapping Methods in Interactive Computer Graphics.” In: *The Visual Computer* 36.5 (May 2020), pp. 1043–1051. DOI: [10.1007/s00371-019-01708-4](https://doi.org/10.1007/s00371-019-01708-4). See page 25.
- [4] H. Steiner, H. Sommerhoff, D. Bulczak, N. Jung, M. Lambers, and A. Kolb. “Fast Motion Estimation for Field Sequential Imaging: Survey and Benchmark.” In: *Image and Vision Computing* 89 (2019), pp. 170–182. DOI: [10.1016/j.imavis.2019.07.001](https://doi.org/10.1016/j.imavis.2019.07.001). See page 34.
- [5] M. Lambers, H. Sommerhoff, and A. Kolb. “Realistic Lens Distortion Rendering.” In: *Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*. June 2018. URL: <http://wscg.zcu.cz/wscg2018/2018-WSCG-Papers-Separated.html>. See page 47.
- [6] D. Presnov, M. Lambers, and A. Kolb. “Robust Range Camera Pose Estimation for Mobile Online Scene Reconstruction.” In: *IEEE Sensors Journal* 18.7 (2018), pp. 2903–2915. DOI: [10.1109/JSEN.2018.2801878](https://doi.org/10.1109/JSEN.2018.2801878). See page 53.
- [7] D. Bulczak, M. Lambers, and A. Kolb. “Quantified, Interactive Simulation of AMCW ToF Camera including Multipath Effects.” In: *MDPI Sensors* 18.1 (2018). DOI: [10.3390/s18010013](https://doi.org/10.3390/s18010013). See page 66.
- [8] M. Lambers. “Mappings between sphere, disc, and square.” In: *Journal of Computer Graphics Techniques* 5.2 (Apr. 2016), pp. 1–21. URL: <http://jcggt.org/published/0005/02/01/>. See page 80.
- [9] M. Lambers. “Lowering the entry barrier for students programming Virtual Reality applications.” In: *Eurographics - Education Papers*. May 2016. DOI: [10.2312/eged.20161024](https://doi.org/10.2312/eged.20161024). See page 101.
- [10] A. Dimitrijević, M. Lambers, and D. Rančić. “Comparison Of Spherical Cube Map Projections Used In Planet-Sized Terrain Rendering.” In: *Facta Universitatis, Series: Mathematics and Informatics* 31.2 (2016), pp. 259–297. URL: <http://casopisi.junis.ni.ac.rs/index.php/FUMathInf/article/view/871>. See page 105.
- [11] M. Lambers, S. Hoberg, and A. Kolb. “Simulation of Time-of-Flight Sensors for Evaluation of Chip Layout Variants.” In: *IEEE Sensors Journal* 15.7 (July 2015), pp. 4019–4026. DOI: [10.1109/JSEN.2015.2409816](https://doi.org/10.1109/JSEN.2015.2409816). See page 144.

- [12] R. Nair, S. Meister, M. Lambers, M. Balda, H. Hoffmann, A. Kolb, D. Kondermann, and B. Jähne. *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Ed. by M. Grzegorzec, C. Theobalt, R. Koch, and A. Kolb. Springer, 2013. Chap. Ground Truth for Evaluating Time of Flight Imaging, pp. 52–74. DOI: [10.1007/978-3-642-44964-2_4](https://doi.org/10.1007/978-3-642-44964-2_4). See page 152.
- [13] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. “Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion.” In: *Proc. 3D Vision (3DV)*. June 2013, pp. 1–8. DOI: [10.1109/3DV.2013.9](https://doi.org/10.1109/3DV.2013.9). See page 175.
- [14] J. Klein, D. Reuling, J. Grimm, A. Pfau, D. Lefloch, M. Lambers, and A. Kolb. *User Interface for Volume Rendering in Virtual Reality Environments*. Tech. rep. Computer Graphics Group, University of Siegen, Feb. 2013. URL: <http://arxiv.org/abs/1302.2024>. See page 183.
- [15] M. Lambers and A. Kolb. “Ellipsoidal Cube Maps for Accurate Rendering of Planetary-Scale Terrain Data.” In: *Proc. Pacific Graphics (Short Papers)*. Sept. 2012, pp. 5–10. DOI: [10.2312/PE/PG/PG2012short/005-010](https://doi.org/10.2312/PE/PG/PG2012short/005-010). See page 194.
- [16] X. Jiang, M. Lambers, and H. Bunke. “Structural Performance Evaluation of Curvilinear Structure Detection Algorithms with Application to Retinal Vessel Segmentation.” In: *Pattern Recognition Letters* 33.15 (Nov. 2012), pp. 2048–2056. DOI: [10.1016/j.patrec.2012.05.008](https://doi.org/10.1016/j.patrec.2012.05.008). See page 200.
- [17] S. Duchêne, M. Lambers, and F. Devernay. “A stereoscopic movie player with real-time content adaptation to the display geometry.” In: *Stereoscopic Displays & Applications XXIII*. Vol. 8288. Proc. SPIE. Jan. 2012. DOI: [10.1117/12.908741](https://doi.org/10.1117/12.908741). See page 209.

The remainder of this document contains these 17 publications in the order in which they appear in this list.

Interactive Creation of Perceptually Uniform Color Maps

M. Lambers¹ 

¹Computer Graphics Group, University of Siegen, Germany

Abstract

A large number of design rules have been identified for color maps used in Scientific Visualization. One of the most important of these is perceptual uniformity, which at the same time is one of the hardest to guarantee when color maps are created from user input. In this paper, we propose parameterized color map models for a variety of application areas. To allow interactive creation of color maps, these models are based on few intuitive parameters, and at the same time guarantee approximate perceptual uniformity.

CCS Concepts

• Human-centered computing → Visualization toolkits;

1. Introduction

Color mapping is a central technique in Scientific Visualization. Toolkits typically include a selection of prefabricated color maps to choose from (e.g. Matplotlib, Matlab, Paraview). In the categorization used by the popular Color Brewer tool [BH], there are *sequential* color maps for ordered values progressing from low to high, *diverging* color maps for ordered data around a central value with extrema in both directions, and *qualitative* color maps that do not imply magnitude differences between values.

While choosing a prefabricated color map is straightforward, creating a custom color map is a demanding task. Bujack et al. summarized color map design goals [BTS⁺18]. The most important ones are *order*, *discriminative power*, and *uniformity*. A natural or intuitive order of color map entries is relatively straightforward to achieve, for example by ensuring a monotonically increasing lightness. Uniformity, on the other hand, requires perceptually equidistant differences in color map entries, which limits the available color choices for a color map designer.

Few methods exist that allow interactive creation of color maps. The most prominent one is the model derived by Wijffelaars et al. [WVVV08] from the Color Brewer techniques [HB03]. However, sequential color maps in this model use a single hue, whereas there is a need for multi-hue sequential color maps for some applications, as evidenced by the selection of prefabricated color maps available in typical visualization toolkits. The Cube Helix model of Green [Gre11] uses multiple hues, but always as a variation of going through the complete hue cycle. The ColorMoves tool [SKR18] allows to adapt and manipulate color maps during exploration of specific data sets; its focus is not on the creation of generic maps.

In this paper, we propose parameterized models for interactive creation of color maps that are approximately perceptually uniform

by construction. These models are based on a limited set of intuitive parameters, but still give the user some artistic freedom, and allow for the creation of multi-hue sequential color maps as well as color maps with constant lightness.

2. Related Work

In this section, we review parameterized color map models suitable for interactive creation of color maps.

McNames [McN06] introduced a model for sequential color maps based on increasing lightness. The color maps are designed to contain monotonically increasing gray values when desaturated (for use in print), but use different hues when used in full color mode for increased discriminative power. However, the maps produced by this model are not perceptually uniform.

Wijffelaars et al. [WVVV08] derived parameterized models for sequential, diverging, and qualitative color maps from the Color Brewer techniques [HB03; BH]. The set of parameters is intuitive. However, there are several limitations. Sequential color maps are based on a single hue varying in lightness from dark to light; multiple hues, which can increase the discriminative power of sequential color maps, are not supported. Color maps with constant lightness, which are useful in 3D visualizations because they do not interfere with additional shading, cannot be created. Furthermore, while the resulting color maps are often approximately perceptually uniform if parameter values near the recommended defaults are chosen, this property is not enforced, and it is easy to create maps that are not perceptually uniform.

Moreland [Mor09] proposed a model for a diverging color map based on two hues, meeting in a neutral color in the middle of the map. The results are similar to the diverging maps produced with

the Wijffelaar model. The model enforces perceptual uniformity in each half of the color map via a custom color space based on CIELAB. Sequential or qualitative maps, or maps with constant lightness, are not covered by this model.

Green [Gre11] proposed the Cube Helix model for a sequential color map specifically for astronomical intensity data. It is based on increasing lightness combined with constantly changing hue to increase its discriminative power. The produced color maps are in effect similar to the ones created by the McNames model, but the Cube Helix model is constructed so that its color maps are approximately perceptually uniform. However, the hues are always cycling through the complete hue cycle and cannot be chosen freely. Furthermore, diverging or qualitative maps, or maps with constant lightness, are not covered by this model.

In contrast to the models listed above, our models support sequential, diverging, and qualitative maps, allow multiple custom hues inside sequential maps for increased discriminative power, and allow color maps with constant lightness for applications that require additional shading.

3. Perceptually Uniform Color Maps in CIELCH

We consider perceptual differences using the euclidean distance measure in the perceptually linear CIELUV color space, as Wijffelaars [WVVV08], Moreland [Mor09] and others [BTS*18] did.

In order to arrive at intuitive parameters, we use the CIELCH representation of this color space based on lightness $L \in [0, 100]$, chromaticity $C \in [0, 100]$, and hue $H \in [0, 2\pi)$. Using $C = L \cdot S$, we can furthermore replace the chromaticity parameter with a more intuitive saturation parameter S .

CIELCH is related to CIELUV through $U = C \cos H$ and $V = C \sin H$. Formulating the euclidean distance in CIELUV using the CIELCH representation leads to the distance measure d :

$$d(\text{LCH}_0, \text{LCH}_1) = \sqrt{(L_0 - L_1)^2 + C_0^2 + C_1^2 - 2C_0C_1 \cos(H_0 - H_1)} \quad (1)$$

The color map models described in the following all compute a color LCH_t for each $t \in [0, 1]$ based on their individual set of parameters.

In the following subsections, we introduce *sequential*, *diverging*, and *qualitative* color map models. The individual models names are highlighted in the text **in bold italics**; each has an example in Fig. 1.

3.1. Sequential Color Maps

Variation in lightness has been identified as the main factor for discriminative power of a sequential color map [Kov15], and when it increases monotonically it also guarantees a natural order of colors [BTS*18].

An obvious approach is therefore to base sequential color maps on monotonically increasing lightness. This is consistent with all other parameterized models described in Sec. 2. However, some applications require color maps of constant lightness in order to combine them with shading in 3D rendering. In the following, we describe models for both cases.



Figure 1: Example results for all methods from Sec. 3.

3.1.1. Varying Lightness

Linearly increasing lightness is modeled with a lightness range parameter $R_L \in (0.5, 1]$:

$$\begin{aligned} L_0 &= (1 - R_L) \cdot 100 \\ L_1 &= R_L \cdot 100 \\ L(t) &= (1 - t)L_0 + tL_1 \end{aligned}$$

This leaves chromaticity (or saturation) and hue as free parameters. To let the user choose one or more hues freely, the chromaticity parameter is the one we use to enforce perceptual uniformity.

With linear increasing lightness, saturation is typically low at both the dark and the light ends, and stronger in the middle of the color map. Using a parameter $R_S \in (0.5, 1]$ for the saturation range and a parameter $S \in [0, 5]$ for the overall saturation of the map, we define:

$$\begin{aligned} S_0 &= 1 - R_S \\ S_{0.5} &= SR_S \\ S_1 &= 1 - R_S \end{aligned}$$

Note that the distances $d(\text{LCH}_0, \text{LCH}_{0.5})$ and $d(\text{LCH}_{0.5}, \text{LCH}_1)$ are only exactly equal for single-hue maps. However, both distances are dominated by the lightness differences in the multi-hue case. Because $C_0 = L_0S_0$ and $C_1 = L_1S_1$ are small, the difference in hue does not have much effect (see Eq. 1). Therefore, the difference



Figure 2: Test pattern for the sequential color map with varying lightness and multiple hues. The pattern details are visible nearly uniformly throughout the color map.



Figure 3: Test pattern for the sequential color map with varying saturation and constant lightness and hue. While the pattern details are visible nearly uniformly throughout the color map, only few details are visible.

between the distances, and therefore the deviation from perceptual uniformity, is small.

In between these points with fixed saturation and therefore chromaticity, the chromaticity is computed so that the resulting colors are perceptually uniform. This is done as follows: For two colors LCH_A and LCH_B with distance D , an interpolation parameter $s \in [0, 1]$ leads to two conditions for the chromaticity C_s :

$$\begin{aligned} d(LCH_s, LCH_A) &= sD \\ d(LCH_s, LCH_B) &= (1-s)D \end{aligned}$$

Solving Eq. 1 for C_s for both of these conditions leads to two quadratic equations with a total of four solutions. Of these, we consider only those that fulfill $\min(C_A, C_B) \leq C_s \leq \max(C_A, C_B)$, and choose the one that produces the smallest absolute error with regard to the two conditions. In some pathological cases, it may happen that no valid solution exists, in which case we fall back to $C_s = \frac{1}{2}(C_A + C_B)$.

Based on the linear lightness and the chromaticity that guarantees (approximate) perceptual uniformity, the user can now choose one or more hues.

Sequential map with varying lightness, single hue. We simply set $H(t) = H_{\text{base}}$ for a single hue $H_{\text{base}} \in [0, 2\pi)$ chosen by the user.

Sequential map with varying lightness, rainbow hues. We set $H(t) = H_{\text{base}} + tr \cdot 2\pi$, where H_{base} is a start hue, and r defines the number of rotations through the hue cycle; these parameters are consistent with the Cube Helix model [Gre11].

Sequential map with varying lightness, custom hues. The user chooses n hue steps H_i , $n \in \mathbb{N}_{>0}, t_i \in [0, 1] \forall i$. The hue for an arbitrary $t \in [0, 1]$ is computed by linear interpolation in this set, with a weight defined by $t_i \leq t < t_{i+1}$. It is important to choose the shortest path between H_i and H_{i+1} , which may require crossing the cycle period at 2π .

3.1.2. Varying Saturation

Varying saturation while keeping lightness constant provides color maps suitable for combination with shading. Allowing multiple hues in such maps does not make much sense since at low saturation and constant lightness, different hues are hardly discernible.

Sequential map with varying saturation. Based on a user-provided saturation range parameter $R_S \in (0.5, 1]$, we define

$$\begin{aligned} S_0 &= 1 - R_S \\ S_1 &= SR_S \end{aligned}$$

With $L(t) = L_{\text{base}}$ and $H(t) = H_{\text{base}}$ from user-defined parameters $L_{\text{base}} \in [0, 100]$ and $H_{\text{base}} \in [0, 2\pi)$, we then compute chromaticity $C(t)$ to enforce perceptual uniformity as described in Sec. 3.1.1.

3.2. Diverging Color Maps

Diverging color maps typically use two dominating hues, one for each half of the color map that meet in the middle in a neutral color. They can be constructed by combining two sequential color maps [WVV08; Mor09].

Such maps include two dominant hues, given by the user parameters $H_{\text{base}} \in [0, 2\pi)$ for the base hue of the first half, and $D_H \in [0, 2\pi)$ for the offset to the base hue of the second half. Adding more hues, as in the rainbow and multi-hue approaches in Sec. 3.1, does not make sense as this introduces confusion. Our diverging color map models are therefore based on the single-hue models from Sec. 3.1, resulting in a *diverging map with varying lightness* and a *diverging map with varying saturation* model.

3.3. Qualitative Color Maps

A qualitative map should not imply an ordering of the data [BH], and therefore should have approximately constant lightness. Additionally keeping the saturation constant leads to the following simple *qualitative map with varying hue* model:

$$\begin{aligned} L(t) &= L_{\text{base}} \\ S(t) &= S_{\text{base}} \\ H(t) &= H_{\text{base}} + tD_H \end{aligned}$$

where $D_H \in [0, 2\pi)$ defines the range of hue values used. Since the hues are equidistant, it follows directly from Eq. 1 that this model produces perceptually uniform maps.

4. Results

We implemented our model as an extension to the open source tool `gencolormap` [Lam], which also provides a web-based demo. Additionally, source code is available as supplementary material to this paper.

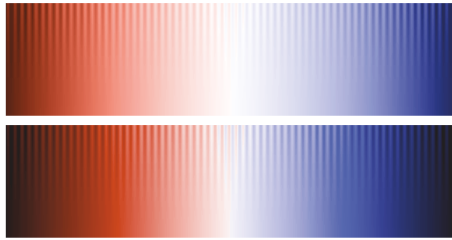


Figure 4: Comparison of test patterns for diverging maps with varying lightness. The top map was created using the Wijffelaars model [WV08] with default parameters except for hue. The bottom map was created using our model and shows stronger discriminative power in the middle of the map.

The tool has an interactive interface that allows to change the parameters of each model using sliders, and the effect is immediately visible. We aimed at intuitive parameters such as “lightness range”, “saturation” etc. so that the effects of parameter changes are as predictable as possible for the user.

Example results for each of the models proposed in this paper are shown in Fig. 1.

Perceptual uniformity (and many other properties) can be analyzed in detail by the tools provided by Bujack et al. [BTS*18], available online [BTS*]. The results of these evaluations cannot be easily summarized in a few numbers and are therefore omitted here for space reasons, but our tool includes a link and an export option for the evaluation web site, encouraging users to validate their custom color maps.

Alternatively, Kovesi designed a simple test pattern [Kov15] that reveals both the perceptual uniformity and the discriminative power of a color map at a single glance. Later, Ware et al. [WTS*17] designed a very similar test pattern for the same purpose. We include Kovesi’s pattern in our interactive tool to provide immediate feedback on the effect of color map adjustments. Note that its interpretation depends on how well the display device reproduces the sRGB color space.

Fig. 2 shows this pattern for the sequential map with varying lightness and multiple hues. The pattern detail visibility is nearly uniform throughout the image, confirming the approximate perceptual uniformity of the map.

Fig. 3 shows this pattern for the sequential map with varying saturation and constant lightness and hue. This example shows that the missing variability in lightness leads to low discriminative power of the resulting map. Nevertheless, some applications require constant lightness.

Fig. 4 compares test patterns for diverging maps with varying lightness, one created using the Wijffelaars model [WV08], and one created using our model. Our map shows stronger discriminative power in the middle of the map, and therefore an improved perceptual uniformity.

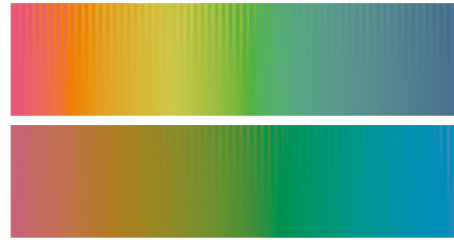


Figure 5: Comparison of test patterns for qualitative maps. The top map was created using the Wijffelaars model [WV08]. The bottom map was created using our model and shows weaker discriminative power and weaker perceptual uniformity.

As an example illustrating limitations of our models, Fig. 5 compares test patterns for qualitative maps, one created using the Wijffelaars model [WV08], and one created using our model. Our map shows weaker discriminative power and, despite guaranteeing perfect perceptual uniformity based on the CIELUV distance measure, it shows weaker perceptual uniformity than the Wijffelaars map in the test pattern. One reason for this might be that the standard observer experiments used as a base for the CIELUV color space (on which our models are based) may not be ideal for applications in scientific visualization [Kov15].

Note that the LCH colors generated by our models need to be converted to the display color space for use in visualization applications. This today is mostly the sRGB color space, which can represent significantly fewer colors than CIELCH. Therefore, color clipping may occur, damaging the perceptual uniformity and other properties of the color map. Reducing the overall saturation of a color map typically reduces instances of color clipping considerably, at the cost of reduced discriminative power of the map. Our tool reports instances of color clipping to the user and at the same time allows adjustment of saturation (and other parameters) so that the user can interactively trade off perceptual uniformity against discriminative power where necessary.

5. Conclusion

This paper introduces a consistent set of color map models that allow interactive creation of sequential, diverging, and qualitative color maps that are approximately perceptually uniform by construction. Multi-hue sequential maps as well as maps with constant lightness are supported.

The approximate perceptual linearity built into our models is based on the CIELUV color space, and therefore subject to its limitations, in particular those related to the requirements of scientific visualization [Kov15].

The sets of user-definable parameters of our models and the measures taken to enforce approximate perceptual uniformity take some artistic freedom from the user, but on the other hand considerably ease the task of custom color map creation.

References

- [BH] BREWER, C. A. and HARROWER, M. *ColorBrewer 2.0*. URL: <http://colorbrewer2.org> (visited on 02/21/2020) 1,3.
- [BTS*] BUJACK, R., TURTON, T. L., SAMSEL, F., et al. *ColorMeasures.org*. URL: <https://colormeasures.org> (visited on 02/21/2020) 4.
- [BTS*18] BUJACK, R., TURTON, T. L., SAMSEL, F., et al. "The Good, the Bad, and the Ugly: A Theoretical Framework for the Assessment of Continuous Colormaps". *IEEE Trans. Visualization and Computer Graphics (TVCG)* 24.1 (Jan. 2018), 923–933. DOI: [10.1109/TVCG.2017.2743978](https://doi.org/10.1109/TVCG.2017.2743978) 1,2,4.
- [Gre11] GREEN, D.A. "A colour scheme for the display of astronomical intensity". *Bulletin of the Astronomical Society of India* 39.2 (June 2011). URL: <https://astron-soc.in/bulletin/11June/Abstracts/289392011.htm> 1–3.
- [HB03] HARROWER, M. and BREWER, C. A. "ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps". *The Cartographic Journal* 40.1 (2003), 27–37. DOI: [10.1179/0008704032350020421](https://doi.org/10.1179/0008704032350020421).
- [Kov15] KOVESI, P. "Good Colour Maps: How to Design Them". *arXiv e-prints*, arXiv:1509.03700 (Sept. 2015). arXiv: [1509.03700](https://arxiv.org/abs/1509.03700) [cs.GR] 2,4.
- [Lam] LAMBERS, M. *gencolormap*. URL: <https://marlam.de/gencolormap> (visited on 02/21/2020) 3.
- [McN06] MCNAMES, J. "An effective color scale for simultaneous color and gray-scale publications". *IEEE Signal Processing Magazine* 23.1 (2006), 82–96. DOI: [10.1109/MSP.2006.1593340](https://doi.org/10.1109/MSP.2006.1593340) 1.
- [Mor09] MORELAND, K. "Diverging Color Maps for Scientific Visualization". *Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, 92–103. DOI: [10.1007/978-3-642-10520-3_9](https://doi.org/10.1007/978-3-642-10520-3_9) 1–3.
- [SKR18] SAMSEL, F., KLAASSEN, S., and ROGERS, D.H. "ColorMoves: Real-time Interactive Colormap Construction for Scientific Visualization". *IEEE Computer Graphics and Applications* 38.1 (2018), 20–29. DOI: [10.1109/MCG.2018.0114615251](https://doi.org/10.1109/MCG.2018.0114615251).
- [WTS*17] WARE, C., TURTON, T.L., SAMSEL, F., et al. "Evaluating the Perceptual Uniformity of Color Sequences for Feature Discrimination". *EuroVis Workshop on Reproducibility, Verification, and Validation in Visualization (EuroRV3)*. 2017. DOI: [10.2312/eurovis.2017111074](https://doi.org/10.2312/eurovis.2017111074).
- [WVV08] WIJFFELAARS, M., VLIEGEN, R., VAN WIJK, J. J., and VAN DER LINDEN, E.-J. "Generating Color Palettes using Intuitive Parameters". *Computer Graphics Forum* 27.3 (2008), 743–750. DOI: [10.1111/j.1467-8659.2008.01203.x](https://doi.org/10.1111/j.1467-8659.2008.01203.x) 1–4.

Distortion Optimized Spherical Cube Mapping for Discrete Global Grid Systems

Aleksandar Dimitrijević*, Peter Strobl**, Martin Lambers***, Aleksandar Milosavljević* and Dejan Rančić*

* University of Niš, Faculty of Electronic Engineering, Niš, Serbia

** European Commission, DG Joint Research Centre, Ispra, Italy

*** Universität Siegen, Siegen, Germany

Aleksandar.Dimitrijevic@elfak.ni.ac.rs, Peter.Strobl@ec.europa.eu, Martin.Lambers@uni-siegen.de,

Aleksandar.Milosavljevic@elfak.ni.ac.rs, Dejan.Rancic@elfak.ni.ac.rs.

Abstract—The amount of geospatial data generated globally, together with the necessity for increased interoperability of these data, call for innovative solutions for global geospatial reference frames. Discrete Global Grid Systems are a class of spatial reference systems that use hierarchical tessellation of cells to partition and address the globe without gaps or overlaps. The specific properties of DGGs make them important candidates for future standard geospatial reference frames and fuel further efforts to investigate their potential for organization, exchange and processing such data. In this paper, we focus on DGGs based on the so-called spherical cube mapping technique and present some first results of how these can be optimized to serve as global reference frames for large volumes of gridded geospatial data.

I. INTRODUCTION

The amount of geospatial data increases with a very high velocity. As an example, high-resolution satellite and airborne footages are collected at rates of many terabytes per day. The organization of such data-collections becomes challenging per se, while processing and analyzing require suitable spatial reference frames. So far most satellite missions have come up with own individual reference grid(s) for global data representation. To facilitate the fusion of data from different missions, standardized multi-resolution reference frames would, however, be necessary.

One option to define a hierarchical tessellation of near equal-area cells at multiple levels of granularity for the entire Earth is called Discrete Global Grid Systems (DGGs) [1]. The importance of DGGs is underlined by the fact that the Open Geospatial Consortium (OGC) has founded the DGG Standard and Domain Working Groups to foster the interoperability of geospatial data. A lot of DGGs has been proposed in recent years, with various methods achieving the proper tessellation of the surface [2]. Most of such systems are based on regular, multi-resolution partitions of polyhedra, called Geodesic DGGs [3]. The two out of five design choices that fully specify a Geodesic DGG, according to [3], are a base regular polyhedron and its orientation relative to the Earth.

A significant number of proposed DGGs are based on the icosahedron and use triangular or hexagonal cells [2]. Despite their good properties in approximating the Earth's surface, the absence of orthogonal axes and cell congruency, as well as a complicated implementation seem to prevent their widespread acceptance. On the other hand, cube-based DGGs introduce greater distortion,

because of the lower number of primary partitions. However, the ease of the implementation and superior properties in data organization and retrieval make them more attractive for the usage in different applications. The main motivation for this paper is to boost the public interest for the application of the cube based DGGs by minimizing area distortion in the ellipsoid to sphere mapping and the distortion of the landmass projection through the orientation of the base cube.

II. RESEARCH QUESTIONS

The term Discrete Global Grid System is relatively new [2], but the need for a global system that would collect spatial data from all over the world is much older. Without better nomenclature, they were referred to as Earth database systems at that time. Some attempts to develop an Earth database system based on a Quadrilateralized Spherical Cube dates from the early 1970s [4]. The proposed system was modified later [5], and served for the Cosmic Background Explorer (COBE) project at NASA. Several decades later, cube-based DGGs regain popularity [6-7], mainly because they provide quadrilateral cells that can be efficiently handled [8].

Although there are numerous spherical cube map projections [9], most of the published papers about them deal with the properties of projecting the sphere to a cube, as their names imply. However, the implementation of DGGs requires the usage of a more accurate approximation of the Earth's surface, such as the WGS84 ellipsoid. This paper provides an answer to the question of what the properties of such projections are when the ellipsoid is projected to a cube and whether the distortions can be minimized by additional transformations.

The second question relates to a possibility to reduce the amount of distortion on the landmass if the projection cube is rotated, so that the areas with larger distortions are placed over the oceans or other water bodies.

The two previously described steps for the WGS84 ellipsoid projection to a cube are combined into a pipeline of transformations. These can, then, serve for constructing DGGs that would provide effective solutions for the needed standardized reference frames, boosting interoperability of global raster data.

III. DISTORTION OPTIMIZATION

In this paper, we focus on the two aspects of distortion optimization: minimizing the influence of ellipsoid to sphere mapping and reducing distortion over certain areas

by base polyhedron rotation. Although the greatest influence on the distortion comes from the chosen sphere to cube mapping, the principles described in this paper are generally applicable to all of them. To illustrate the impact of proposed methods, we have chosen adjusted spherical cube [10-11], as easy to implement, yet very efficient method to project sphere to a cube [9]. The forward transformation, i.e. mapping the spherical (ϕ, θ) to rectangular (x, y) coordinates of the cube face is defined by (1) and (2).

$$x = \phi \cdot 4/\pi \quad (1)$$

$$y = \arctan(\tan(\theta) / \cos(\phi)) \cdot 4/\pi \quad (2)$$

The adjusted spherical cube mapping is neither equal-area, nor conformal projection, but the maximum-to-minimum area distortion is 1.4142, which is far better than most other non-equal-area spherical cube map projections, for which this parameter ranges from 2.0, for the Continuous Cube mapping [12], up to 5.2 for the Tangential Spherical Cube [9].

A. Transformation Pipelines

In order to provide the WGS 84 to Cube-map coordinates transformation, and vice versa, proper pipelines are defined. Fig.1 depicts the main steps in both the forward and the inverse pipeline, combined to a cycle of coordinate transformations.

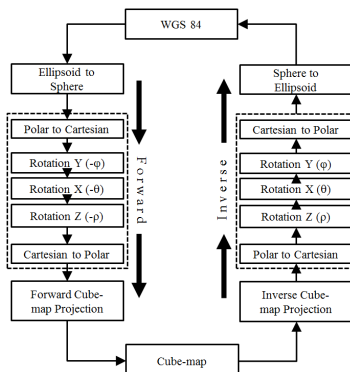


Figure 1. Coordinate transformation cycle consisting of the two pipelines – forward and inverse

The forward transformation starts with the ellipsoid to sphere transformation, presented in the next section. This step is optional and serves to reduce a certain type of distortion, according to the property we want to preserve.

The next step rotates the base cube, diminishing the distortion over the areas of interest. Since the distribution of the distortion is fixed across the faces of the cube, and depends on the chosen projection only, the impact on certain areas can be changed by rotating the base cube.

The last step in the forward pipeline is mandatory. It defines the actual sphere to cube transformation.

The inverse pipeline converts spherical cube map coordinates back to WGS84, consisting of the reverse order of the inverse transformations from the forward pipeline.

B. Ellipsoid to Sphere Transformation

The ellipsoid to sphere transformation is the first stage in the forward pipeline. It is not a unique process and depends on the property that should be preserved. A common way to perform this step is to transform geodetic latitude to some “auxiliary” latitude.

The geodetic latitude is an angle between the equatorial plane and the vector perpendicular to the surface of the ellipsoid at a given point. It is slightly greater than any auxiliary latitude, except at the Equator and poles, where they are all equal. Spherical forms of map projections can be adapted for use with the ellipsoid by substituting the geodetic latitude with one of the various auxiliary latitudes. The auxiliary latitudes were systematically described and all formulas derived by O. Adams [13], in 1921, but wider popularity is gained much later with Snyder’s working manual [14].

There are six auxiliary latitudes, each with certain properties:

- geocentric (ϕ) – an angle between the equatorial plane and the radius vector,
- parametric (η) – the parallel on the sphere (with the radius equal to the semi-major axis) has the same radius as the parallel of geodetic latitude,
- conformal (χ) – preserves angles,
- authalic (β) – preserves surface area,
- rectifying (μ) – preserves distances along meridians and
- isometric (ψ) – equal increments of isometric latitude and longitude correspond to equal distance displacements along meridians and parallels.

Geocentric and parametric latitudes are the simplest to compute. In both cases, the ratios of tangents of given auxiliary and geodetic latitude are constants. Rectifying latitude represents the other extreme on the calculation scale. It cannot be expressed in the closed-form and requires series or numerical integration. Isometric latitude is also specific. It rapidly diverges from the geodetic latitude, tending to infinity at the poles. Both, rectifying and isometric latitudes, are out of the scope of this paper.

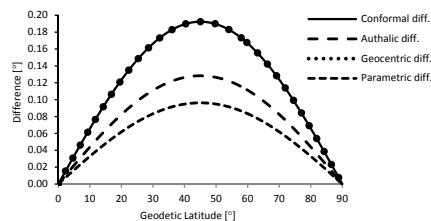


Figure 2. Auxiliary Latitudes – The divergence from the geodetic latitude

The divergence from the geodetic latitude of the four most frequently used auxiliary latitudes is shown in Fig.2. The difference is maximal at around 45°. It is interesting

to notice that geocentric and conformal latitudes are almost the same; hence, being much easier to compute, the geocentric latitude usually substitutes the conformal latitude in the calculations.

One of the main problems in cartography is preserving sizes and shapes. However, as in the projecting a sphere to a plane, projecting an ellipsoid to a sphere cannot preserve both of them. The projection can be either a conformal or equal-area. For preserving angles, geocentric latitudes can be used as a good approximation; while preserving surface area requires application of authalic latitudes.

The authalic latitude (β) is very complex to compute and requires multiple iterations for the inverse transformation. The equations (3) through (5) define forward (geodetic to authalic), while (6) through (8) define inverse (authalic to geodetic) transformation.

$$\beta = \arcsin(q/q_p) \quad (3)$$

$$q = (1 - e^2) \left\{ \frac{\sin \theta}{(1 - e^2 \sin^2 \theta)} - \frac{1}{(2e)} \right\} \ln \left[\frac{1 - e \sin \theta}{1 + e \sin \theta} \right] \quad (4)$$

$$q_p = q_{(\theta=90^\circ)} = (1 - e^2) \left\{ \frac{1}{(1 - e^2)} - \frac{1}{(2e)} \right\} \ln \left[\frac{(1 - e)}{(1 + e)} \right] \quad (5)$$

$$q = q_p \sin \beta \quad (6)$$

$$\theta_0 = \arcsin(q/2) \quad (7)$$

$$\theta_{i+1} = \theta_i + \left[\frac{(1 - e^2 \sin^2 \theta_i)^2}{2 \cos \theta_i} \right] \left\{ \frac{q}{(1 - e^2) - \sin \theta_i / (1 - e^2 \sin^2 \theta_i)} + \frac{1}{(2e)} \ln \left[\frac{1 - e \sin \theta_i}{1 + e \sin \theta_i} \right] \right\} \quad (8)$$

Aside from its complexity, the inverse authalic transformation loses its precision toward the poles, as shown in Fig.3. With a single iteration, the error is about 2 degrees at the pole, which corresponds to approximately 200km.

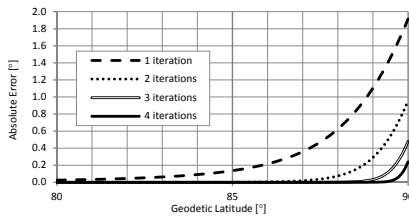


Figure 3. Authalic Latitude Inverse Function Error $|\theta - \text{InvAuth}(\text{Auth}(\theta))|$

Because of the very poor properties of the authalic latitude, like complex computation, iterative calculation of the inverse transformation with a loss of the precision in the proximity of the poles, a better solution is desirable. So, we propose an approximation, defined in (9), that is easy to compute, requires no iterations, and retains a very high precision throughout with a maximum deviation of about 0.1 arc-second (3m) around 25° latitude (see Fig.4).

$$\beta' = \arctan \left[(1 - e^2)^{2/3} \tan \theta \right] \quad (9)$$

Note that the approximated authalic latitude (9) has a form similar to the geocentric latitude ($\phi = \arctan \left[(1 - e^2) \tan \theta \right]$) and the parametric latitude ($\eta = \arctan \left[(1 - e^2)^{1/2} \tan \theta \right]$), with values somewhere in between the two.

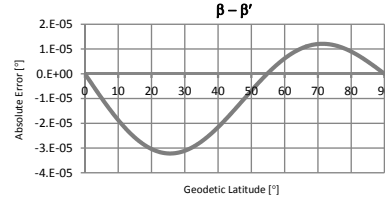


Figure 4. The difference between the authalic (β) and the approximated authalic (β') latitude

The impact of the chosen auxiliary latitude on the ellipsoid to sphere mapping distortion illustrated by the example of the adjusted spherical cube, is shown in section IV.

C. Base Cube Orientation

The distribution of the distortion depends on the chosen spherical cube map projection. Usually, the minimums are located at the centers of the faces, and the distortion increases toward the edges and corners of the base cube [9]. Hence, the impact on the area of interest can be diminished by rotating the cube and moving those areas toward the center of the faces.

The second phase in the forward pipeline performs the transformation by converting coordinates into the Cartesian coordinate system, rotating about all three axes, and transforming them back to the polar coordinate system.

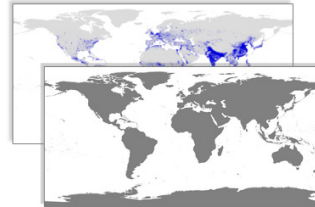


Figure 5. Raster masks used for the optimal base cube orientation

The optimal orientation can be found by varying rotation angles (φ , θ and ρ in Fig.1), from -45° to 45° , around all three axes, and comparing distortions over the areas of interest. These areas are confined by raster masks (Fig.5) defining landmass, population density, or any other criterion used for estimating an optimal orientation. The raster maps used as masks can be in any projection. However, for the sake of simplicity and efficiency, avoiding additional transformations, the maps used in experiments, as shown in Fig.5, are in LatLon WGS84 projection (EPSG:4326). The calculation is done for each pixel of all faces of the cube, that projects to a masked area, using the inverse pipeline. Since the calculation time is directly proportional to the resolution of the cube faces, the lower resolution is used for a wide range of angles, while higher resolution ones are used for fine-tuning of the base cube orientation, around expected extremes.

IV. RESULTS AND DISCUSSION

A. The Effects of Chosen Ellipsoid to Shere Transformation

The two most frequently used metrics to depict shape deformations in the projection process are angular and areal distortions. Ideally, two lines should intersect at the same angle, both on the surface of the globe and on the projected map. If the projection is conformal, the angles are preserved and, hence, the shape of the features. For the non-conformal projections, the angular distortion represents the maximum deviation from the correct angle at a given location.

On the other hand, the projection also may alter the scale of the features. The ratio of the projected and original area is known as the areal distortion. Equal-area projections preserve the area. Unfortunately, the projection cannot be both conformal and equal-area. Preserving one feature leads to sacrifice of the other, and sometimes the distortion of the unpreserved feature can be severe. So, in most of the cases a compromise is required, and, hence, the projections that are neither conformal nor equal-area are very commonly used. The adjusted spherical cube is one of such projections.

Table I summarizes the effects of the distortion using different sphere-to-ellipsoid mappings. The first row contains parameters of the perfect sphere, while the next three contain distortion for the WGS84 ellipsoid using geodetic, geocentric and the approximated authalic latitude mappings, respectively. Each row is divided into three sub-rows, for the side face, top face and the cube as a whole (an averaged value for the four side faces and two top faces). The values are separately shown for the side and top faces to illustrate the asymmetry of the mappings. Table I does not contain the minimal value angular distortion column, since the value is always 0. The maximum-to-minimum is added as an additional column to the areal distortion, as it, probably, depicts the essential aspect of the surface preserving – a cell size variation across the surface of the map. Or, in our case, across the surface of the cube face.

TABLE I.
Distortion Effects of Various Sphere-to-Ellipsoid Mappings (Auxiliary Latitudes) on the Different Cube Faces

Type	Face	Angular distortion			Areal distortion		
		Max.	Avg.	Min.	Max.	Max./Min.	Avg.
Sphere	Side	31.085	11.569	1.621	2.293	1.414	1.925
	Top	31.085	11.569	1.621	2.293	1.414	1.925
	All	31.085	11.569	1.621	2.293	1.414	1.925
Geodetic	Side	30.962	11.570	1.632	2.308	1.414	1.934
	Top	31.332	11.588	1.610	2.293	1.424	1.921
	All	31.332	11.576	1.610	2.308	1.433	1.929
Geo-centric	Side	31.085	11.569	1.621	2.300	1.419	1.927
	Top	31.084	11.569	1.632	2.300	1.410	1.934
	All	31.085	11.569	1.621	2.300	1.419	1.929
Approx. authalic	Side	31.044	11.567	1.625	2.298	1.414	1.929
	Top	31.167	11.575	1.625	2.298	1.414	1.929
	All	31.167	11.570	1.625	2.298	1.414	1.929

As it is expected, geocentric latitude produces the smallest angular distortion, while approximated authalic produces the smallest area distortion. If geocentric latitude is used in ellipsoid to sphere mapping, there is an increase of about 0.34% in area distortion (max/min ratio), while angular distortion is kept at the level of a perfect sphere. On the other hand, approximated authalic latitude keeps the areal distortion; while maximum angular distortion is increased by 0.26%. The usage of the geodetic latitude yields the largest distortions:

- 0.8% the increase of maximum angular distortion,
- 0.056% the increase of average angular distortion and
- 1.35% the increase of area distortion (max/min ratio).

B. The Optimal Orientation to Minimize Landmass Distortion

There are lots of different criteria that can be used for choosing the best orientation of the base cube. One of the most prominent goals is to preserve continental plates of being split by the cube edges and reduce overall distortion of the landmass. Without rotations, all continents, except for Antarctica, are in quite unfavorable positions with regard to the cube faces, as shown in Fig. 7.

If the rotation angles are confined to integer numbers, the minimal angular distortion of the continental plates is gained for the following rotation angles: $\varphi = 17^\circ$, $\theta = -10^\circ$ and $\rho = 32^\circ$. Fig. 6 illustrates the position of the base cube, after rotating by the defined angles.

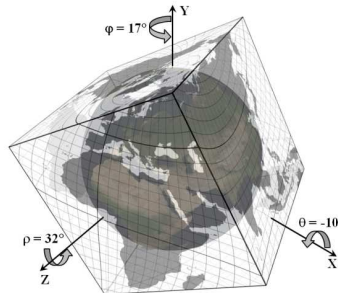


Figure 6. Optimal base cube orientation for the landmass distortion minimization

The rotation angles differ for minimum areal or aspect distortion, but we have chosen to minimize the angular distortion, because it has a wider range of possible values and hence a more noticeable difference between consecutive values of rotation angles. Also, the proposed rotation yields visually a very effective result, as can be seen in Fig. 8.

By using the proposed base cube orientation and approximated authalic latitude, an average angular distortion is reduced from 11.21° to 9.03° , while at the same time an average areal distortion is decreased from 1.92 to 1.86. Fig.8 shows the position of the continental plates on the cube faces for the optimal orientation of the base cube.

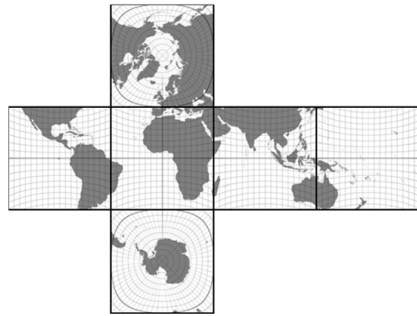


Figure 7. Cube-map faces for the initial base cube orientation (without rotations)

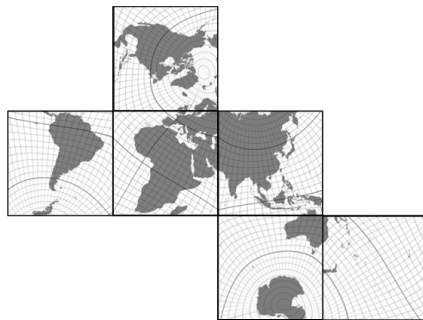


Figure 8. Cube-map faces for the optimal base cube orientation reducing landmass distortion

V. CONCLUSIONS

The need to store and organize large amounts of multiresolution geospatial data bring into play DGGs as a powerful concept. Geodetic DGGs based on a cube, despite their relatively large distortion of the stored data, have a great potential to be accepted by a wide range of users due to the simplicity of implementation.

The effect of distortion can be reduced, to some extent, by choosing the appropriate mapping of the ellipsoid to the sphere and the orientation of the base cube. Given the almost spherical shape of the planet Earth, the choice of auxiliary latitude does not significantly affect the reduction of distortion imposed by ellipsoids to the sphere mapping. However, it is desirable to use the appropriate auxiliary latitude according to the type of projection, to preserve certain properties. For conformal projections, it is desirable to use geocentric latitude, while for equal-area projections it is desirable to choose authalic latitude. As the authalic latitude is very complex to compute, requires more iterations for the inverse transformation, and even with more iterations loses precision near the poles, an approximate function is proposed in this paper that eliminates all these shortcomings.

The orientation of the base cube cannot affect the overall distortion, but it can significantly reduce their

impact on specific areas of interest. We have shown that by appropriate rotation the average angular distortion of continental plates can be reduced by almost 20% in the case of adjusted spherical cubes, while the area distortion is reduced by a much more modest 3%.

The proposed methods are part of the measures that should pave the way for enhanced DGGs based on spherical cubes. Further research will be focused on other aspects of DGGs, such as hierarchical spatial partitioning of cube pages, consideration of characteristics and efficiency of individual projections of spherical cubes, as well as finding an efficient method for visualization of such organized spatial data.

ACKNOWLEDGMENT

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

- [1] Open Geospatial Consortium, "Topic 21: Discrete Global Grid Systems Abstract Specification", 2017.
- [2] A. J. Kimerling, K. Sahr, D. White, and L. Song, "Comparing Geometrical Properties of Global Grids", *Cartography and Geographic Information Science* Vol. 26, No.4, 1999, pp. 271-288
- [3] K. Sahr, D. White and A. J. Kimerling, "Geodesic Discrete Global Grid Systems," *Cartography and Geographic Information Science* Vol. 30, No.2, 2003, pp. 121-134.
- [4] F. Chan and E. O'Neill, "Feasibility study of a quadrilateralized spherical cube earth data base," Tech. Report EPRF 2-75 (CSC), Environmental Prediction Research Facility, Apr. 1975.
- [5] E. O'Neill and R. Laubscher, "Extended studies of a quadrilateralized spherical cube earth data base," Tech. Report NEPRF 3-76 (CSC), Naval Environmental Prediction Research Facility, May 1976.
- [6] H. Alborzi and H. Samet, "Augmenting SAND with a spherical data model," In Proceedings of the First International Conference on Discrete Global Grids, Santa Barbara, CA, USA, 26-28 March 2000.
- [7] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, "HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere," *The Astrophysical Journal*, 622, 2005, pp. 759-771.
- [8] A. M. Amiri, F. Samavati and P. Peterson, "Categorization and Conversions for Indexing Methods of Discrete Global Grid Systems," *ISPRS International Journal of Geo-Information*, 4, 320-336, 2015.
- [9] A. Dimitrijević, M. Lambers and D. Rančić, "Comparison of Spherical Cube Map Projections Used in Planet-Sized Terrain Rendering," *Facta Universitatis, Series: Mathematics and Informatics* 31(2), 2016, pp. 259-297.
- [10] R. Lerbour, *Adaptive streaming and rendering of large terrains*, PhD thesis, Université de Rennes 1, 12 2009.
- [11] R. Lerbour, J.-E. Marvie and P. Gautron, "Adaptive real-time rendering of planetary terrains", in Full Paper Proc. Int. Conf. Computer Graphics, Visualization and Computer Vision (WSCG), 2010, pp. 89-96.
- [12] C. M. Grimm, B. Niebruegge, "Continuous cube mapping," *Journal of Graphics, GPU, and Game Tools* Vol.12, No.4, 2007, pp. 25-34, DOI 10.1080/2151237X.2007.10129250
- [13] O. Adams, "Latitude developments connected with geodesy and cartography: with tables including a table for Lambert equal-area meridional projection," U.S. Coast and Geodetic Survey Spec. Pub. No. 67, 1921.
- [14] J. Snyder, "Map projections: A working manual," Professional Paper No.1395, US Geological Survey, 1987.



Survey of cube mapping methods in interactive computer graphics

M. Lambers¹

Published online: 12 June 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

The standard cube mapping technique implemented in graphics pipelines, while useful in many scenarios, has significant shortcomings for important application areas in interactive computer graphics, e.g., dynamic environment mapping, omnidirectional shadow maps, or planetary-scale terrain rendering. Many alternative mapping methods have been proposed over the years with the purpose of reducing area and/or angular distortions. In this paper, we give an overview of methods suitable for interactive applications and analyze their properties. Furthermore, we evaluate a set of additional transformation functions and identify a simple new method with favorable distortion properties.

Keywords Cube maps · Environment maps · Distortion

1 Introduction

Mappings between a sphere surface and a planar surface have been studied for centuries, starting with world maps and celestial maps [26]. Since the sphere surface is not developable, such mappings always exhibit either area or angular distortions, and often both. Equal-area mappings preserve area ratios, at the cost of large angular distortions, and conformal mappings preserve angles locally, at the cost of large area distortions.

Mapping the complete sphere surface to a circle or square results in strong distortions [15]. Mapping portions of the sphere surface to corresponding faces of a polyhedron significantly lowers these distortions as the number of faces grows [27], at the cost of interruptions at the face boundaries that often manifest themselves as C^1 discontinuities of the mapping function.

In computer graphics, especially in interactive applications, using a cube as the polyhedron is of particular interest, since the number of faces is low and each face is a square, which eases the management of image-like data as well as the application of hierarchical methods such as quad trees or

mipmaps. The cube map method implemented by standard graphics pipelines is the simplest form of mapping between cube and sphere surface. It is equivalent to gnomonic projection for each cube face and exhibits strong area and angular distortions.

Application areas for mappings between cube and sphere surfaces in interactive computer graphics include dynamic environmental maps for illumination [12] and shadow computations [25], planetary-scale terrain rendering [6], and procedural texturing [32]. Though these areas seem diverse, they all map image-like data that are available on a sphere surface to a cube surface, and during rendering sample that cube surface to reconstruct the original sphere surface data. In the case of environment mapping, the data on the sphere surface are given by infinite views in all directions from a single point. In the case of planetary rendering or procedural texturing, the data are directly associated with a sphere surface. In both cases, the methods for forward mapping (sphere to cube) and inverse mapping (cube to sphere) are the same.

In all the aforementioned application areas, alternatives to the standard cube map have been proposed to avoid its distortion problems: With reduced map distortions, lower cube map resolutions can be used to achieve a comparable sampling quality during rendering, resulting in lower memory consumption and computational costs.

This paper gives an overview of the methods that have been proposed, with a focus on applications in interactive computer graphics. Formulas for forward and inverse mapping are given, and the methods are analyzed and compared

Electronic supplementary material The online version of this article (<https://doi.org/10.1007/s00371-019-01708-4>) contains supplementary material, which is available to authorized users.

M. Lambers
martin.lambers@uni-siegen.de

¹ Computer Graphics Group, University of Siegen,
57076 Siegen, Germany

regarding key properties. Furthermore, a new method is identified in a set of candidate methods inspired by previous approaches. This new method combines simplicity with favorable distortion properties.

Section 3 details the needs of example application areas and resulting requirements for cube mapping methods. Section 4 categorizes the known mapping methods and provides implementation details. Section 5 provides numerical analysis results for relevant methods. Section 6 concludes with recommendations. The supplementary material contains C++ source code that implements all mapping and analysis methods used in this paper.

2 Related work

In this paper, we focus on mapping methods for use in interactive computer graphics applications, specifically those that sample cube maps for rendering purposes. This requires that the mapping function is continuous along all cube edges, which excludes, e.g., the equal-area mapping method proposed by Arvo [1] since it exhibits discontinuities along several cube edges [32]. Furthermore, the subdivision of the sphere into six areas that are subsequently mapped to cube faces must match the standard cube map subdivision so that common graphics pipeline functionality can be leveraged to access and filter cube maps. This excludes the HEALPix cube map scheme [4] and variants [31], and the isocube map [30], which use six areas of equal size but differing shape (polar versus equatorial areas).

Furthermore, we exclude methods that have prohibitively high computational costs or require iterative approximations. This exclusion especially affects mapping methods that are *exactly* equal area or conformal: Equal-area mappings require iterative approximations [11,27] or complex and costly mathematical expressions [24] in either the forward or inverse mapping (the QSC method described in Sect. 4.2, despite its complexity, is the cheapest equal-area method and is included here because it has been used in planetary terrain rendering), and conformal mappings are based on Jacobian elliptic functions [18] or Taylor series approximations thereof [22]. Other affected mappings include the method used in the Outerra planetary 3D engine [13] which requires iterative computations [6], the original COBE method [5] which suffers from inversion precision problems [3,6], and the polynomial and COBE-variant methods proposed by Zucker and Higashi which only achieve acceptable inversion precision using iterative refinement methods [32].

The methods presented in this paper provide tradeoffs between area and angular distortions, while keeping complexity and computational costs low.

Zucker and Higashi compared a few of the methods described in this paper with respect to area distortions and

computational costs, but did not consider angular distortions [32]. We consider angular distortions, too, because their impact on sampling quality is not negligible [6,12,14,28]. Furthermore, while Zucker and Higashi aimed to minimize the area distortion RSME, we aim to minimize the maximum distortion errors, i.e., to optimize the worst case behavior.

A different small subset of the methods described in this paper was compared by Lambers and Kolb [16] and Dimitrijević et al. [6], but with a strong focus on the needs of planetary-scale terrain rendering. Consequently, Dimitrijević et al. apply quality measures that were derived from a method to use textures in this context. In contrast, we address a more general application field and therefore use more general quality measurement methods based on the standard analysis of Tissot's indicatrix [26].

We include additional methods from the field of environment mapping into our analysis (continuous cube [10] and unicube [12]), and we systematically evaluate a set of new methods inspired by previous approaches, which have favorable distortion properties while being simple to implement and cheap to compute.

3 Application area requirements

Gathering of information from a cube map works similarly for all application areas. Sampling a standard cube map is typically done by specifying a lookup vector \mathbf{d} . From this vector, the cube face and the coordinates on that face are easily determined. Transforming a lookup vector \mathbf{d} from standard cube map space to one of the alternative cube map spaces described in Sect. 4 requires the forward transformation function f .

Application requirements differ in the cube map creation step. For applications in pseudorandom sample point distribution [32] or in planetary terrain rendering where the cube map is created by sampling image-like data given in some other map space [16], the inverse transformation f^{-1} is needed: For each sample point in the chosen cube map space, first the standard cube map space coordinates are computed using f^{-1} , and these are then transformed to the input map space to sample the original data. In this case, it is important that the inverse of f is numerically precise to avoid sampling at wrong positions during the rendering stage. This is why we excluded methods relying on iterative approximations.

If, on the other hand, the cube map is created by rendering geometry into the cube faces, as in dynamic environment mapping [12] or for omnidirectional shadow maps, then only the forward transformation f is required. When rendering into a cube face, the x and y components of the normalized device coordinates are equivalent to the standard cube map coordinates for the current cube face, and accordingly can be transformed to an alternative cube map space using the forward transformation f in a vertex shader.

The problem with this approach is that straight lines do not map to straight lines anymore, as if one would render onto a curved surface instead of a plane [8]. Similar problems arise with parabolic maps for shadow mapping [25] or with lens distortion rendering via geometry preprocessing [17]. Since the error that is introduced grows with the screen space line length, finely tessellated geometry in the scene is required to keep that error small. Another problem, related to the first one, is that straight lines that cross cube faces may appear to be discontinuous. The remedy proposed by Ho et al. for their unicube method [12] can be applied to any of the mapping methods listed in this paper.

In interactive computer graphics applications, the function f is typically implemented in a shader and should therefore be cheap to compute. To achieve good sampling quality, area distortions should be low. Angular distortions should not be too high either since they also affect sampling quality [6, 12, 14, 28]. It therefore makes sense to chose a method with low maximum area distortion error and acceptable angular distortions to guarantee a minimum sampling quality.

4 Cube map methods

In this section, we describe all methods in detail, starting with the standard cube map as implemented in graphics pipelines. Subsequent methods are listed in chronological order.

Analogous to Zucker and Higashi [32], we formulate all mapping methods as modifications of the standard cube map approach. This allows to leverage the hardware-accelerated graphics pipeline functionality to get cube map coordinates that then only need to be adjusted.

4.1 Standard cube map

The standard cube map implemented in most graphics pipelines was proposed by Greene [9]. It provides a mapping between the unit sphere and the unit cube surfaces by applying gnomonic projection centered on each cube face, i.e., on the Cartesian coordinate axes: Sphere surface coordinates (x, y, z) are mapped to coordinates $(u, v) \in [-1, +1]^2$ on cube face $i \in \{0, \dots, 5\}$, with i corresponding to $+x, -x, +y, -y, +z, -z$.

A great advantage of the standard cube map is that cube maps can be easily created in standard graphics pipelines by rendering one image for each cube side. This is equivalent of shooting rays from the origin to form equidistant grids on the cube faces, and mapping the intersections with the cube and sphere surfaces to each other. See the left side of Fig. 1.

Gnomonic projection is known to suffer from area and angular distortions that grow rapidly with increasing distance from the projection center [26] (in our case, the centers of each cube face). The alternative mappings described in the

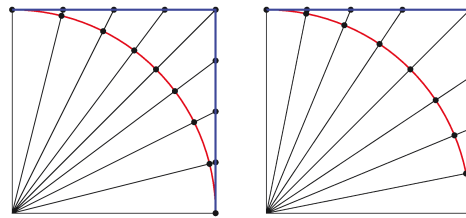


Fig. 1 Gnomonic projection (left) use even sample distances on the cube (blue), resulting in uneven sample distances on the sphere (red). With an adjustment function (right), sample distances are more evenly distributed on the sphere

remainder of this section all strive to reduce these distortions. However, as a consequence, the creation of cube maps by rendering images becomes more complex; see Sect. 3 for details.

We describe each method in terms of modifying the standard cube map coordinates (u, v) on one cube face to arrive at new coordinates (u', v') on the same cube face, either using a univariate function $f : [-1, 1] \rightarrow [-1, 1]$ applied equally to both u and v , or using a bivariate function $f : [-1, 1]^2 \rightarrow [-1, 1]^2$ applied to (u, v) .

In practice, univariate functions f can be applied to all components of the vector $\frac{\mathbf{d}}{\max\{|d_x|, |d_y|, |d_z|\}}$, where d is the cube map lookup direction. This works since $f(-1) = -1$ and $f(1) = 1$, so only the cube face coordinates are actually modified [12].

4.2 Quadrilateralized spherical cube

O'Neill and Laubscher [21] developed the quadrilateralized spherical cube (QSC) model based on previous work by Chan and O'Neill [5]. They inscribed a cube to a sphere and defined hierarchical structures on each cube side for data storage. For that purpose, they designed the QSC projection to be equal area and at the same time limit angular distortions. In computer graphics, QSC has been used in planetary terrain rendering [16]. Although its computation is comparably complex and costly, we include this method here since it is the only known equal-area method with an analytical inverse.

$$f \begin{pmatrix} u \\ v \end{pmatrix} = \tan(v) \begin{pmatrix} \cos(\mu) \\ \sin(\mu) \end{pmatrix}, \text{ with}$$

$$\varphi = \cos^{-1} \left(\frac{1}{\sqrt{u^2 + v^2 + 1}} \right)$$

$$\theta = \text{atan2}(u, -v)$$

$$\mu = \tan^{-1} \left(\frac{12}{\pi} \right)$$

$$\left(\theta + \cos^{-1} \left(\sin(\theta) \cos \left(\frac{\pi}{4} \right) \right) - \frac{\pi}{2} \right)$$

$$\begin{aligned}
 t &= 1 - \cos\left(\tan^{-1}\left(\frac{1}{\cos(\theta)}\right)\right) \\
 \tan(v) &= \sqrt{\frac{1 - \cos(\varphi)}{\cos^2(\mu)t}} \\
 f^{-1}\begin{pmatrix} u' \\ v' \end{pmatrix} &= \tan(\varphi) \begin{pmatrix} \sin(\theta) \\ -\cos(\theta) \end{pmatrix}, \text{ with} \\
 \tan(v) &= \sqrt{u'^2 + v'^2} \\
 \mu &= \text{atan2}(v', u') \\
 \theta &= \tan^{-1}\left(\frac{\sin\left(\frac{\pi}{12}\tan(\mu)\right)}{\cos\left(\frac{\pi}{12}\tan(\mu)\right) - \frac{1}{\sqrt{2}}}\right) \\
 t &= 1 - \cos\left(\tan^{-1}\left(\frac{1}{\cos(\theta)}\right)\right) \\
 \varphi &= \cos^{-1}(1 - \cos^2(\mu)\tan^2(v)t) \tag{1}
 \end{aligned}$$

The above formulae assume $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, i.e., they apply only to one quarter of a cube face. The other quarters are handled by rotating them into the quarter of definition. This introduces \mathcal{C}^1 discontinuities at the cube face diagonals. While this method exhibits no area distortions, the average angular distortions of this equal-area method are quite strong. However, the maximum angular distortion error is lower than for all other methods.

4.3 Tangent adjustment

The tangent adjustment of the standard cube map modifies the cube map coordinates in the following way:

$$\begin{aligned}
 f(w) &= \frac{\pi}{4} \tan^{-1}(w) \\
 f^{-1}(w') &= \tan\left(w' \frac{\pi}{4}\right) \tag{2}
 \end{aligned}$$

This method has been rediscovered repeatedly. In the context of computer graphics, it was used by Lerbour et al. for planetary terrain rendering [19] and by Bitterli et al. in dynamic environment mapping [2]. Earlier uses of the method can be found in other fields [22,23].

The motivation for this method is that points on the sphere are distributed more evenly, thereby reducing distortions. See Fig. 1.

Zucker and Higashi introduce a parameter c [32]:

$$\begin{aligned}
 f(w) &= \frac{\tan^{-1}(cw)}{\tan^{-1}(c)} \\
 f^{-1}(w') &= \frac{\tan(w' \tan^{-1}(c))}{c}
 \end{aligned}$$

They numerically find $c \approx 1.1823$ to be optimal in terms of area distortion RMSE reduction, while Eq. 2 uses $c = 1$.

4.4 Nowell's method

Nowell devised a mapping from cube surface to sphere surface based on Cartesian coordinates [20,29]. The formulas below assume that a given point (x, y, z) on the sphere surface maps to (u, v) on the cube face at $x = 1$; the other faces are handled by symmetry.

$$\begin{aligned}
 f(x, y, z) &= \begin{pmatrix} \frac{\text{sgn}(y)}{\sqrt{2}} \sqrt{t + 2y^2 - 2z^2 + 3} \\ \frac{\text{sgn}(z)}{\sqrt{2}} \sqrt{t - 2y^2 + 2z^2 + 3} \end{pmatrix}, \text{ with} \\
 t &= -\sqrt{(2z^2 - 2y^2 - 3)^2 - 24y^2} \\
 f^{-1}(u', v') &= \begin{pmatrix} \sqrt{1 - \frac{u'^2}{2} - \frac{v'^2}{2} + \frac{u'^2 v'^2}{3}} \\ u' \sqrt{\frac{1 - v'^2}{2} + \frac{v'^2}{3}} \\ v' \sqrt{\frac{1 - u'^2}{2} + \frac{u'^2}{3}} \end{pmatrix} \tag{3}
 \end{aligned}$$

This method has been used in environment mapping [29]. Its area and angular distortions are a clear improvement over standard cube maps, but better methods exist. Furthermore, the formulation in terms of Cartesian coordinates complicates its application in a graphics pipeline because the hardware support for cube mapping cannot be used (or has to be undone).

4.5 Continuous cube

The continuous cube method was introduced by Grimm and Niebruegge for the purpose of environment mapping [10]. The original formulation is based on Cartesian coordinates. Here, we give the formulation based on cube face coordinates so that the standard cube map functionality can be used:

$$\begin{aligned}
 f\begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} \frac{\tan^{-1}\left(\frac{u}{\sqrt{2}}\right)}{\sin^{-1}\left(\frac{1}{\sqrt{3}}\right)} \\ \frac{\tan^{-1}(v \cdot \cos(\tan^{-1}(u)))}{\sin^{-1}\left(\frac{1}{\sqrt{2+u^2}}\right)} \end{pmatrix} \\
 f^{-1}\begin{pmatrix} u' \\ v' \end{pmatrix} &= \begin{pmatrix} t \\ \tan\left(v' \sin^{-1}\left(\frac{1}{\sqrt{2+t^2}}\right)\right) \\ \frac{t}{\cos(\tan^{-1}(t))} \end{pmatrix}, \text{ with} \\
 t &= \sqrt{2} \tan\left(u' \sin^{-1}\left(\frac{1}{\sqrt{3}}\right)\right) \tag{4}
 \end{aligned}$$

Similar to the Nowell method, the area and angular distortions of the continuous cube are an improvement over standard cube maps.

4.6 Unicube

The unicube map was proposed by Ho et al. [12] as an improvement over the previous isocube map [30]. The fol-

lowing adjustment function was derived to improve sampling uniformity compared to both the standard cube map and the isocube map:

$$f(w) = \frac{6}{\pi} \sin^{-1} \left(\frac{w}{\sqrt{2w^2 + 2}} \right)$$

$$f^{-1}(w') = \frac{\sin \left(\frac{\pi}{6} w' \right)}{\sqrt{\frac{1}{2} - \sin^2 \left(\frac{\pi}{6} w' \right)}} \quad (5)$$

The unicube method has low area distortions.

4.7 Everitt's method

Zucker and Higashi [32] extracted the following adjustment function from source code provided by Everitt [7]:

$$f(w) = w(c + (1 - c)|w|)$$

$$f^{-1}(w') = \text{sgn}(w') \left(\frac{c - \sqrt{c^2 - 4(c - 1)|w'|}}{2(c - 1)} \right) \quad (6)$$

While Everitt originally used $c = 1.5$ and $c = 1.375$, Zucker and Higashi determined numerically that $c \approx 1.4511$ is optimal in terms of area distortion RMSE minimization.

4.8 Sigmoid adjustment (new method)

The aforementioned adjustment functions and their inversions all take the shape of a sigmoid function $[-1, 1] \rightarrow [-1, 1]$. In order to identify adjustment function candidates with favorable properties, we took different analytically invertible sigmoid functions and introduced a parameter c to each of them to tune their behavior. In particular, we tested algebraic sigmoid functions

$$f(w) = \frac{w\sqrt{1+c^2}}{\sqrt{1+c^2w^2}},$$

logistic sigmoid functions

$$f(w) = \frac{2e^c + 2}{e^c - 1} \left(\frac{1}{1 + e^{-cw}} - \frac{1}{2} \right),$$

smoothstep functions

$$f(w) = \frac{\frac{1}{2}(3(cw + 1)^2 - (cw + 1)^3) - 1}{\frac{1}{2}(3(c + 1)^2 - (c + 1)^3) - 1}, \quad c \in (0, 1],$$

the hyperbolic tangent function

$$f(w) = \frac{\tanh(cw)}{\tanh(c)},$$

and the parameterized tangent function described in Sect. 4.3.

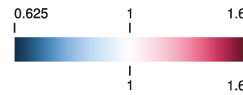


Fig. 2 Color map for D_A (top) and D_I (bottom) in Tables 1 and 2. Note that $D_I \geq 1$. Values outside the range displayed here are clamped in Tables 1 and 2

Varying the parameter c can aim to minimize either the maximum error or the RMSE of either area or angular distortions. The resulting values for c will differ; see the first two entries in Table 2 for an example.

We minimize the maximum area distortion error, thus improving the worst case behavior of each mapping method. Note that all sigmoid function variations have nearly the same maximum angular distortion error (reached at the cube face borders, where the difference to the standard cube map method reaches its minimum), except for the smoothstep function, which is worse. Therefore, trying to minimize this error does not make sense. Note also that while the area and angular RMSEs are not minimal when minimizing the maximum area distortion error, they are still in an acceptable range.

As a result of our tests, we recommend the algebraic sigmoid function with parameter $c = 0.8700$. See Sect. 5 for details. This function has the additional advantage of being cheap to compute:

$$f(w) = w \sqrt{\frac{1+c^2}{1+c^2w^2}}$$

$$f^{-1}(w') = \frac{w'}{\sqrt{1+c^2-c^2w'^2}} \quad (7)$$

5 Evaluation

To analyze the area and angular distortions of all cube map variants, we use the measures D_A and D_I [15] derived from Tissot's indicatrix [26]. The basic idea of these measurements is that a map projection maps an infinitesimal circle on the sphere onto an infinitesimal ellipse on the map. From the semi-major axis a and semi-minor axis b of this ellipse, several measurements of projection quality can be derived, e.g., the local scale factor $s = ab$ which is equivalent to the determinant of the Jacobian matrix.

$$D_A = \frac{ab}{R}, \quad R = \frac{4}{\frac{4\pi}{6}} = \frac{6}{\pi}$$

$$D_I = \frac{a}{b} \quad (8)$$

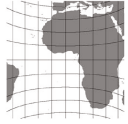
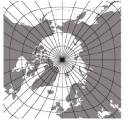


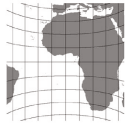
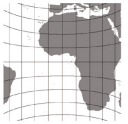



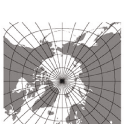


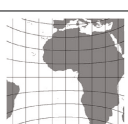
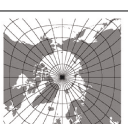
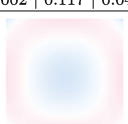

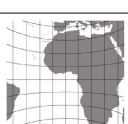
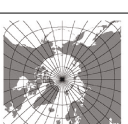
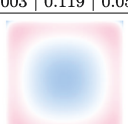
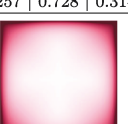
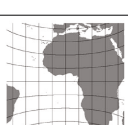
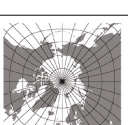
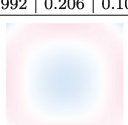
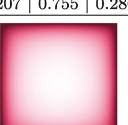
Table 1 Comparison of cube map variants proposed in the literature

Method	Example cube face at $x=1$	Example cube face at $z=1$	D_A (ideal: 1) avg max err RMSE	D_I (ideal: 1) avg max err RMSE	Remarks
Standard cube map			 1.153 1.710 0.464	 1.281 0.730 0.324	
QSC			1.000 0.000 0.000	 1.329 0.548 0.346	No area distortions. Lowest maximum angular distortion. Expensive. C^1 discontinuities at face diagonals.
Tangent Adjustment			 1.008 0.199 0.089	 1.234 0.729 0.280	Comparably low distortions. Simple.
Nowell's Method			 1.001 0.205 0.034	 1.235 0.727 0.295	Comparably low distortions. Works on cartesian coordinates.
Continuous Cube			 1.030 0.592 0.183	 1.179 0.728 0.224	Comparably high area distortions. Expensive.
UniCube			 1.001 0.171 0.038	 1.288 0.728 0.348	Low area distortions. High angular distortion RMSE.
Everitt's Method with $c = 1.4511$			 1.003 0.183 0.053	 1.298 0.860 0.349	Low area distortions. High maximum angular distortion.

R is the ratio between the area of a cube face and the sixth part of the sphere surface area and is used for normalization. The optimal value for both D_A and D_I is 1. Since $a \geq b$, $D_I \geq 1$.

These two distortion measurements are color-coded as shown in Fig. 2. Note that in Tables 1 and 2, values outside the range covered by the color map are clamped.

Table 2 Comparison of parameterized sigmoid adjustment functions

Method	Example cube face at $x=1$	Example cube face at $z=1$	D_A (ideal: 1) avg max err RMSE	D_I (ideal: 1) avg max err RMSE	Remarks
Tangent Sigmoid with $c = 1.1823$ [32]			 1.002 0.143 0.041	 1.290 0.728 0.345	c chosen to minimize area distortion RMSE.
Tangent Sigmoid with $c = 1.1968$			 1.002 0.138 0.041	 1.295 0.728 0.351	c chosen to minimize the maximum area distortion error.
Algebraic Sigmoid with $c = 0.8700$ (recommended method)			 1.002 0.117 0.044	 1.271 0.728 0.327	Lowest maximum area distortion error. Simple and cheap computation.
Logistic Sigmoid with $c = 1.9443$			 1.003 0.119 0.055	 1.257 0.728 0.314	Comparable to algebraic adjustment.
Smoothstep Sigmoid with $c = 0.7507$			 0.992 0.206 0.101	 1.207 0.755 0.280	Worst of the sigmoid function candidates.
Hyperbolic Tangent Sigmoid with $c = 0.9721$			 1.003 0.119 0.055	 1.257 0.728 0.314	Comparable to algebraic adjustment.

The top row shows the tangent adjustment variant proposed by Zucker and Higashi [32] (Sect. 4.3). The following rows show sigmoid function variations with parameter c chosen to minimize the maximum area distortion error (Sect. 4.8)

The results of the distortion evaluations, along with example maps, are listed in Table 1 (for all methods previously proposed in the literature) and Table 2 (for all tested alternative sigmoid functions). For the sigmoid functions, we manually varied parameter c to minimize the maximum area distortion.

The QSC method is area preserving and has the lowest maximum angular distortion of all tested methods, making it the method with the highest quality by a wide margin. However, its complex and costly computations limit its usefulness for interactive applications.

The sigmoid adjustment functions are simpler and cheaper to compute than the alternative methods proposed in the lit-

erature (with the exception of Everitt's method), and they all achieve a lower maximum angular distortion error. They all perform similarly, with the same maximum angular distortion error that is largely independent of the parameter c , and comparable area and angular distortions.

We recommend the algebraic sigmoid adjustment function with parameter $c = 0.87$ since it is simple, cheap to compute, and has the lowest maximum area distortion error while both area and angular distortion RMSE are in an acceptable range.

Furthermore, we tested the numerical accuracy of consecutive forward and inverse transformation for each point in a regular grid of size 512×512 representing one cube face. Since all methods listed in this paper have an analytical inverse, the distance between original point and the result of consecutive transformation is very small, in the order of 10^{-6} for single precision floating point, except for the Nowell method, where it is in the order of 10^{-4} .

6 Conclusion

This paper gives an overview of alternative cube mapping methods used in interactive computer graphics applications. Most of the methods proposed in the literature, e.g., continuous cube and unicube, are outperformed by simpler methods found via systematic evaluation of sigmoid functions, e.g., the algebraic sigmoid function. The best performing mapping method is QSC, which preserves area and additionally limits angular distortions. Since its computations are rather complex and costly, the sigmoid functions provide a good compromise for interactive applications. However, the exact effects of different cube map methods on rendering speed and other system attributes depend on the application area and need to be evaluated in the application context.

Supplementary material

The supplementary material includes C++ code implementing all mapping methods, along with analysis code that can be used to reproduce all results shown in this paper.

Acknowledgements The polygonal world map data used in the example maps in Tables 1 and 2 are provided by Bjorn Sandvik, http://thematicmapping.org/downloads/world_borders.php, license CC BY-SA 3.0.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest.

References

- Arvo, J.: Stratified sampling of 2-manifolds. In: SIGGRAPH Course Notes (2001)
- Bitterli, B., Novák, J., Jaros, W.: Portal-masked environment map sampling. *Comput. Graph. Forum* **34**(4), 13–19 (2015). <https://doi.org/10.1111/cgf.12674>
- Calabretta, M., Greisen, E.: Representations of celestial coordinates in FITS. *Astron. Astrophys.* **395**(3), 1077–1122 (2002). <https://doi.org/10.1051/0004-6361:20021327>
- Calabretta, M.R., Roukema, B.F.: Mapping on the HEALPix grid. *Mon. Not. R. Astron. Soc.* **381**(2), 865–872 (2007). <https://doi.org/10.1111/j.1365-2966.2007.12297.x>
- Chan, F., O'Neill, E.: Feasibility study of a quadrilateralized spherical cube earth data base. In: Technical Report EPRF 2-75 (CSC), Environmental Prediction Research Facility. <https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/ADA010232.xhtml> (1975). Accessed 28 Nov 2018
- Dimitrijević, A., Lambers, M., Rančić, D.: Comparison of spherical cube map projections used in planet-sized terrain rendering. *Facta Univ., Ser.: Math. Inform.* **31**(2), 259–297 (2016)
- Everitt, C.: "Projection" repository. <https://github.com/casseveritt/projection/> (2016). Accessed 28 Nov 2018
- Gascuel, J.D., Holzschuch, N., Fournier, G., Péroche, B.: Fast non-linear projections using graphics hardware. In: Proceedings Symposium Interactive 3D Graphics and Games (I3D), pp. 107–114 (2008). <https://doi.org/10.1145/1342250.1342267>
- Greene, N.: Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.* **6**(11), 21–29 (1986). <https://doi.org/10.1109/MCG.1986.276658>
- Grimm, C.M., Niebruegge, B.: Continuous cube mapping. *J. Graph., GPU, Game Tools* **12**(4), 25–34 (2007). <https://doi.org/10.1080/2151237X.2007.10129250>
- Harrison, E., Mahdavi-Amiri, A., Samavati, F.: Optimization of inverse Snyder polyhedral projection. In: International Conference on Cyberworlds, pp. 136–143 (2011). <https://doi.org/10.1109/CW.2011.36>
- Ho, T.Y., Wan, L., Leung, C.S., Lam, P.M., Wong, T.T.: Unicube for dynamic environment mapping. *IEEE Trans. Vis. Comput. Graph.* **17**(1), 51–63 (2011). <https://doi.org/10.1109/TVCG.2009.205>
- Kemen, B., Hrabcak, L.: Outerra. <http://www.outerra.com> (2014). Accessed 28 Nov 2018
- Kooima, R., Leigh, J., Johnson, A., Roberts, D., SubbaRao, M., DeFanti, T.A.: Planetary-scale terrain composition. *IEEE Trans. Vis. Comput. Graph.* **15**(5), 719–733 (2009). <https://doi.org/10.1109/TVCG.2009.43>
- Lambers, M.: Mappings between sphere, disc, and square. *J. Comput. Graph. Tech.* **5**(2), 1–21 (2016)
- Lambers, M., Kolb, A.: Ellipsoidal cube maps for accurate rendering of planetary-scale terrain data. In: Proceedings Pacific Graphics (Short Papers), pp. 5–10 (2012). <https://doi.org/10.2312/PE/PG/PG2012short/005-010>
- Lambers, M., Sommerhoff, H., Kolb, A.: Realistic lens distortion rendering. In: Proceedings International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG). <http://wscg.zcu.cz/wscg2018/2018-WSCG-Papers-Separated.html> (2018). Accessed 28 Nov 2018
- Lee, L.: Conformal projections based on Jacobian elliptic functions. *Cartogr.: Int. J. Geogr. Inf. Geovisualization* **13**(1), 67–101 (1976). <https://doi.org/10.3138/X687-1574-4325-WM62>
- Lerbour, R., Marvie, J.E., Gautron, P.: Adaptive real-time rendering of planetary terrains. In: Proceedings International Conference Computer Graphics, Visualization and Computer Vision (WSCG). http://wscg.zcu.cz/WSCG2010/Papers_2010/!_2010_FULL-proceedings.pdf (2010). Accessed 28 Nov 2018

20. Nowell, P.: Mapping a cube to a sphere. <http://mathproofs.blogspot.de/2005/07/mapping-cube-to-sphere.html> (2005). Accessed 28 Nov 2018
21. O'Neill, E., Laubscher, R.: Extended studies of a quadrilateralized spherical cube earth data base. Technical Report NEPRF 3-76 (CSC), Naval Environmental Prediction Research Facility. <http://www.dtic.mil/docs/citations/ADA026294> (1976). Accessed 28 Nov 2018
22. Rančić, M., Purser, R.J., Mesinger, F.: A global shallow-water model using an expanded spherical cube: gnomonic versus conformal coordinates. *Q. J. R. Meteorol. Soc.* **122**(532), 959–982 (1996). <https://doi.org/10.1002/qj.49712253209>
23. Ronchi, C., Iacono, R., Paolucci, P.S.: The “cubed sphere”: a new method for the solution of partial differential equations in spherical geometry. *J. Comput. Phys.* **124**(1), 93–114 (1996). <https://doi.org/10.1006/jcph.1996.0047>
24. Roşca, D., Plonka, G.: Uniform spherical grids via equal area projection from the cube to the sphere. *J. Comput. Appl. Math.* **236**(6), 1033–1041 (2011). <https://doi.org/10.1016/j.cam.2011.07.009>
25. Scherzer, D., Wimmer, M., Purgathofer, W.: A survey of real-time hard shadow mapping methods. *Comput. Graph. Forum* **30**(1), 169–186 (2011). <https://doi.org/10.1111/j.1467-8659.2010.01841.x>
26. Snyder, J.: *Map Projections—A Working Manual*, Professional Paper, vol. 1395. US Geological Survey (1987). <https://doi.org/10.3133/pp1395>
27. Snyder, J.: An equal-area map projection for polyhedral globes. *Cartographica* **29**(1), 10–21 (1992). <https://doi.org/10.3138/27H7-8K88-4882-1752>
28. Snyder, J., Mitchell, D.: Sampling-efficient mapping of spherical images. In: Microsoft Research Technical Report. <https://www.microsoft.com/en-us/research/publication/sampling-efficient-mapping-spherical-images/> (2001). Accessed 28 Nov 2018
29. Various: Mapping a sphere to a cube. <http://stackoverflow.com/questions/2656899/mapping-a-sphere-to-a-cube> (2010). Accessed 28 Nov 2018
30. Wan, L., Wong, T.T., Leung, C.S.: Isocube: exploiting the cube-map hardware. *IEEE Trans. Vis. Comput. Graph.* **13**(4), 720–731 (2007). <https://doi.org/10.1109/TVCG.2007.1020>
31. Wong, T.T., Wan, L., Leung, C.S., Lam, P.M.: *Shader X4: Advanced Rendering Techniques*, Chap. Real-Time Environment Mapping with Equal Solid-Angle Spherical Quad-Map, pp. 221–233. Charles River Media (2006)
32. Zucker, M., Higashi, Y.: Cube-to-sphere projections for procedural texturing and beyond. *J. Comput. Graph. Tech. (JCGT)* **7**(2), 1–22 (2018)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Martin Lambers received the Diploma degree in mathematics from the University of Münster, Germany, in 2006, and the Dr.Ing. degree in computer science from the University of Siegen, Germany, in 2011. He is currently a senior researcher with the Computer Graphics and Multimedia Systems Group, University of Siegen. His research interests include simulation, processing, analysis, and visualization of multimodal sensor data.



Fast motion estimation for field sequential imaging: Survey and benchmark[☆]

Holger Steiner^a, Hendrik Sommerhoff^b, David Bulczak^b, Norbert Jung^a,
Martin Lambers^b, Andreas Kolb^{b,*}

^aHochschule Bonn–Rhein–Sieg, St. Augustin, Germany

^bUniversity of Siegen, Germany

ARTICLE INFO

Article history:

Received 16 May 2018

Received in revised form 11 July 2019

Accepted 15 July 2019

Available online 25 July 2019

Keywords:

Field sequential imaging

Motion estimation

Optical flow

ABSTRACT

Field sequential (FS) imaging comprises image acquisition systems that capture image channels in temporal sequence in order to provide the final image. A classical application is multispectral imaging. In case of dynamic scenes, the sequential nature of the acquisition imposes motion artifacts, i.e., spatially misaligned images channels. Compensating motion artifacts for this kind of imagery is non-trivial, as common methods for motion estimation rely on the intensity consistency constraint that is violated in FS imaging.

This paper surveys approaches to motion compensation in the context of FS imaging. We focus on accuracy in handling intensity inconsistent data and, secondarily, speed, as FS imaging is commonly done in real-time. We introduce a conceptual classification for algorithmic approaches for motion estimation for FS imagery and discuss known and modified approaches to tackle the intensity inconsistencies between adjacent image channels using image transformation and intensity correction methods. As result, we get a set of 379 variants of motion estimation methods applicable to FS data streams. We evaluate these methods using our benchmark database, which comprises data sets from the Middlebury and the MPI Sintel databases, modified to emulate FS imagery, as well as additionally captured multispectral short wave infrared (SWIR) and sRGB image sequences, as well as simulated Time-of-Flight (ToF) image sequences that consist of four channels (called phase images). In order to quantify the motion estimation techniques, we use a ranking scheme similar to Middlebury and combine it with a run-time evaluation.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Field sequential (FS)¹ imaging systems acquire several *channel* images sequentially at full spatial resolution of the final image. These kind of image acquisition systems mainly appear in *multispectral imaging*, but also in range imaging, e.g. in *Time-of-Flight (ToF)* range imaging. In the case of multispectral imaging, the final image is composed of this set of spectral channels, while in ToF range imaging, the final depth image is computed from the channels (called *phase images* in the ToF context).

Multispectral FS imaging system are capable of capturing high-density spectral information of object surfaces and thus offer several

advantages over grayscale or RGB cameras in applications such as remote sensing, astronomy, agriculture, medicine or food quality control [2], as well as high quality color image reproduction and conservation of art [3]. While *simultaneous* multispectral image acquisition use, e.g., static filters such as the Bayer pattern or beam splitters, multispectral FS imaging systems are realized using, e.g., broad band imagers combined with interchangeable band pass filters mounted on a filter wheel [3–5], electronically tunable filters [6], or active (narrow band) illumination setups [7]. The multispectral FS imaging approaches are more flexible in selecting the spectral bands and allow for the acquisition of a much larger number of spectral channels than simultaneous approaches.

ToF cameras calculate the camera-object distance by estimating the time delay that actively emitted light takes to travel from the light source to the object surface and back to the sensor's pixel. Therefore, the amplitude of the emitted light signal is modulated and the backscattered light signal is correlated at pixel-level in the sensor. It takes at least three different *phase images* (channels in our notation) in order to reconstruct a distance image [8–10].

[☆] This paper has been recommended for acceptance by Sinisa Todorovic.

* Corresponding author.

E-mail address: andreas.kolb@uni-siegen.de (A. Kolb).

URL: <https://www.cg.informatik.uni-siegen.de> (A. Kolb).

¹ Derived from Field Sequential Color Capturing for color imaging [1].

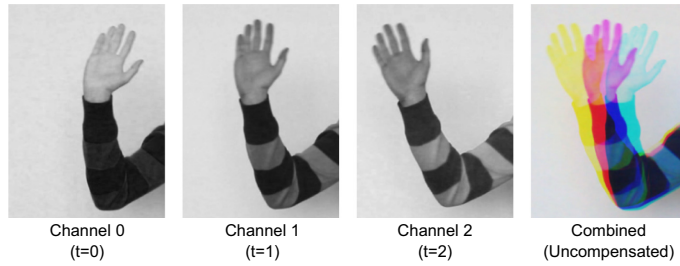


Fig. 1. A waving hand recorded using field-sequential color capturing, with channels captured at subsequent times t . When the channels are combined in a multispectral image, the color breakup effect occurs.

In case of dynamic scenes, all FS image capturing systems suffer from motion artifacts, as moving objects will not match between the different channels; see Fig. 1. Depending on the amount of motion and the application requirements, raw FS imagery cannot be used without compensating the motion artifacts. Although motion estimation has a long and successful history in computer vision, existing motion estimation techniques cannot handle FS imagery properly, as it *strongly violates the intensity consistency assumption* between adjacent channels, which most state of the art motion estimation techniques rely upon [11].

In this paper, we describe simple and generic approaches in order to apply existing motion estimation approaches to FS imagery that, in most cases, incorporate up to three components:

Motion estimation scheme: There are various basic concepts on how to estimate motion within an FS image stream (see Section 3). *Corresponding channel matching (CCM)* methods estimate motion fields between corresponding channels of adjacent images, thus preventing the intensity inconsistency problem at the cost of larger temporal gaps that need to be bridged. In contrast to this, *neighboring channel matching (NCM)* approaches estimate temporally dense motion fields between neighboring channels within an image or across neighboring images, which requires the handling of intensity inconsistency; see also Fig. 2.

Image transformation & intensity correction: Any NCM method needs to handle the intensity inconsistency. This can be either done by transforming the image in another domain (e.g. gradients) or by correcting the intensity by some preprocessing procedure.

Intensity consistent motion estimation: Finally, the motion between several channels within or across the FS images are estimated using a state-of-the-art method (see Section 2).

We present a first thorough analysis and discussion of motion estimation approaches that are applicable to FS imaging systems. As motion compensation for FS imaging primarily makes sense for

real-time image capturing, we mainly focus on online estimation methods. Thus, the performance of any FS motion estimation method is defined by both, *high motion estimation accuracy* and *low processing time*.

This paper provides the following methodological and technical contributions:

- A set of *general concepts for motion estimation schemes* that are applicable to FS imagery, refining the basic principles of *corresponding channel matching (CCM)* and *neighboring channel matching (NCM)* (see Section 3).
- A *benchmark data set* including different test scenarios from both domains, FS multispectral imagery as well as phase image sequences from ToF cameras. These data sets include translational and rotational movements and partially comprise ground truth data. Regarding multispectral imagery, we also include existing data sets such as Middlebury or MPI Sintel (Section 6).
- An *in-depth evaluation* with respect to compensation accuracy and processing time of a large set of FS motion estimation methods comprising the components listed above (Section 7).

As an overall contribution, this paper provides simplified and quantified means to select the most promising methods for motion estimation on FS data depending on sample-based application scenarios.

The remainder of the paper is structured as follows. Section 2 gives an overview on existing motion estimation algorithms based on optical flow and block matching. Section 3 discusses the general approaches applicable to field sequential motion estimation. In Sections 4 and 5, we describe the image transformation and intensity correction schemes that we use in order to compensate for

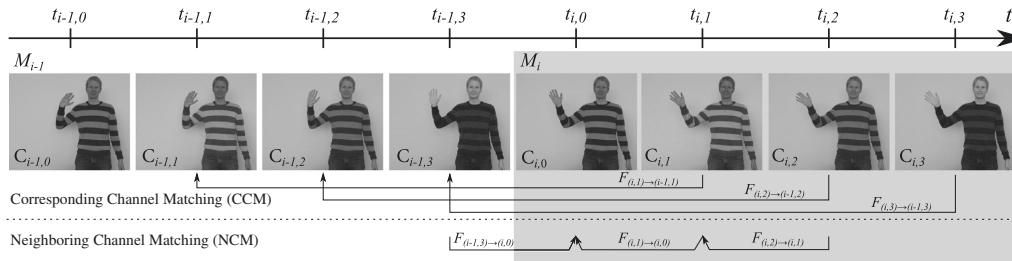


Fig. 2. Flow calculations for two successive FS images with four channels using either corresponding channel matching (CCM) or neighboring channel matching (NCM).

the intensity inconsistency of FS imagery. Section 6 presents the evaluation of all 379 algorithm combinations.

2. State of the art in motion estimation

Optical flow (OF), *Block matching* (BM), and *Deep neural network* based methods are the most prominent classes of approaches to estimate dense motion fields between consecutive images. As a fully comprehensive survey of motion estimation techniques is beyond the scope of this work, interested readers are referred to the work of Fotun et al. [12], to get deeper insight into optical flow computation methods, and to the survey on block-based methods by Jakubowski and Pastuszak [13].

In this paper, we focus on methods that allow for sufficiently fast motion estimation, i.e. for which fast implementations are available or which have the potential to be implemented in a near-to-realtime fashion. Similar as the Middlebury [11] and MPI Sintel [14] benchmarks, this survey and benchmark paper is open to be extended to any motion estimation technique, e.g. for more accurate (and potentially slower) approaches in the future.

The original approaches on the calculation of *optical flow* have been proposed by Horn and Schunck [15] and Lucas and Kanade [16]. They assumed that every change in a pixel's brightness is due to motion. They compute the flow field using brightness gradients and a constraint on motion smoothness. Brox et al. [17] extended this assumption by a gradient constancy constraint to deal with slight changes in brightness and an enhanced smoothness assumption. Another approach by Zach et al. [18] is based on total variation (TV) regularization, using the L^1 norm (TV- L^1) and claims to be very robust against illumination changes and occlusions. Both, Refs. [17] and [18], are available as real-time GPU-based implementation. Werlberger et al. [19] proposed to replace the TV regularization with the Huber norm (Huber- L^1). They presented a library called *FlowLib*, which contains GPU accelerated implementations of their algorithm in different variations. Additionally, Werlberger [20] proposed alternative data terms, representing the structure of the image rather than intensities. Their normalized cross-correlation (NCC), census transform and consistency of gradients approaches provide better compensation for intensity variations.

More modern variants that surpass the accuracy of the aforementioned approaches and could be better suited for FS imagery include the *Large displacement optical flow* (LDOF) presented by Brox and Malik [21] as well as the *EpicFlow* described by Revaud et al. [22]. Both methods deal with the common problem of variational optical flow methods, which tend to select the local minimum closest to the initialization, i.e., a well matching point with the smallest motion. For this purpose, LDOF incorporates descriptor matching techniques into the variational approach to emphasize matches with higher accuracy even in the presence of similar looking image areas. EpicFlow relies on a sparse-to-dense approach which detects and preserves edges. The *FlowFields* method presented by Bailer et al. [23] builds up on the edge preserving interpolation of EpicFlow, but improves on its results by using a new hierarchical correspondence field search strategy based on either census or SIFTflow as data term.

The basic idea of *block matching*, on the other hand, is to divide an image into *macro blocks* of a given block size b and to find the best matching block in a reference image using error functions such as the sum of absolute differences (SAD).

Usually, only translational motion is taken into account. To avoid blocking artifacts at object boundaries, different techniques such as overlapping blocks, adaptive block size, multiscale approaches and filtering have been proposed [24]. The search range can be limited to a maximum displacement range (p -value). A simple full search tests all possible block displacements within this range. More efficient search strategies can be applied, e.g., temporal motion prediction,

which reduce the number of calculations at the cost of accuracy [25]. Due to its high degree of parallelism, BM can be efficiently implemented on GPUs or FPGAs to achieve real-time processing.

Recently, motion estimation based on convolutional *deep neural networks* emerged [26] and soon surpassed OF and BM methods in quality, as demonstrated e.g. by the KITTI benchmark [27]. As acquiring ground truth motion data for training purposes is challenging, unsupervised variants have been proposed [28], but they do not typically reach the same level of quality. In this paper, we evaluate the recent pre-trained networks FlowNet2 [29] and LiteFlowNet [30] in the context of FS imagery.

Ref. [7] is the first work addressing the motion compensation problem for multispectral short wave infrared (SWIR) FS imagery. They estimate forwards and backwards motion fields using state of the art OF methods for each pair of related waveband channels, which are intensity consistent by nature. Due to the interpolation over large time spans, this *corresponding channel matching* (CCM) approach yields rather poor results on scenes involving non-constant motion.

In the context of Time-of-Flight (ToF) cameras, various motion estimation approaches have been proposed. Most of them apply OF [31, 32] or BM [33] on the set of phase images, thus implementing a *neighboring channel matching* (NCM) approach. To deal with the intensity inconsistency problem, they estimate motion on normalized phase intensity images.

A different and computationally very efficient approach to motion compensation has been proposed by Schmidt and Jahne [34]. Their *pixelwise artifact correction* (PAC) method detects motion artifacts on pixel level by assuming temporally smooth intensity variation in the non-motion case: if the first channel contains no discontinuity but at least one of the following channels does, then the pixel is assumed to be affected by motion.

3. Field sequential motion estimation schemes

Consider a field sequential (FS) image stream consisting of images M_i , with $i \in \mathbb{N}$ being a sequential number, which themselves contain n channels $C_{i,w}$, which were acquired at sequential times $t_{i,w}$, $w = 0, \dots, n-1$, with w being the channel index. Furthermore, a discrete and equidistant acquisition time $\Delta t = t_{i,w} - t_{i,w-1}$ is assumed for each channel and a constant acquisition time $T = t_{i,0} - t_{i-1,0} = n\Delta t$ for the full image, as illustrated in Fig. 2. Formally, motion estimation for FS image streams has to compute the displacement vector fields $F_{(i,w) \rightarrow (i,0)}$ between any channels $C_{i,w}$, $w > 0, \dots, n-1$ and the first channel $C_{i,0}$, which serves as reference.

By applying a displacement vector field $F_{(i,w) \rightarrow (i,0)}$ to channel $C_{i,w}$, all pixel values $p(x,y)$ from $C_{i,w}$ are shifted according to the two-dimensional displacement vectors $d(x,y) = F_{(i,w) \rightarrow (i,0)}(x,y)$. In the final "corrected" images $\tilde{C}_{i,w}$, the positions of moving objects will match those in the reference channel $C_{i,0}$, if the motion estimation has been accurate.

When optical flow is calculated directly between adjacent channels of the image sequence, i.e. between $C_{i,w}$ and $C_{i,w+1}$, purely intensity-based optical flow algorithms will produce invalid displacement vectors due to the violation of the intensity consistency assumption. In the following, we describe two fundamental concepts to overcome this problem, i.e., *corresponding channel matching* (CCM; see Section 3.1) and *neighboring channel matching* (NCM; see Section 3.2), and discuss approaches to modify existing motion estimation techniques in order to be applied with either concept.

3.1. Corresponding channel matching (CCM)

By using two consecutive FS images M_{i-1} and M_i and estimating motion only between pairs of corresponding channels, as shown in Fig. 2, the violation of the intensity inconsistency problem can

be avoided. Despite the larger displacement between the compared images, state-of-the-art motion estimation techniques will most likely produce accurate displacement vectors based on this method. Assuming a constant and linear motion between corresponding channels $C_{i-1,w}$ and $C_{i,w}$, every flow vector $F_{(i-1,w) \rightarrow (i,w)}(x,y)$ is regarded as a linear combination of n identical partial vectors describing a pixel's movement between $C_{i,w}$ and $C_{i,w-1}$,

$$F_{(i,w) \rightarrow (i,w+1)}(x,y) \equiv \frac{1}{n} F_{(i-1,w) \rightarrow (i,w)}(x,y). \quad (1)$$

3.1.1. Bidirectional, all channel optical flow (CCM-B)

The CCM method presented by Steiner et al. [7] calculates a *forward flow* $F_{(i-1,w) \rightarrow (i,w)}$ and a *backward flow* $F_{(i,w) \rightarrow (i-1,w)}$, $w = 1, \dots, n-1$ for each pair of channels $(C_{i-1,w}, C_{i,w})$, $w > 0$. Both forward and backward flows are applied with weights $\frac{(n-w)}{n}$ and $\frac{w}{n}$ in order to interpolate a motion corrected channel $\tilde{C}_{i,w}$, $w = 1, \dots, n-1$, for the reference time $t_{i,0}$:

$$\tilde{C}_{i,w} = \frac{(n-w)}{n} F_{(i-1,w) \rightarrow (i,w)}[C_{i-1,w}] \oplus \frac{w}{n} F_{(i,w) \rightarrow (i-1,w)}[C_{i,w}]. \quad (2)$$

The bidirectional interpolation function \oplus calculates the intensity of every pixel in $\tilde{C}_{i,w}$ by averaging the corresponding pixel values in both $C_{i-1,w}$ and $C_{i,w}$. In conjunction with the detection of occlusions, this function provides high interpolation accuracy. The main disadvantage of this approach is its extremely high computational complexity, as it requires $2 \cdot (n-1)$ OF calculations for each FS image.

In the following, we discuss alternative approaches that reduce computational complexity, but generally also reduce the accuracy of compensation; see Section 7.

3.1.2. Unidirectional, all channel optical flow (CCM-U)

This approach simplifies the interpolation method by using only one OF calculation for each pair of channels $C_{i-1,w}$ and $C_{i,w}$, $w = 1, \dots, n-1$. It uses either the forwards or backwards flow depending on the current channel, to keep the length of the resulting motion vectors and, thus, the expected error as small as possible:

$$\tilde{C}_{i,w} = \begin{cases} \frac{w}{n} \cdot F_{(i,w) \rightarrow (i-1,w)}[C_{i,w}] & \text{if } w \leq \frac{n}{2} \\ \frac{(n-w)}{n} \cdot F_{(i-1,w) \rightarrow (i,w)}[C_{i-1,w}] & \text{if } w > \frac{n}{2} \end{cases} \quad (3)$$

3.1.3. Unidirectional, partial channel optical flow (CCM-1/CCM-2)

The number of OF calculations can be further decreased by interpolating a given flow field to subsequent channels. Assume that the backwards flow $F_{(i,u) \rightarrow (i-1,u)}$ for channel u is known and the motion is constant during the acquisition time of both FS images. Then, the backward flow $F_{(i,v) \rightarrow (i-1,v)}$ for channel $v > u$ can be interpolated using $F_{(i,u) \rightarrow (i-1,u)}$:

$$F_{(i,v) \rightarrow (i-1,v)} = -\frac{v-u}{n} \cdot F_{(i,u) \rightarrow (i-1,u)}[F_{(i,u) \rightarrow (i-1,u)}]. \quad (4)$$

The motion corrected image $\tilde{C}_{i,v}$ is calculated according to Eq. (3) (case $w \leq \frac{n}{2}$). This way, the number of required optical flow calculations for each cube can be reduced down to one (CCM-1). However, the more flow fields are interpolated from a previous one, the higher the approximation error will be if the assumption of constant motion does not hold true. Neglecting this fact, one calculated flow field could even be extrapolated over several image cubes, further reducing the processing time at the cost of an even higher approximation error. In practice and depending on the amount and nature

of expected motion in the scene, it seems to be a better choice to interpolate only a limited number of flow fields from others.

In the case of four (or more) channels per FS image, a more accurate interpolation can be achieved if a second flow field $F_{(i,w) \rightarrow (i-1,w)}$ of a subsequent channel w is used to bidirectionally interpolate $F_{(i,v) \rightarrow (i-1,v)}$, $u < v < w$ (CCM-2):

$$F_{(i,v) \rightarrow (i-1,v)} = -\frac{v-u}{n} \cdot F_{(i,u) \rightarrow (i-1,u)}[F_{(i,u) \rightarrow (i-1,u)}] \oplus \frac{w-v}{n} \cdot F_{(i,w) \rightarrow (i-1,w)}[F_{(i,w) \rightarrow (i-1,w)}]. \quad (5)$$

3.2. Neighboring channel matching (NCM)

The NCM approach commonly applied to ToF images [31–33] estimates motion fields $F_{(i,w) \rightarrow (i,w-1)}$ between adjacent channels $C_{i,w}$, $w > 0$ directly. To compensate motion in $C_{i,w}$, all partial flow fields $F_{(i,w) \rightarrow (i,w-1)}$ are applied to $C_{i,w}$ sequentially:

$$\tilde{C}_{i,w} = F_{(i,1) \rightarrow (i,0)}[\dots [F_{(i,w) \rightarrow (i,w-1)}[C_{i,w}]]]. \quad (6)$$

NCM is a potentially more accurate alternative to CCM, as it keeps the object displacement minimal for each flow calculation and allows to compensate dynamic changes of motion speed and direction during the acquisition of the FS image. Obviously, though, it has to handle the intensity inconsistency problem between different spectral channels.

The expected total interpolation error can further be reduced by estimating motion between adjacent channels of two neighboring FS images forwards or backwards towards the closest reference channel; see Fig. 2. The compensated image $\tilde{C}_{i,w}$ can then be found by sequentially applying the resulting flow vectors either forwards or backwards:

$$\tilde{C}_{i,w} = \begin{cases} F_{(i,1) \rightarrow (i,0)}[[F_{(i,w) \rightarrow (i,w-1)}[C_{i,w}]]] & \text{if } w \leq \frac{n}{2} \\ F_{(i-1,n-1) \rightarrow (i,0)}[[F_{(i-1,w) \rightarrow (i-1,w+1)}[C_{i-1,w}]]] & \text{if } w > \frac{n}{2} \end{cases} \quad (7)$$

4. Image transformation and correlation

In general, the realization of NCM methods requires handling the intensity differences between neighboring channels. Here, three different types of operations can be applied:

1. transformation of the image (channel) into another domain (e.g. gradients),
2. finding dense correlations between neighboring channels (using, e.g., cross-correlation), or
3. applying intensity correction (using, e.g., equalization)

Even though intensity transformation and correlation approaches can be applied sequentially, this is rarely done in literature. Therefore, we decided to apply only one of the methods and describe the related methods in this section. Intensity correction methods are summarized in Section 5.

The following image transformation approaches are evaluated in this respect, in Section 7.

Census transform, proposed by Zabih and Woodfill [35], describes the local spatial structure around a specific pixel of an image by calculating a binary vector $p_{x,y}$ for each pixel: if a neighboring pixel has a lower intensity than $p_{x,y}$, a 1 will be added to the

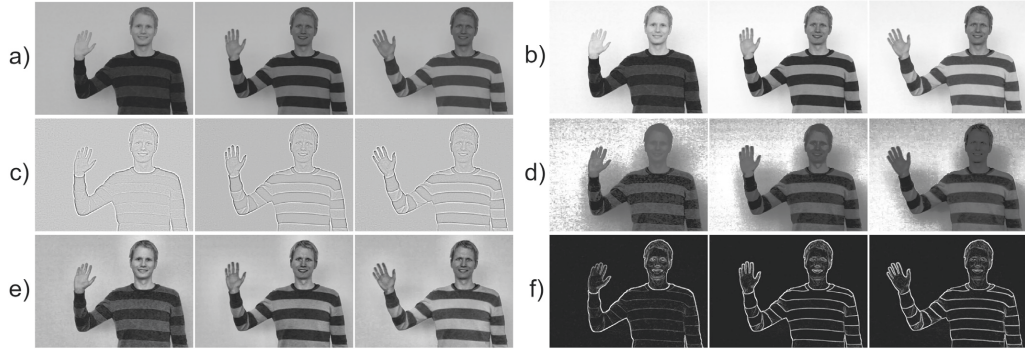


Fig. 3. Examples of methods applied on an FS image with three channels: a) original image; b) global normalization; c) local normalization; d) histogram equalization; e) CLAHE; f) gradients.

vector, otherwise a 0. After the transformation, correspondence is calculated by finding the minimum Hamming distance.

Image gradients describe the intensity variations in a pixel's local neighborhood and can be computed, e.g., using a Sobel filter [36]; see Fig. 3 f).

We investigated the following correlation based approaches that can be applied to two images in order to solve the intensity inconsistency problem:

Mutual information is based on the entropy of an image pair and yields a high value if the information gain of a new image in addition to an existing image is low, i.e., if two images of the same scene are geometrically aligned. Mutual information is known to be robust against non-linear intensity relationships and has been proposed for both multispectral and multimodal image registration applications [37, 38]. It can be used as a cost function for block matching, but it cannot be linearized for the use in OF algorithms.

Cross-spectral feature detection is frequently used in multispectral or multimodal image registration [37], where image transformation or warping parameters are estimated based on detected features. While these methods cannot be used to estimate dense motion fields between two images directly, they might be used for the registration of blocks in block matching algorithms.

Normalized cross-correlation (NCC): Cross-correlation is commonly used as cost function in order to find the position of specific features in an image [36]. NCC additionally normalizes the image which improves the robustness against illumination changes. NCC can be used as an inverse cost function for BM, as well as a linearized data term in OF [39, 20].

4.1. Preliminary method selection

As we face the fundamental problem of combinatorial complexity when evaluating a large amount of approaches making up the final motion estimation method, we executed preliminary tests in order to exclude approaches for which we observe significant drawbacks in our context of motion estimation for FS imagery. Census transform, image gradients and cross-correlation deliver valuable results, so we use them in our exhaustive evaluation in Section 7. Applying mutual information was found to be computationally extremely expensive². Thus, we excluded mutual information as its application

² We tested the fast approximative implementation from Shams and Barnes [40] in combination with block matching. Here, the average execution time for a single image pair with a resolution of 640 × 480 pixels and a (small) search window of 11 × 11 pixels requires ≈300 s on a typical PC.

to a larger set of test sequences and motion estimation methods would be impracticable. Cross-spectral feature detection delivered significantly inferior results when testing with some of our multi-spectral data sets from Section 7. Thus, we also excluded this method from our full evaluation.

5. Intensity correction methods

There are several ways to address the intensity inconsistency problem in case of NCM motion estimation using intensity correction approaches. In the following, we assume a grayscale image I that is intensity corrected, resulting in \tilde{I} . Fig. 3 illustrates their effect on the channels of an FS image.

The following approaches to reduce the intensity inconsistency between the spectral channels are evaluated in Section 7.

Global linear normalization is a simple linear mapping of the used intensity range $[I_{\min}, I_{\max}]$ to a new range $[0, \tilde{I}_{\max}]$ [41]; see Fig. 3 b).

Local linear normalization compensates for non-uniform illumination within an image [42]; see Fig. 3 c). Using the windowed mean $m_l(x, y)$ and variance $\sigma_l(x, y)$ for each pixel (x, y) , the normalized intensity $\tilde{I}(x, y)$ computes as:

$$\tilde{I}(x, y) = \frac{I(x, y) - m_l(x, y)}{\sigma_l(x, y)}. \quad (8)$$

Histogram equalization uniformly distributes the intensity values over the available intensity range [41]; see Fig. 3 d). It normalizes the histogram $H(i)$ of an input image and calculates the cumulative distribution $H'(i)$, which is used to remap the intensity values:

$$\tilde{I}(x, y) = H'(I(x, y)) \quad \text{with} \quad H'(i) = \sum_{0 \leq j \leq i} H(j). \quad (9)$$

Contrast limited adaptive histogram equalization (CLAHE) performs the histogram equalization in a local per-pixel window. As this operation tends to amplify noise in homogeneous areas, the CLAHE algorithm introduces a clipping limit for histogram redistribution [43]; see Fig. 3 e).

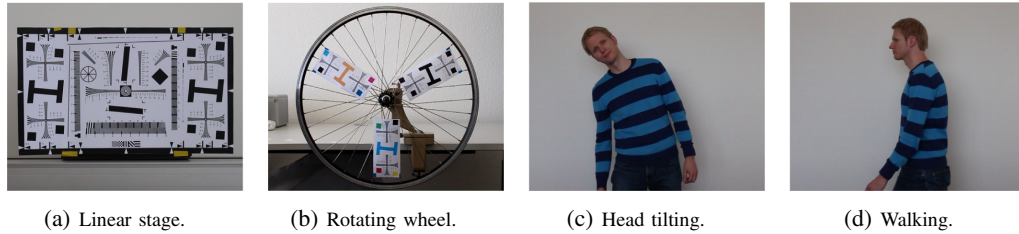


Fig. 4. Examples of the test scenarios included in the FS multimodal data set.

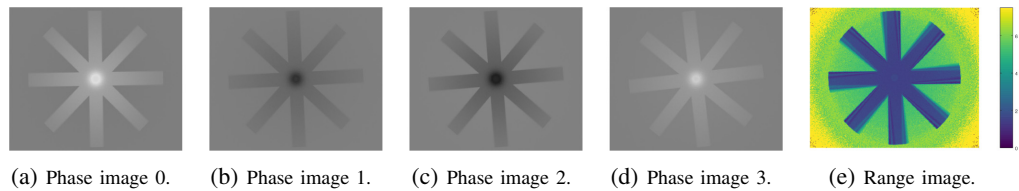


Fig. 5. Sequence of four simulated ToF raw phase images (channels) and the resulting depth frame.

6. Evaluation setup

6.1. Datasets

For evaluation of the motion estimation methods for FS imagery, we have prepared five different types of data sets. In all cases, the number of channels is $n = 3$ or $n = 4$.

Middlebury: All **Middlebury** evaluation samples with at least 8 frames from the Middlebury benchmark³ by Baker et al. [11].

Please note that sequences with less than 8 frames, as well as other, frequently used data sets such as KITTI that do not provide RGB sequences of sufficient length cannot be incorporated in our evaluation.

From each of the sample sequences, we derive an **RGB** FS image sequence with ground truth images by dropping two of the three color channels yielding a 3-channel FS image sequence. Furthermore, we generate **RGB-R** FS image sequences that contain a fourth channel generated by converting the RGB image into a grayscale image with reduced brightness. This channel resembles the so-called *dark reference frame* commonly acquired in *active multispectral camera systems* in order to subtract background illumination.

MPI Sintel: The **MPI Sintel** data set⁴ by Butler et al. [14] (marked as “final”) is used in the **RGB** and **RGB-R** data sets.

Own sRGB: This data set is acquired using an active multispectral SWIR video sequence for three different scenarios (see Fig. 4): *Linear stage* (laterally moving test pattern), *rotating wheel*, and *human movement* showing a person’s upper-body performing several movement patterns. We generated FS data **RGB**, **RGB-R** analogous to **Middlebury** and **MPI Sintel**.

Short Wave Infrared (SWIR): The human movement data in **Own sRGB** provides the SWIR test data. Ground Truth is available using the additional RGB video stream (see below for details).

³ See <http://vision.middlebury.edu/flow/>.

⁴ See <http://sintel.is.tue.mpg.de/>.

Time-of-Flight (ToF): ToF ground truth is extremely hard to access, as the channels, commonly called phase images in ToF imaging, are directly processed in the camera and there is no option to either trigger the exposure of phase images nor to access the explicit timing of the exposition. Therefore, we use simulated ToF imagery based on Lambers et al. and Bulczak et al. [10, 44], for which ground truth motion fields can be explicitly extracted. We used two sample scenes, one with a lateral moving object and one with a rotating star-like shape. Both scenes have been acquired with two different velocities.

The simulator generates four phase images (channels) per depth frame and includes noise and motion artifacts (see Fig. 5). We deactivate the simulation of background intensity in order to be able to apply the same quality measures as for the multispectral data sets that do not include this intensity bias.

Note that all data sets except **MPI Sintel** contain indoor scenes as motion estimation in the context of both multispectral and ToF sensors typically targets such scenes.

6.2. Details for RGB and MS data set acquisition

The multispectral data sets have been captured using an active FS-based SWIR camera system with three wavebands and a dark reference channel (-R), in combination with a high quality RGB camera with the same frame rate. The cameras were arranged in a staring imager configuration in a well-lit environment, which is common for FS NIR imaging systems. Although the subject is the same in all of the human movement sequences, different movement patterns have been captured to provide diversity.

Any negative effect caused by demosaicing of the RGB camera’s Bayer pattern is accounted for by recording in high definition with 1920×1080 pixels and downsampling the images to the resolution of the SWIR camera’s images, i.e. 636×508 pixels.

The ground truth for our MS data set is created from the additional RGB sequences. As the SWIR camera system uses $n = 4$

channels, the RGB camera simultaneously acquires four virtual channels, i.e. R, G, B, and dark reference (-R). For evaluation, we use a cross-compensation approach that applies the optical flow calculated for the SWIR imagery to the RGB image sequence and compares the result to the corresponding RGB full frame.

To match the field of view of the RGB camera to the SWIR camera, the RGB imagery is shifted and cropped appropriately. However, the baseline between both cameras of ≈ 20 cm induces a slightly different perspective and thus a mismatch in the motion fields. To estimate this mismatch, we recorded a second data set where the SWIR camera was replaced by a second RGB camera. Applying the same cross-compensation procedure to this stereo-like setup, we found a *baseline error* for the comparison $IE_{\text{base}} \approx 2.7$. As IE_{base} is by far lower than the error of the best FS motion compensation method with $IE \approx 6.6$, we find our cross-compensation approach to be valid within this range.

6.3. Quality measures

The objective comparison of a compensated image with the ground truth image is performed using the following quality measures:

1. *Interpolation Error (IE)* [11] is defined as the root mean square of the L2 norm of the vector of spectral channel differences between the interpolated and ground truth images, analog to the Middlebury OF evaluation,

2. *Structural Similarity Index Metric (SSIM)*, which describes the similarity of images based on structural information and is inspired by the human visual perception [45], and
3. *Spectral Error (SE)*, which we define as the root mean square of all pixel's spectral angular distance [41].

7. Results and discussion

With 379 combinations of methods and preprocessing options, the total amount of results is very extensive. Here, we present only a representative selection and summarize the findings. The complete results can be found in the digital supplemental material, which presents all details about influences of individual steps and changes of preprocessing methods or algorithms to the results.

Table 1 states all methods and algorithms applied and explains the abbreviations used in the following evaluation. CCM and NCM are the **General concepts** applicable to motion estimation for FS imagery. For CCM, we need to specify the **Motion estimation scheme** that defines how the full flow for an FS image frame is computed, e.g., using the uni- or bidirectional approach (see Section 3). Remember that CCM-2 only makes sense if we have four (or more) channels per FS image, thus CCM-2 can only be applied to **RGB-R** and **ToF** data sets. Either of the resulting concept-scheme combinations is applied to the original or a transformed version of the channels (**Image transformation**; see Section 4) that has optionally been processed by an **Intensity correction** method (see Section 5). The resulting combination can be implemented using any kind of BM or

Table 1
Abbreviations used in Table 2, Figs. 6 and 7, 8, 9, 10, and 11.

General concept	
CCM	Corresp. channel matching
NCM	Neighboring channel matching
CCM motion estimation scheme	
-B	All channels bidirectional
-U	All channels unidirectional
-2	Partial (2 channels)
-1	Partial (1 channel)
Image transformation	
-I	Intensity (i.e. no transformation)
-TG	Transformation to gradients
-TC	Transformation to census
-C	Correlation
Intensity correction	
N	Global normalization
L	Local normalization
H	Histogram equalization
C	Contrast limited adaptive Histogram equalization
Algorithms	
BM	Block Matching, sum of absolute differences
FBM	Fast Block Matching, BM with restricted set of candidates
LK	Lucas-Kanade OF
Br	Brox OF
TVL1	TV-L1
HL1	Huber-L1
FHL1	Fast Huber-L1, parameters optimized for speed [19]
HQS	Huber-L1 with quadratic fitting, sum of absolute differences [20]
H1C	Huber regularization term, L1 data term, compensation of brightness constancy violations
H2C	Huber regularization term, L2 data term, compensation of brightness constancy violations
TGVC	2nd order Total Generalized Variation w. Census transform [46, 47]
FF	FlowFields
LDOF	Large Displacement Optical Flow
PAC	Pixelwise Artifact Correction
FN	FlowNet2 [29]
LFN	LiteFlowNet [30]

* = optimized for speed by authors of this paper, see Section 7.

** = part of algorithm.

Table 2

Results of the top-20 ranking approaches with respect to the multispectral data set (**Own sRGB**, **Middlebury** and **MPI Sintel**) (top section), the top-20 ranking approaches with respect to the **ToF** data sets (middle section), and selected additional approaches (lower section) on our data sets; values are given as averaged ranks unless otherwise noted.

Concept,	Image	Int.	Algo.	Tot. MS	Tot. ToF	Own sRGB		SWIR	Middlebury		MPI Sintel		Time
				Avg.	Avg.	RGB	RGB-R		RGB	RGB-R	RGB	RGB-R	
Uncomp			-	296.59	161.08	261.97	285.52		251.78	283.2	303.64	318.19	-
NCM	-TG**		HL1	45.11	239.42	25.18	28.55	43.41	106.97	29.75	40.56	41.39	0.23
NCM	-I	N	Br	46.54	203.17	45.39	20.21	94.74	86.63	24.53	36.25	18.00	0.32
NCM	-TG**	C	HL1	46.64	248.67	31.12	25.82	36.81	129.00	38.95	39.75	25.06	0.24
NCM	-I	L	LDOF	47.67	151.33	44.76	23.21	45.19	67.83	23.00	64.81	64.92	9.68
NCM	-TG	L	LDOF	50.58	155.42	58.85	30.70	50.00	77.82	35.58	50.17	50.94	7.83
NCM	-TG	N	LDOF	50.84	136.33	95.73	36.18	24.30	71.17	24.87	61.92	41.72	5.29
NCM	-TG**	N	HL1	55.61	285.58	36.88	21.12	136.19	110.17	41.18	32.00	11.72	0.27
NCM	-I	N	LDOF	57.18	129.67	156.15	25.52	73.63	62.18	27.70	39.75	15.36	7.88
NCM	-(I+TG)**	C	HL1	57.78	264.42	78.21	33.61	34.67	142.03	37.23	46.78	31.92	0.16
NCM	-TG	N	LK	58.15	134.75	55.21	20.82	47.96	105.17	44.18	90.61	43.08	0.07
NCM	-TG	N	LK*	59.82	127.17	67.76	14.27	47.41	91.95	46.87	100.06	50.42	0.04
NCM	-TG+TG**		HL1	65.51	46.83	37.30	29.64	41.48	139.57	59.55	74.58	76.44	0.23
NCM	-I	L	LFN	66.80	175.25	109.67	88.39	25.59	50.82	98.53	32.56	62.03	7.65
NCM	-TG	N	HQS	69.23	114.83	44.67	51.85	65.74	123.03	44.73	109.28	45.28	0.18
NCM	-I	C	Br	69.68	188.92	20.55	73.73	58.81	94.00	142.82	18.39	79.47	0.32
NCM	-I	H	LDOF	70.21	225.50	124.42	40.76	176.56	73.87	27.78	34.83	13.28	7.02
NCM	-TG	H	LDOF	74.73	234.00	95.00	35.97	149.89	97.67	45.35	56.58	42.64	7.98
NCM	-TG	L	Br	74.97	134.17	74.48	37.33	59.07	111.23	83.02	87.25	72.39	0.48
NCM	-TG**	C	HL1*	75.27	224.25	54.61	44.55	83.07	148.95	78.47	65.42	51.83	0.05
NCM	-TG+TG**	N	HL1	75.50	78.25	52.61	45.48	80.37	141.53	90.15	69.94	48.39	0.27
CCM-B	-I		FHL*	118.11	22.25	100.39	139.45	52.00	69.28	109.83	173.03	182.78	0.03
CCM-B	-I		HL1*	118.98	23.25	100.88	140.03	54.19	70.28	111.20	173.06	183.25	0.04
CCM-B	-TG**		HL1	97.82	26.42	88.88	128.85	48.59	51.52	95.45	127.42	144.06	0.43
CCM-B	-TG**		TGVC	102.18	26.50	88.82	123.30	46.96	60.13	96.10	145.92	154.06	0.49
CCM-B	-I		H1C*	119.19	29.50	102.42	139.61	53.22	71.22	110.62	173.94	183.33	0.05
CCM-B	-I		HQS*	106.55	31.33	85.03	122.06	50.59	69.55	112.53	147.69	158.42	0.05
CCM-B	-TG**		HL1*	111.00	33.17	105.33	143.03	48.30	66.50	103.77	151.94	158.14	0.07
NCM	-TG+TG**	C	HL1	87.49	34.92	67.42	46.70	52.30	179.73	113.70	84.14	68.42	0.24
CCM-B	-(I+TG)**		HL1*	105.81	35.08	95.48	132.24	46.89	72.57	108.85	139.56	145.06	0.05
CCM-B	-TG**		TGVC*	139.29	36.00	140.79	168.48	67.63	97.97	126.10	190.50	183.56	0.08
CCM-B	-I		FHL	103.49	36.42	88.45	121.70	54.30	55.17	94.28	144.06	166.44	0.19
CCM-B	-I		HL1	103.49	36.83	88.85	123.52	49.89	55.95	94.15	145.47	166.64	0.23
NCM	-TG+TG**	C	TGVC*	118.22	37.00	77.94	49.48	119.59	155.17	154.13	139.42	131.78	0.07
CCM-B	-(I+TG)**		HL1	87.33	38.42	80.76	112.45	46.56	44.68	87.57	116.28	123.03	0.29
CCM-B	-I		TVL1	122.72	38.58	112.52	143.94	65.70	61.52	107.15	174.67	193.53	0.68
NCM	-TG+TG**	C	HL1*	110.30	39.92	80.82	56.97	71.70	195.67	145.85	122.19	98.89	0.06
NCM	-TG		H1C*	194.64	41.75	114.00	164.76	119.48	226.85	298.55	192.53	246.33	0.04
CCM-B	-I		H1C	104.54	46.50	88.82	122.55	50.22	58.07	97.87	148.28	166.00	0.30
NCM	-TG+TG**		HL1	65.51	46.83	37.30	29.64	41.48	139.57	59.55	74.58	76.44	0.23
NCM	-TG+TG**		HL1*	92.62	47.17	59.21	44.85	59.26	166.10	94.98	115.06	108.86	0.06
NCM	-I		BM	314.19	281.00	283.85	351.30	322.52	279.05	342.77	287.03	332.83	11.50
NCM	-TG		BM	244.12	270.25	218.12	233.18	259.44	217.22	272.50	246.06	262.31	11.62
CCM-B	-I		Br	84.74	49.00	70.15	97.24	56.52	49.38	77.60	111.83	130.44	0.66
CCM-U	-I		Br	131.95	118.50	114.97	152.61	85.96	77.17	127.20	170.50	195.25	0.32
NCM	-TG		Br	138.52	78.92	60.06	167.42	127.93	112.13	199.38	102.00	200.72	0.33
NCM	-I		TVL1	316.43	285.17	265.79	354.12	317.85	285.47	366.20	267.78	357.78	1.12
NCM	-I	H	FBM	183.24	277.25	189.91	156.36	255.37	209.50	124.17	193.50	153.86	0.20
NCM	-I		HL1	317.31	256.25	271.61	363.64	311.44	293.20	360.43	272.28	348.56	0.12
NCM	-TG**		TGVC	129.21	303.92	142.33	109.45	189.85	163.15	121.43	80.97	97.28	0.26
NCM	-I	N	LK	304.08	215.17	284.70	354.48	292.48	231.80	361.27	250.17	353.69	0.05
NCM	-C**		HL1	154.97	312.25	196.64	123.27	271.15	191.98	122.22	106.36	73.17	0.16
NCM	-I	H	HL1	210.25	346.00	237.79	137.36	316.67	275.88	192.97	210.06	101.00	0.14
CCM-U	-I		LDOF	114.82	112.83	135.30	161.21	48.22	55.87	104.55	131.33	167.25	8.50
NCM	-I		LDOF	174.20	135.33	118.91	282.45	142.85	61.90	290.15	51.47	271.67	5.61
NCM	-I		FF	346.42	346.42	307.55	326.67	273.44	184.35	261.63	266.42	307.03	53.54
NCM	-I	L	FF	291.13	332.00	289.82	280.85	314.78	182.32	168.47	264.39	251.25	53.01
CCM-B	-I		FN	106.84	121.00	120.42	123.85	39.70	50.12	79.85	110.44	124.33	30.46
NCM	-I	L	FN	154.72	191.00	178.09	149.94	56.37	111.50	139.95	87.69	105.56	15.40
NCM	-I	H	LFN	106.75	333.92	162.88	131.09	164.70	63.97	109.43	38.72	76.44	7.53
CCM-1	-I		LFN	200.87	89.67	222.55	262.21	100.30	145.48	209.13	213.42	253.00	2.52

dense OF algorithm. In this work, GPU-accelerated implementations of Brox-, TV-L1, Lucas-Kanade (LK), LDOF and Huber-L1-based optical flow, as well as full search and fast approximate BM algorithms from standard libraries (OpenCV 2.4.11 and FlowLib 3.0) are used. They are complemented by a (multithreading) CPU implementation of FlowFields, which is not currently available as a GPU-accelerated version. All Brox- and FlowLib-based algorithms have been applied twice, once with recommended (quality-oriented) parameters and

once with parameters optimized for speed, which is denoted with *. Optimal parameters have been found experimentally⁵.

As some of the algorithms already include an image transformation, e.g. to gradients, we explicitly mark this with **. In

⁵ BM: search field parameter $p = 20$, block size $bs = 25$; Brox: 3 instead of 10 inner and solver iterations each; FlowLib: 3 instead of 10 iterations and warps each.

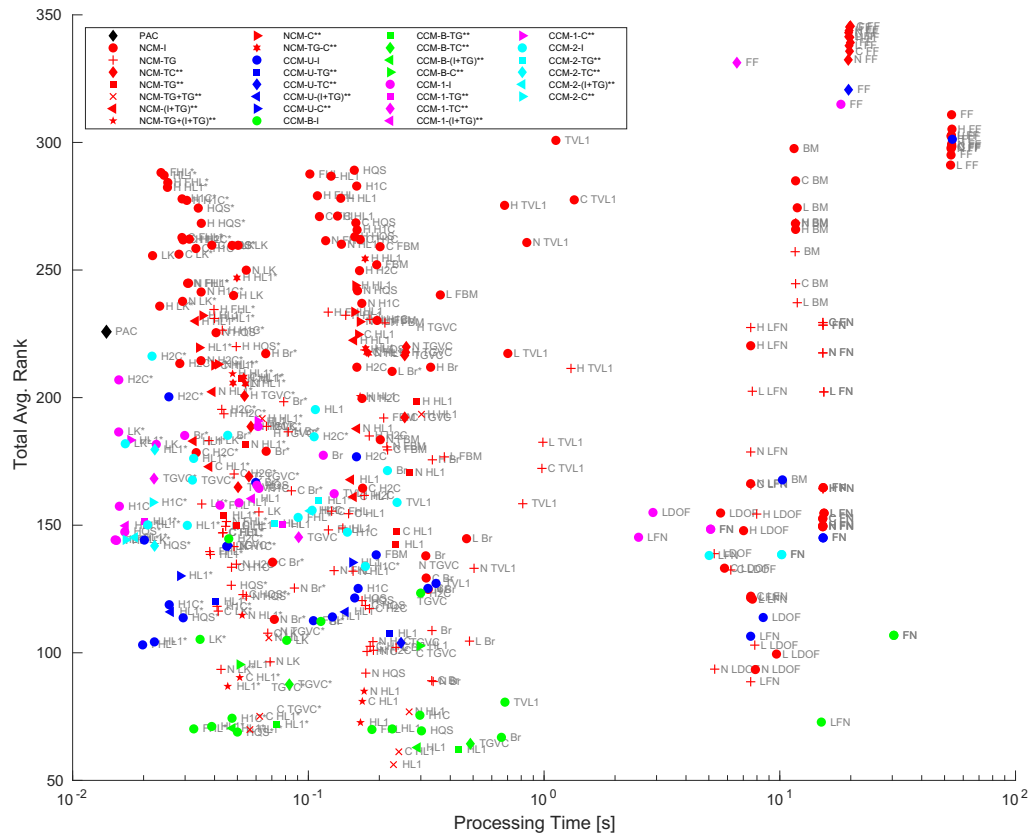


Fig. 6. Scatter plot showing accuracy and processing time of the described approaches. An SVG version of this plot is available in the Supplementary material.

our evaluation, we also feed gradient images to these algorithms, leading to an overall image transformation of type $TG+TG^{**}$. Some algorithms internally use intensity and gradient images for estimating motion, denoted as $(I+TG)^{**}$. If fed with gradient images, these algorithms read $TG+(I+TG)^{**}$.

Computational efficiency is measured on a standard desktop computer with a recent Intel CPU and nVidia graphics card using the FS multispectral **RGB-R** data set. Note, however, that there is some variation in system setup between methods, so the timing information in Table 2 and Fig. 6 should be interpreted as a rough estimate.

Analog to the Middlebury evaluation, all methods were ranked for each test sequence based on all described quality measures. Table 2 shows the average ranks of the top-20 combinations of algorithms and approaches with respect to multispectral data set (including **Own sRGB**, **Middlebury** and **MPI Sintel**) in the upper part. The middle part of Table 2 shows the ranking with respect to the **ToF** data set. For comparison, results of the original algorithms without optimization for FS data sequences and different CCM optimizations have been added in the lower part. In addition, Fig. 6 allows to easily compare the motion compensation performance to the computational efficiency of the different methods. A higher resolution plot can also be found in the supplemental material. All abbreviations are explained in Table 1.

To illustrate the performance of different approaches, example images from each data set and a selection of motion compensation results are shown in Figs. 7, 8, 9, 10, and 11.

7.1. Comparing CCM and NCM

The multispectral data sets (including **Middlebury** and **MPI Sintel**) are handled best with NCM methods. This is mainly due to the fact, that multispectral data sets exhibit less intensity inconsistencies than ToF data. Furthermore, non-linear motion can be better captured using neighboring channel matching (NCM) due to shorter interpolation intervals. CCM-methods, on the other hand, produce primarily superior results on **ToF** data sets. However, several NCM methods work well on **ToF** data sets. Namely $NCM-TG+TG^{**}$ with the HL1 algorithm, which is among the top-20 for both, multispectral and ToF.

7.2. Varying the number of OF in CCM

Table 2 includes CCM results using all channels bi- (CCM-B) and unidirectional (CCM-U), first and last channel (CCM-2), as well as first channel only (CCM-1) based on the Brox algorithm that performs best for CCM. For all algorithms, a reduction of the number of

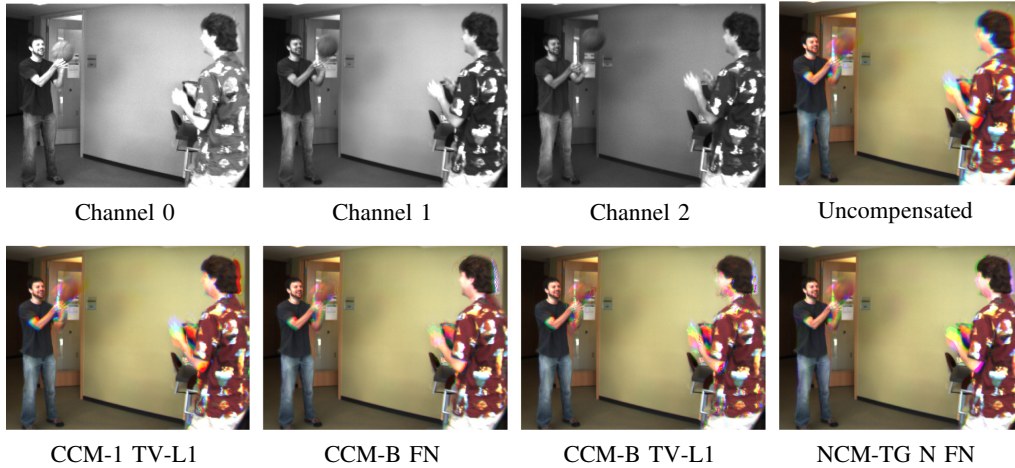


Fig. 7. Examples for an image before and after motion compensation (data set Middlebury).

OF calculations decreases the processing time almost proportionally, while simultaneously increasing the error in a very predictable way; see Fig. 6.

7.3. Handling of inconsistent intensities with NCM

Without image transformation and intensity correction, only the Brox algorithm (NCM-I-Br) is capable of handling the inconsistent intensities with NCM-I methods to some degree. Applying image transformation only (NCM-I- $\langle \text{none} \rangle$), transformation to gradient, potentially applied twice, i.e. during preprocessing and, again, within the algorithm itself, clearly yields the best results. The census transformation and correlation-based methods (NCM-C) cannot compete. Applying intensity correction only (NCM-I- \ast), Brox and LDOF perform well on multispectral data if global or local normalization is applied.

7.4. Influence of the OF algorithm

Ignoring image transformation and intensity correlation-based, there is no clear tendency in terms of OF algorithms, neither for

the multispectral nor for the ToF data sets. While LDOF, as one of the more modern approaches is quite successful on multispectral data sets, more classical approaches like Brox and Huber-based algorithms yield comparable results; on ToF data sets they even dominate. Surprisingly, the most modern algorithm in the evaluation, FlowFields, performed worst. A possible explanation for this finding could be that its SIFTflow matching approach is optimized for color rather than grayscale images.

When taking processing time into account, the normal and speed-optimized Huber-L1 (HL1, FHL1), as well as Lucas-Kanade optical flow (LK) deliver outstanding results.

7.5. Pixelwise artifact correction (PAC)

This approach from Schmidt and Jahne [34] is the only one specifically developed to correct ToF raw data. It performs comparably bad regarding quality, but the approach is computationally very effective and fast, although it does not rely on GPU acceleration.

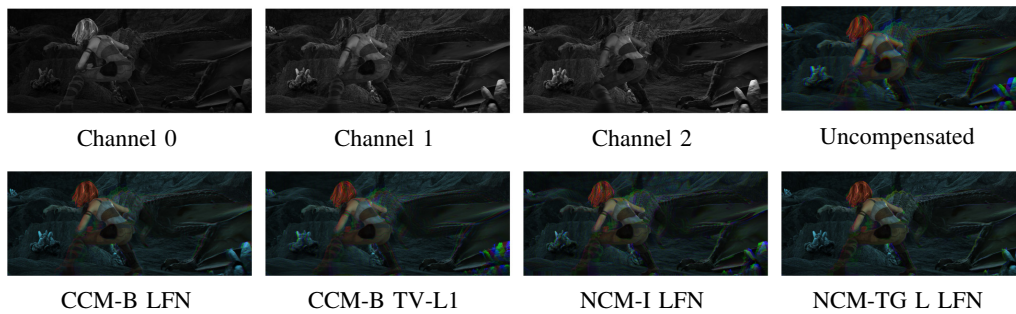


Fig. 8. Examples for an image before and after motion compensation (data set MPI Sintel).

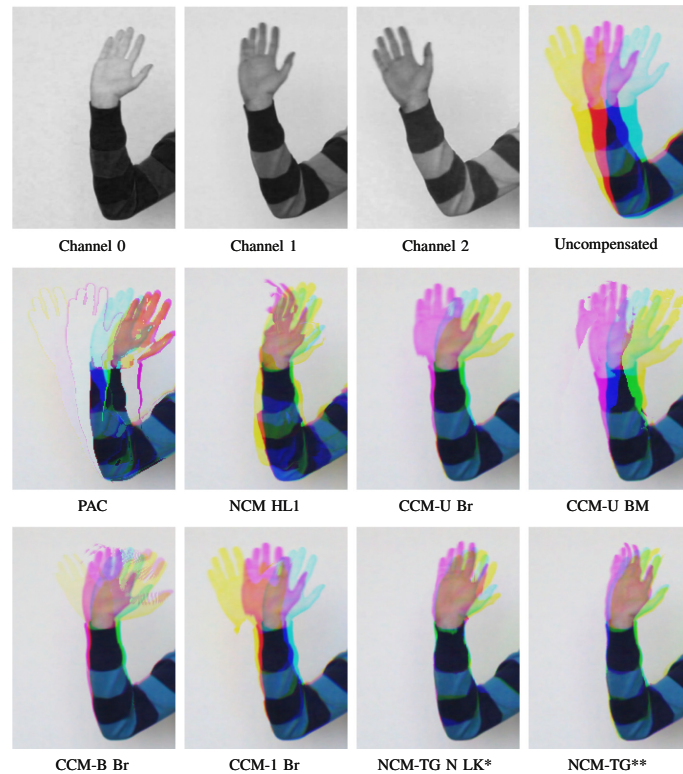


Fig. 9. Examples for an image before and after motion compensation (data set Own sRGB).

7.6. Deep neural networks

The deep neural network based methods FlowNet2 (FN) and Lite-FlowNet (LFN) both perform well. Note that we use public available pre-trained implementations of both. These have been trained on RGB data, whereas here, we apply them to individual channels of our FS imagery which are interpreted as grayscale images. Training either method specifically for a given FS type (multispectral or ToF) will likely result in improved quality.

8. Conclusions

This paper presents and evaluates approaches to apply existing motion estimation methods to field-sequential (FS) imagery, originating from multispectral dynamic scene captures or Time-of-Flight cameras. The major challenge here is the assumption of consistent intensities for corresponding pixels made by most motion estimation approaches, which is in general not fulfilled for adjacent channels of FS imagery.

While corresponding channel matching (CCM) methods estimate motion fields between corresponding channels of successive FS images to avoid intensity inconsistencies, neighboring channel matching (NCM) estimates motion fields between neighboring channels within a single FS image, which requires a successful handling of inconsistent intensities between the channels but (potentially)

benefits from interpolation for shorter time intervals and displacement vectors.

We combine existing motion estimation schemes with known image transformation and/or intensity correction methods, leading to an overall set of 379 combinations of FS motion compensation approaches, implemented using state of the art algorithms.

We present the new FS database containing data sets with ground truth acquired using RGB, multispectral SWIR, and ToF camera simulators, which will be available to the scientific public in order to promote further research in this field. Our evaluation also involves data from the **Middlebury** and the **MPI Sintel** data sets.

Due to the variety in the FS database that includes also strongly intensity inconsistent ToF phase images as well as moderate inconsistent multispectral imagery, there is not “the best” method superior to others. There is, however, a clear tendency, that NCM methods are more successful for moderate intensity inconsistency. For strong intensity inconsistency, CCM methods perform best, while NCM in combination with gradient transformation (potentially applied twice) still give good results.

Datasets

The full data used in this paper is available at <https://www.cg.informatik.uni-siegen.de/data/fsmotion2019/>.

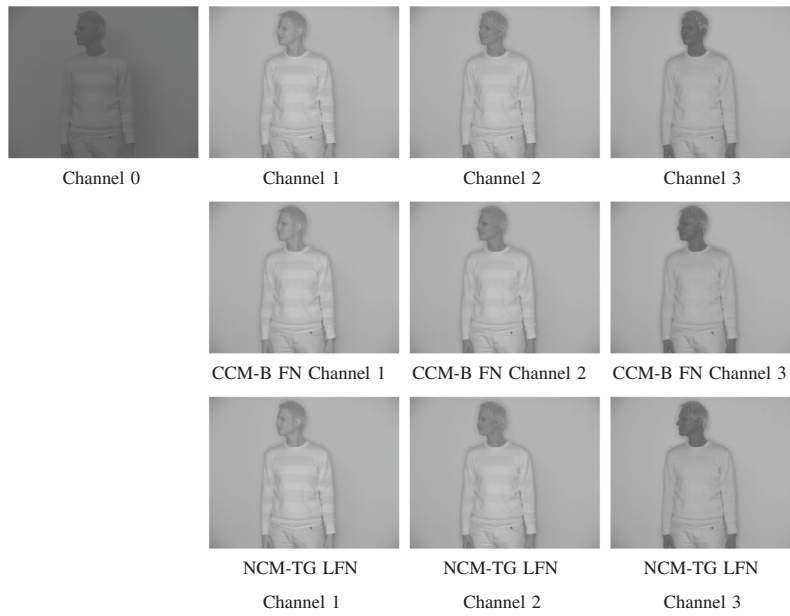


Fig. 10. Examples for an image before and after motion compensation (data set SWIR).

Declaration of Competing Interest

The authors declare that they do not have a conflict of interest.

1564 “Imaging New Modalities” and the DFG project grant KO-2960/12-1.

Acknowledgments

This research was partially funded by German Research Foundation (DFG) within the Research Training Group GRK

Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.imavis.2019.07.001>.

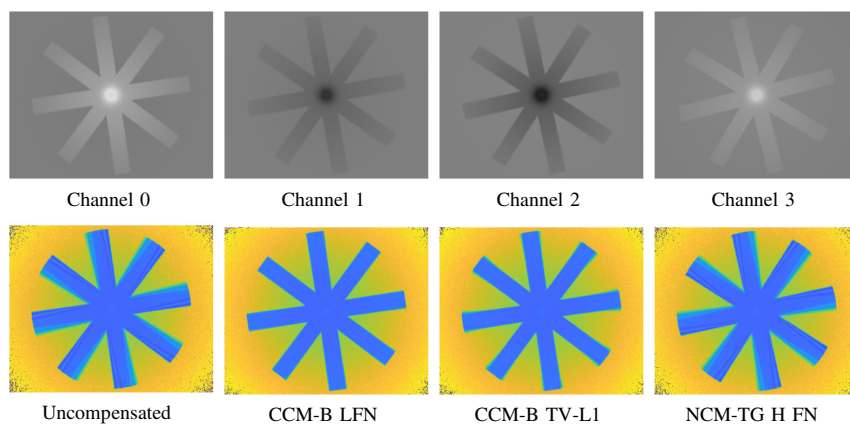


Fig. 11. Examples for an image before and after motion compensation (data set ToF).

References

- [1] S. Daly, X. Feng, Method and system for field sequential color image capture using color filter array 2004, US Patent 6,690,422.
- [2] A. Gowen, C. O'Donnell, P. Cullen, G. Downey, J. Frias, Hyperspectral imaging an emerging process analytical tool for food quality and safety control, *Trends Food Sci. Technol.* 18 (12) (2007) 590–598. ISSN 0924-2244. <https://doi.org/10.1016/j.tifs.2007.06.001>.
- [3] J. Brauers, T. Aach, S. Helling, Multispectral image acquisition with flash light sources, *J. Imaging Sci. Technol.* 53 (3), (2009) 31103–1–31103-10. <https://doi.org/10.2352/J.ImagingSci.Technol.2009.53.3.031103>.
- [4] S. Helling, E. Seidel, W. Biehlig, Algorithms for spectral color stimulus reconstruction with a seven-channel multispectral camera, *Conf. on Colour in Graphics, Imaging, and Vision 2004*(1), 2004, pp. 254–258.
- [5] T. Bourlai, N. Narang, B. Cukic, L. Hornak, On designing a SWIR multi-wavelength facial-based acquisition system, *Infrared Technology and Applications XXXVIII*, 8353, 2012, pp. 83530R. <https://doi.org/10.1117/12.919392>.
- [6] N. Gal, Imaging spectroscopy using tunable filters: a review, *Proc. SPIE Wavelet Applications VII*, 4056, 2000, pp. 50–64. <https://doi.org/10.1117/12.381686>.
- [7] H. Steiner, S. Sporrer, A. Kolb, N. Jung, Design of an active multispectral SWIR camera system for skin detection and face verification, *J. Sens.* (1), (2016) article ID 9682453. <https://doi.org/10.1155/2016/9682453>.
- [8] R. Lange, P. Seitz, Solid-state time-of-flight range camera, *IEEE J. Quantum Electron.* 37 (3) (2001) 390–397.
- [9] A. Kolb, E. Barth, R. Koch, R. Larsen, Time-of-flight cameras in computer graphics, *Computer Graphics Forum (Eurographics STAR)*, 29, 2010, pp. 141–159.
- [10] M. Lambers, S. Hoberg, A. Kolb, Simulation of time-of-flight sensors for evaluation of chip layout variants, *IEEE Sensors* 15 (7) (2015) 4019–4026.
- [11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, R. Szeliski, A database and evaluation methodology for optical flow, *Int. J. Comput. Vis.* 92 (1) (2011) 1–31. <https://doi.org/10.1007/s11263-010-0390-2>.
- [12] D. Fortun, P. Bouthemy, C. Kervrann, Optical flow modeling and computation: a survey, *Comput. Vis. Image Underst.* 134 (2015) 1–21. ISSN 1077-3142. <https://doi.org/10.1016/j.cviu.2015.02.008>.
- [13] M. Jakubowski, G. Pastuszak, Block-based motion estimation algorithms – a survey, *Opto-Electron. Rev.* 21 (1) (2012) 86–102. ISSN 1896-3757. <https://doi.org/10.2478/s11772-013-0071-0>.
- [14] D.J. Butler, J. Wulff, G.B. Stanley, M.J. Black, et al. A naturalistic open source movie for optical flow evaluation, in: A. Fitzgibbon (Ed.), *European Conf. on Computer Vision (ECCV)*, Springer-Verlag, 2012, pp. 611–625. Part IV, LNCS 7577.
- [15] B.K. Horn, B.G. Schunck, Determining Optical Flow, *Proc. SPIE*, 0281, 1980, pp. 319–331. <https://doi.org/10.1117/12.965761>.
- [16] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, *Proc. Int. Conf. Artificial Intelligence (IJCAI)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1981, pp. 674–679.
- [17] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: T. Pajdla, J. Matas (Eds.), *Computer Vision (ECCV)*, Lecture Notes in Computer Science 3024, Springer Berlin Heidelberg, 2004, pp. 25–36. ISBN 978-3-540-21981-1. https://doi.org/10.1007/978-3-540-24673-2_3.
- [18] C. Zach, T. Pock, H. Bischof, A duality based approach for realtime TV-L1 optical flow, *Proc. DAGM Conf. Pattern Recognition*, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 214–223. ISBN 978-3-540-74933-2.
- [19] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, H. Bischof, Anisotropic Huber-L1 optical flow, *Proc. British Conf. Machine Vision (BMVC)*, 2009, London, UK.
- [20] M. Werlberger, *Convex Approaches for High Performance Video Processing*, Ph.D. thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria, 2012.
- [21] T. Brox, J. Malik, Large displacement optical flow: descriptor matching in variational motion estimation, *IEEE Trans. Pattern Anal. Mach. Intell.* 33 (3) (2011) 500–513. ISSN 0162-8828. <https://doi.org/10.1109/TPAMI.2010.143>.
- [22] J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid, EpicFlow: edge-preserving interpolation of correspondences for optical flow, *Computer Vision and Pattern Recognition*, 2015.
- [23] C. Baillet, B. Taetz, D. Stricker, Flow fields: dense correspondence fields for highly accurate large displacement optical flow estimation, *Proc. IEEE Int. Conf. on Computer Vision*, 2015.
- [24] B.-D. Choi, J.-W. Han, C.-S. Kim, S.-J. Ko, Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation, *IEEE Trans. Circuits Syst. Video Technol.* 17 (4) (2007) 407–416. ISSN 1051-8215. <https://doi.org/10.1109/TCSVT.2007.893835>.
- [25] E. Cuevas, D. Zaldivar, M.A. Perez-Cisneros, D. Oliva, Block matching algorithm based on differential evolution for motion estimation, *Eng. Appl. Artif. Intell.* 26 (1) (2013) 488–498.
- [26] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, T. Brox, FlowNet: Learning Optical Flow with Convolutional Networks, *IEEE Int. Conf. Computer Vision (ICCV)*, 2015. <https://doi.org/10.1109/ICCV.2015.316>.
- [27] M. Menze, A. Geiger, Object Scene Flow for Autonomous Vehicles, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015. <https://doi.org/10.1109/CVPR.2015.7298925>.
- [28] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, H. Zha, Unsupervised deep learning for optical flow estimation, *Proc. AAAI Conference on Artificial Intelligence*, 2017. <https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/viewPaper/14388>.
- [29] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017. <https://doi.org/10.1109/CVPR.2017.179>.
- [30] T. Hui, X. Tang, C.C. Loy, LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018. <https://doi.org/10.1109/CVPR.2018.00936>.
- [31] M. Lindner, A. Kolb, Compensation of Motion Artifacts for Time-of-Flight Cameras, *Proc. Dynamic 3D Imaging, LNCS*, 5742, Springer, 2009, pp. 16–27.
- [32] D. Lefloch, T. Hoegg, A. Kolb, Real-Time Motion Artifacts Compensation of ToF Sensors Data on GPU, *Proc. SPIE Defense, Security - Three-Dimensional Imaging, Visualization, and Display*, 2013, pp. 87380U–87380U-7.
- [33] T. Högg, D. Lefloch, A. Kolb, Real-Time Motion Artifact Compensation for PMD-ToF Images, *Proc. Workshop Imaging New Modalities, German Conference of Pattern Recognition (GCPR)*, LNCS, 8200, Springer, 2013, pp. 273–288.
- [34] M. Schmidt, B. Jähne, Efficient and robust reduction of motion artifacts for 3D Time-of-Flight cameras, *Proc. Int. Conf. 3D Imaging (IC3D)*, 2011, pp. 1–8. <https://doi.org/10.1109/IC3D.2011.6584391>.
- [35] R. Zabih, J. Woodfill, Non-parametric local transforms for computing visual correspondence, *Computer Vision (ECCV)*, Lecture Notes in Computer Science, 801, Springer Berlin Heidelberg, 1994, pp. 151–158. ISBN 978-3-540-57957-1. <https://doi.org/10.1007/BFb0028345>.
- [36] B. Jähne, *Digital Image Processing*, Springer Berlin Heidelberg, 2005.
- [37] B. Zitova, J. Flusser, Image registration methods: a survey, *Image Vis. Comput.* 21 (11) (2003) 977–1000. ISSN 0262-8856. [https://doi.org/10.1016/S0262-8856\(03\)00137-9](https://doi.org/10.1016/S0262-8856(03)00137-9).
- [38] J. Kern, M. Pattichis, Robust multispectral image registration using mutual-information models, *IEEE Trans. Geosci. Remote Sens.* 45 (5) (2007) 1494–1505. ISSN 0196-2892. <https://doi.org/10.1109/TGRS.2007.892599>.
- [39] F. Steinbruecker, T. Pock, D. Cremers, Advanced Data Terms for Variational Optical Flow Estimation, *Proc. Vision, Modeling and Visualization Workshop*, 2009, pp. 155–164.
- [40] R. Shams, N. Barnes, Speeding up Mutual Information Computation Using NVIDIA CUDA Hardware, *Proc. Conf. Digital Image Computing Techniques and Applications*, 2007, pp. 555–560. <https://doi.org/10.1109/DICTA.2007.4426846>.
- [41] M. Petrou, C. Petrou, *Image Processing: The Fundamentals*, 2nd edn. ed., John Wiley & Sons, 2010.
- [42] D. Sage, *Local Normalization*, Biomedical Image Group, EPFL, 2011, <http://bigwww.epfl.ch/sage/soft/localnormalization/>.
- [43] S.M. Pizer, E.P. Amburn, J.D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J.B. Zimmerman, K. Zuiderveld, Adaptive histogram equalization and its variations, *Comput. Vision Graphics Image Process.* 39 (3) (1987) 355–368. ISSN 0734-189X. [https://doi.org/10.1016/S0734-189X\(87\)80186-X](https://doi.org/10.1016/S0734-189X(87)80186-X).
- [44] D. Bulczak, M. Lambers, A. Kolb, Quantified, interactive simulation of AMCW ToF camera including multipath effects, *Sensors* 18 (1) (2018) 13.
- [45] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, *IEEE Trans. Image Process.* 13 (4) (2004) 600–612. ISSN 1057-7149. <https://doi.org/10.1109/TIP.2003.819861>.
- [46] K. Bredies, K. Kunisch, T. Pock, Total generalized variation, *SIAM J. Imag. Sci.* 3 (3) (2010) 492–526. <https://doi.org/10.1137/090769521>.
- [47] R. Ranftl, K. Bredies, T. Pock, Non-local Total Generalized Variation for Optical Flow Estimation, *Computer Vision (ECCV)*, 2014, pp. 439–454. ISBN 978-3-319-10590-1.

Realistic Lens Distortion Rendering

Martin Lambers
Computer Graphics Group
University of Siegen
Hoelderlinstrasse 3
57076 Siegen
martin.lambers@
uni-siegen.de

Hendrik Sommerhoff
Computer Graphics Group
University of Siegen
Hoelderlinstrasse 3
57076 Siegen
hendrik.sommerhoff@
student.uni-siegen.de

Andreas Kolb
Computer Graphics Group
University of Siegen
Hoelderlinstrasse 3
57076 Siegen
andreas.kolb@
uni-siegen.de

ABSTRACT

Rendering images with lens distortion that matches real cameras requires a camera model that allows calibration of relevant parameters based on real imagery. This requirement is not fulfilled for camera models typically used in the field of Computer Graphics.

In this paper, we present two approaches to integrate realistic lens distortions effects into any graphics pipeline. Both approaches are based on the most widely used camera model in Computer Vision, and thus can reproduce the behavior of real calibrated cameras.

The advantages and drawbacks of the two approaches are compared, and both are verified by recovering rendering parameters through a calibration performed on rendered images.

Keywords

Lens distortion, Camera calibration, Camera model, OpenCV

1 INTRODUCTION

In Computer Graphics, the prevalent camera model is the pinhole camera model, which is free of distortions and other detrimental effects. Real world cameras, on the other hand, use lens systems that lead to a variety of effects not covered by the pinhole model, including depth of field, chromatic aberration, and distortions. This paper focusses on the latter.

In Computer Vision, distortions must be taken into account during 3D scene analysis. A variety of camera models have been suggested to model the relevant effects; Sturm et al. [1] give an overview. The dominant model in practical use is a polynomial model based on the work of Heikkilä [2, 3] and Zhang [4] and is implemented in the most widely used Computer Vision software packages: OpenCV [5] and Matlab/Simulink [6]. In the following, we refer to this camera model as the standard model. Typical Computer Vision applications estimate the distortion parameters of the standard model for their camera system in a calibration step, and then undistort the input images accordingly before using them in further processing stages.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

For a variety of applications, including analysis-by-synthesis techniques [7], sensor simulation [8], and special effects in films [9], it is useful to apply the reverse process, i.e. to synthesize images that exhibit realistic distortions by applying a camera model. Using the standard model for this purpose has the advantage that model parameters of existing calibrated cameras can be used directly, with immediate practical benefit to all application areas mentioned above.

In this paper, we present and compare two ways of integrating realistic distortions based on the standard camera model into graphics pipelines. One is based on preprocessing the geometry, and the other is based on postprocessing generated images. We show that both methods have unique advantages and limitations, and the choice of method therefore depends on the application. We verify both approaches by showing that standard model calibration applied to synthesized images recovers the distortion parameters with high accuracy.

2 RELATED WORK

In Computer Graphics, camera models that are more realistic than the pinhole model are typically based on a geometric description of the lens system that is then integrated into ray tracing pipelines [10, 11]. This approach is of limited use if the goal is to render images that match the characteristics of an existing camera, as suitable parameters cannot be derived automatically. Furthermore, this approach excludes rasterization pipelines, which is problematic for applications that benefit from fast image generation.

In contrast, using a Computer Vision camera model allows to apply parameters obtained by calibrating a real camera and, as shown in Sec. 3, can be done in any graphics pipeline.

Sturm et al. [1] give an overview of camera models in Computer Vision. Most models account for radial distortion (e.g. barrel and pincushion distortion, caused by stronger bending of light rays near the edges of a lens than at its optical center) and tangential distortion (caused by imperfect parallelism between lens and image plane). Some also account for thin prism distortion (caused by a slightly decentered lens, modeled via an oriented thin prism in front of a perfectly centered lens), and tilted sensor distortion (caused by a rotation of the image plane around the optical axis).

The complete formulas for the standard model [5] compute distorted pixel coordinates from undistorted pixel coordinates and use parameters k_1, \dots, k_6 for radial distortion, p_1, p_2 for tangential distortion, s_1, \dots, s_4 for thin prism distortion, and τ_1, τ_2 for tilted sensor distortion.

In practice, thin prism distortion and tilted sensor distortion are usually ignored, and radial distortion is limited to two or at maximum three parameters (the others are assumed to be zero). This is documented by the fact that the calibration functions of OpenCV¹ and Matlab/Simulink² estimate only the parameters k_1, k_2, p_1, p_2 and optionally k_3 by default.

In the following, we focus on the standard camera model of Computer Vision, and apply it to arbitrary rendering pipelines via either geometry preprocessing or image postprocessing.

3 METHOD

We first summarize the standard model in Sec. 3.1, focussing on the aspects relevant for this paper and incorporating its intrinsic camera parameters into the projection matrix of a pinhole camera model. On this basis, simulating lens distortion can be done in one of two ways:

- By *preprocessing geometry*. In this approach, each vertex of the input geometry is manipulated such that its position in image space after rendering corresponds to a distorted image.
- By *postprocessing images*. In this approach, an undistorted image is rendered based on the pinhole camera model, and distorted in a postprocessing step based on the standard model.

These approaches are described in detail in the Sec. 3.2 and Sec. 3.3.

¹ https://docs.opencv.org/3.4.0/dc/dbb/tutorial_py_calibration.html

² <https://mathworks.com/help/vision/ug/camera-calibration.html>

```
vec4 clipCoord = P * position;
vec2 ndcCoord = clipCoord.xy / clipCoord.w;
vec2 pixelCoord = vec2(
    (ndcCoord.x * 0.5 + 0.5) * w,
    (0.5 - ndcCoord.y * 0.5) * h);
// apply the standard model to pixelCoord
ndcCoord.x = (pixelCoord.x / w) * 2.0 - 1.0;
ndcCoord.y = 1.0 - (pixelCoord.y / h) * 2.0;
clipCoord.xy = ndcCoord * clipCoord.w;
```

Algorithm 1: GLSL code fragment for applying the standard model in the vertex shader.

3.1 The Standard Model

The standard model, reduced to the part that is relevant in this discussion, has the following parameters: the camera intrinsic parameters, consisting of the principal point c_x, c_y and the focal lengths f_x, f_y (both in pixel units), the radial distortion parameters k_1, k_2 , and the tangential distortion parameters p_1, p_2 . The model computes distorted pixel coordinates u, v from undistorted pixel coordinates x, y by first computing normalized image coordinates s, t with distance r to the principal point, applying the distortion, and then reverting the normalization [5]:

$$s = \frac{x - c_x}{f_x}$$

$$t = \frac{y - c_y}{f_y}$$

$$r^2 = s^2 + t^2 \quad (1)$$

$$d = 1 + k_1 r^2 + k_2 r^4$$

$$u = (sd + (2p_1 st + p_2(r^2 + 2s^2)))f_x + c_x$$

$$v = (td + (p_1(r^2 + 2t^2) + 2p_2 st))f_y + c_y$$

Here, the undistorted pixel coordinates x, y are equivalent to pixel coordinates generated with the pinhole camera model of a standard graphics pipeline when the camera intrinsic parameters c_x, c_y, f_x, f_y are accounted for in the projection matrix. This matrix is typically defined by a viewing frustum given by the clipping plane coordinates l, r, b, t for the left, right, bottom, and top plane. These values have to be multiplied by the near plane value n ; here we assume $n = 1$ for simplicity. Given the image size $w \times h$, suitable clipping plane coordinates can be computed from the camera intrinsic parameters as follows:

$$l = -\frac{c_x + 0.5}{f_x}$$

$$r = \frac{w}{f_x} + l$$

$$b = -\frac{c_y + 0.5}{f_y}$$

$$t = \frac{h}{f_y} + b$$

Using this frustum to define the projection matrix in a standard graphics pipeline accounts for the camera intrinsic parameters of the standard model. The remaining problem is to integrate the lens distortion parameters k_1, k_2, p_1, p_2 . This is discussed in the following sections.

3.2 Preprocessing Geometry

In this approach, each input vertex is manipulated such that its image space coordinates match the distorted coordinates of the standard model.

In a standard graphics pipeline, this manipulation is typically done in the vertex shader. Since the standard model operates on pixel coordinates, we first apply the projection matrix from Sec. 3.1 to each vertex, resulting in clip coordinates, and then divide by the homogeneous coordinate to get normalized device coordinates (NDC). By applying the viewport transformation, these are transformed to window coordinates, which are equivalent to pixel coordinates in the standard model. After modifying the x and y components of the window coordinates to account for lens distortion according to Eq. 1, we transform back to clip coordinates. See Alg. 1 for an OpenGL vertex shader code fragment.

This approach has two limitations.

First, modifying clip coordinates in this way means that a fundamental assumption of the graphics pipeline, namely that straight lines in model space map to straight lines in image space, is no longer fulfilled. This leads to errors. A similar problem occurs in graphics applications that project onto non-planar surfaces, e.g. shadow maps [12] and dynamic environment maps [13] that aim to reduce memory usage. There, the errors are considered acceptable if the tessellation of the input geometry is fine enough such that triangle edges in image space are short. Whether this condition is met in our case depends on the application.

Second, our vertex modification takes place before clipping, and therefore includes vertices that lie outside the domain of the standard model. Depending on the distortion parameters, transforming these vertices may place them into image space, resulting in invalid triangles that ruin the rendering result. To avoid this problem, we discard triangles that contain at least one vertex outside of the view frustum. A tolerance parameter δ can be applied during this test to avoid holes in the final image caused by triangles that are partly inside the frustum: a vertex is discarded if its unmodified NDC xy coordinates lie outside $[-1 - \delta, 1 + \delta]^2$. Since the preprocessing approach requires a finely detailed geometry anyway, simply using $\delta = 0.1$ should work fine. We used this value for all of our tests.

For certain types of distortion, mainly barrel distortion (see Fig. 1), we must additionally account for vertices

that lie outside of the pinhole camera frustum but may be mapped into image space nonetheless. This is done by adding a distortion-dependent value D to the parameter δ . Given the inverse of the standard model (see Sec. 3.3 for details), we can determine a lower bound for D automatically by undistorting the distorted image space corner coordinates $(0, 0), (w, 0), (w, h), (0, h)$, transforming them to NDC coordinates, and setting D to the maximum of the absolute value of each coordinate, minus one.

3.3 Postprocessing Images

In this approach, the scene is first rendered into an undistorted image using an unmodified graphics pipeline based on a pinhole camera with the projection matrix from Sec. 3.1. The result is then transformed into a distorted image by applying the standard model in a postprocessing step, e.g. using a fragment shader.

This postprocessing step requires the computation of undistorted pixel coordinates (x, y) from distorted pixel coordinates (u, v) , i.e. the inverse of Eq. 1. This inversion is not a trivial problem; several approaches exist, but none supports the full set of parameters of the original standard model. For example, Drap and Lefèvre propose an exact inversion, but for radial distortion only [14].

We apply ideas by Heikkilä [3] to invert Eq. 1 using an approximation based on Taylor series. Note that his camera model differs from the standard model; in particular, it computes undistorted pixel coordinates from distorted pixel coordinates. Nevertheless, his inversion process is still applicable. The resulting formulas support radial distortion parameters k_1, k_2 and tangential distortion parameters p_1, p_2 , which is sufficient in practice:

$$\begin{aligned} s &= \frac{u - c_x}{f_x} \\ t &= \frac{v - c_y}{f_y} \\ r^2 &= s^2 + t^2 \\ d_1 &= k_1 r^2 + k_2 r^4 \\ d_2 &= \frac{1}{4k_1 r^2 + 6k_2 r^4 + 8p_1 t + 8p_2 s + 1} \\ x &= (s - d_2(d_1 s + 2p_1 s t + p_2(r^2 + 2s^2)))f_x + c_x \\ y &= (t - d_2(d_1 t + p_1(r^2 + 2t^2) + 2p_2 s t))f_y + c_y \end{aligned} \quad (2)$$

Note that the postprocessing step can only fill areas in the distorted image for which information exists in the undistorted image. For certain types of distortion, mainly barrel distortion (see Fig. 1), this means that some areas of the result remain unfilled. This can only be alleviated by using both an enlarged frustum and an increased resolution when rendering the undistorted

	Preprocessing geometry	Postprocessing images
Distortion model completeness	full	limited to radial and tangential
Prerequisites	finely detailed geometry	none
Result completeness	full	may have unfilled areas
Rendered data types	all	limited to interpolatable, relocatable data
Complexity	geometry-dependent	resolution-dependent

Table 1: Comparison of the pre- and postprocessing approaches to lens distortion rendering based on the standard model. See Sec. 3.4 for details.

image. Note that while it is possible to derive suitable frustum and resolution parameters by computing undistorted coordinates for the distorted image space corner coordinates $(0,0)$, $(w,0)$, (w,h) , $(0,h)$, similar to method described for the preprocessing approach, we did not do so in our tests for simplicity.

3.4 Discussion

In this section, we discuss several aspects of the preprocessing and postprocessing approaches, summarized in Tab. 1.

Distortion model completeness: In the preprocessing approach, we apply the forward standard model and thus can use the full formulas unchanged, i.e. with support for all parameters, including thin prism and tilted sensor distortion if relevant. The postprocessing approach requires the inverse model, and no inversion is known that accounts for all parameters. It is therefore limited to radial and tangential distortion with parameters k_1, k_2, p_1, p_2 , but this should be sufficient for the majority of applications.

Prerequisites: Applying the preprocessing approach requires finely tessellated geometry to keep errors small. Not all applications may be able to make such guarantees. The postprocessing approach does not have this limitation.

Result completeness: While the preprocessing approach can map geometry outside of the pinhole camera view frustum into the distorted image, such information is not available to the postprocessing approach unless an enlarged frustum and increased resolution are used for the undistorted image. See Fig. 1.

Rendered data types: The postprocessing approach will usually map undistorted pixels with interpolation to the distorted image. This is fine e.g. for RGB images, but may break for other kinds of data that special applications may render into images, e.g. object IDs or 2D pixel flow. In these cases, only the preprocessing approach can be applied.

Computational complexity: The complexity of the postprocessing approach depends on the number of vertices in the input geometry, while the complexity of the postprocessing approach depends on the number of output

pixels. While the preprocessing approach can be integrated directly into any pipeline, the postprocessing approach requires an additional render pass.

4 RESULTS

We implemented both the preprocessing and the postprocessing approach in a standard OpenGL rendering pipeline. To verify that our implementation produces results that match the OpenCV/Matlab implementation of the standard model, we varied the model parameters $c_x, c_y, f_x, f_y, k_1, k_2, p_1, p_2$, then rendered a set of 17 images of size 800×600 for each parameter set, containing the standard OpenCV checkerboard calibration pattern in various 3D positions and orientations, and then used the OpenCV `calibrate.py` script to estimate the model parameters from the rendered images. Note that OpenCV also supports a circle grid calibration pattern, but we chose to use the more widely used checkerboard pattern.

In most cases, the original parameters were recovered with high accuracy, even though the rendered set of images was of low quality for calibration purposes. The average recovery error was less than 1% for $c_x, c_y, f_x, f_y, k_2, p_1, p_2$. Interestingly, for k_1 the error was significantly larger, however this did not cause noticeable errors in the undistorted images that were produced for verification purposes. For a few sets, calibration failed, mostly caused by parts of the checkerboard pattern not being visible in some images.

Fig. 2 shows a visual verification: first, an undistorted image is rendered, then a distorted one with a specific set of parameters, and this distorted image is finally undistorted using OpenCV with the same parameters. The first and last image show only minimal differences.

5 CONCLUSION

We presented two methods to accurately render images that match the characteristics of real cameras regarding implicit parameters and lens distortion. Both methods are based on the most widely used camera model in Computer Vision, and can be integrated into any rendering pipeline.

We highlighted the specific advantages and drawbacks of each approach to help implementers pick the right approach for a given application.

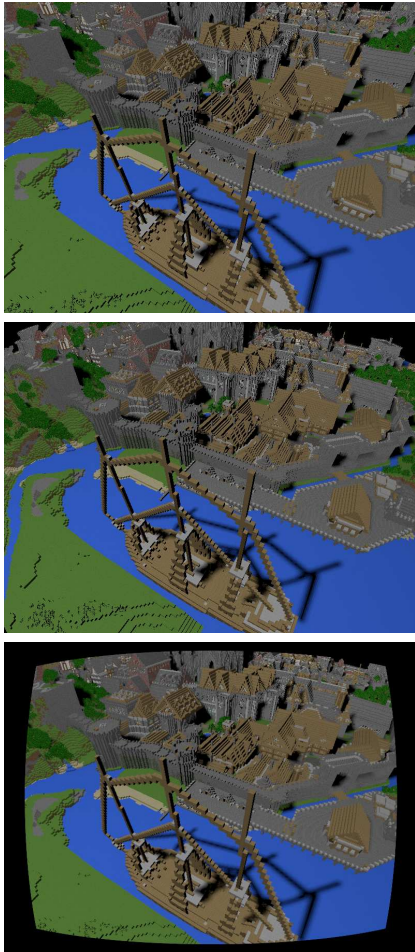


Figure 1: Effects of barrel distortion ($k_1 = -0.11, k_2 = 0, p_1 = 0, p_2 = 0$). From top to bottom: undistorted image, distorted image from preprocessing geometry, and distorted image from postprocessing the undistorted image.



Figure 2: From top to bottom: undistorted image of size 800,600 rendered with intrinsic parameters $c_x = 399.5, c_y = 299.5, f_x = f_y = 400$, distorted image rendered with parameters $k_1 = -0.05, k_2 = 0.01, p_1 = 0.03, p_2 = -0.01$, and undistorted image produced from the distorted image by OpenCV using the same parameters.

6 ACKNOWLEDGMENTS

The work is partially funded by the German Research Foundation (DFG), grants Ko-2960-12/1 and Ko-2960-13/1.

The scene displayed in Fig. 1 and Fig. 2 is the Rungholt scene from McGuire graphics data [15].

7 REFERENCES

- [1] P. Sturm, S. Ramalingam, S. Gasparini, and J. Barreto. *Camera Models and Fundamental Concepts Used in Geometric Computer Vision*. Now Foundations and Trends, 2011.
- [2] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*, pages 1106–1112, Jun 1997.
- [3] J. Heikkilä. Geometric camera calibration using circular control points. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, Oct 2000.
- [4] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.
- [5] OpenCV contributors. OpenCV camera model description. https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html#ga7dfb72c9cf9780a347f3e3d1c47e5d5a. Accessed 2018-03-14.
- [6] Mathworks. Matlab / Simulink camera parameters. <https://www.mathworks.com/help/vision/ref/cameraparameters.html>. Accessed 2018-03-14.
- [7] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 536–551, 2014.
- [8] M. Lambers, S. Hoberg, and A. Kolb. Simulation of time-of-flight sensors for evaluation of chip layout variants. *IEEE Sensors Journal*, 15(7):4019–4026, July 2015.
- [9] D. Roble. Vision in film and special effects. *ACM SIGGRAPH Comput. Graph. Newsletter*, 33(4):58–60, November 1999.
- [10] C. Kolb, D. Mitchell, and P. Hanrahan. A realistic camera model for computer graphics. In *Proc. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 317–324, 1995.
- [11] J. Wu, C. Zheng, X. Hu, and C. Li. An accurate and practical camera lens model for rendering realistic lens effects. In *Int. Conf. on Computer-Aided Design and Computer Graphics*, pages 63–70, September 2011.
- [12] D. Scherzer, M. Wimmer, and W. Purgathofer. A survey of real-time hard shadow mapping methods. *Computer Graphics Forum*, 30(1):169–186, 2011.
- [13] T. Y. Ho, L. Wan, C. S. Leung, P. M. Lam, and T. T. Wong. Unicube for dynamic environment mapping. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 17(1):51–63, Jan 2011.
- [14] P. Drap and J. Lefèvre. An exact formula for calculating inverse radial lens distortions. *MDPI Sensors*, 16(6), 2016.
- [15] Morgan McGuire. Computer graphics archive. <https://casual-effects.com/data>, July 2017.

Robust Range Camera Pose Estimation for Mobile Online Scene Reconstruction

Dmitri Presnov, Martin Lambers, and Andreas Kolb

Abstract—We present a multi-sensor camera tracking method for a real-time 3-D reconstruction on mobile devices. Our approach combines the iterative closest point (ICP) pose estimation using the low-resolution range maps delivered by a PMD pico-flex time-of-flight camera with the 6-DOF pose estimates given by inertial tracking. In contrast to prior approaches, we do not rely on additional sensors, such as 2-D visual odometry. We fuse the results of the inertial tracking with the extrapolated ICP pose estimates using an extended Kalman filter (EKF). Subsequently, the output of the EKF is used as initial guess for the next ICP-based pose estimation. This approach yields an efficient and robust pose tracking for the spatially and temporally low-resolution range data given in mobile applications, and, at the same time, it results in a consistent geometric reconstruction, as the final pose minimizes the error with respect to the scene geometry.

Index Terms—Scene reconstruction, mobile, tracking, extended Kalman filter.

I. INTRODUCTION

REAL-time 3D scene reconstruction from depth data is a well-established research area where several approaches have been proposed [1]–[3]. However, because of high computational requirements and due to the absence of highly integrated range cameras, their implementation was limited for a long time to highend platforms including robots that involving PCs or laptops and rather energy intensive and bulky range cameras such as the Microsoft Kinect. Alternative approaches utilize the standard colour camera of a mobile device, i.e. smartphone or tablet, in order to extract 3D information from RGB image streams [4]–[7]. In the last years, however, the availability of highly integrated Time-of-Flight (ToF) depth cameras such as the Real3™ arensensor from infineon [8] integrated in ToF cameras such as the picoflex from pmd-technologies [9] or in mobile devices such as Lenovo's PHAB2 Pro, makes mobile 3D reconstruction possible.

The 3D scene reconstruction problem requires the estimation of the camera's pose (position and orientation)

and the reconstruction of the scene in parallel. Thus, it is structurally very similar to simultaneous localization and mapping (SLAM), which also can involve ToF cameras [10]. 3D reconstruction and SLAM, however, have different foci: While 3D scene reconstruction aims for high geometric quality of the reconstructed scene, in SLAM trajectory and pose estimation have the highest priority. In this paper, we mainly address 3D scene reconstruction, a survey of visual SLAM approaches can be found, e.g., in [11].

Commonly, in 3D scene reconstruction the camera pose estimation is formulated as a registration problem. Given a sequence of overlapping depth maps delivered by the range camera, the camera pose is estimated by finding the best alignment between two successive depth maps (frame-to-frame registration) or between the current frame with the reconstructed model of all preceding frames (frame-to-model alignment). There are various registration approaches, e.g. iterative closest point (ICP), RANSAC, PCA, or cross-correlation [12], [13]; see Salvi *et al.* [14] for an overview. In online scene reconstruction, the ICP algorithm is commonly applied.

Camera tracking based on, e.g., ICP, is a task demanding high computing and data cost even in the standard, desktop 3D scene reconstruction pipelines. Since the accuracy of 3D map registration heavily depends on the size of overlapping areas and the magnitude of relative transformation that has to be estimated, it requires both, significant CPU and GPU capabilities and high spatial and temporal resolution of the depth data. However, mobile devices with highly integrated ToF cameras comprise restricted computational resources and significantly lower temporal and spatial resolution (see Tab. I), making high quality online 3D scene reconstruction hard to achieve.

On the other hand, online scene reconstruction can benefit from additional sensory information, e.g. from an inertial measurement unit (IMU). A common approach to embed these motion data is to provide them as initial guess for geometric registration, i.e. to ICP. This initial guess can be either obtained by mere integration of the IMU data [15], or by sensor fusion, e.g. with an extended Kalman filter (EKF), that delivers more robust pose predictions. However, because the IMU lacks any complementary data source for position estimation, fusion algorithms require either a third sensory input from, e.g., a camera-based feature tracker [16] or they are restricted to the rotational component only [17]. Some approaches apply a final fusion of the ICP results with other sensory information. This approach, however, is only beneficial, if the ICP pose is less reliable, leading to unwanted geometric reconstruction errors.

Manuscript received December 15, 2017; revised January 18, 2018; accepted January 31, 2018. Date of publication February 5, 2018; date of current version March 9, 2018. This work was supported in part by the German Research Foundation in the context of the Collaborative Research Center (SFB) 1187 Media of Cooperation, and in part by the sub-project A06 Visual integrated medical cooperation. The associate editor coordinating the review of this paper and approving it for publication was Dr. Rosario Morello. (Corresponding author: Dmitri Presnov.)

The authors are with the Computer Graphics Group, Center for Sensor Systems, University of Siegen, Siegen 57068, Germany (e-mail: dmitri.presnov@uni-siegen.de; martin.lambers@uni-siegen.de; andreas.kolb@uni-siegen.de).

Digital Object Identifier 10.1109/JSEN.2018.2801878

1558-1748 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

TABLE I
KEY MEASURES OF A TYPICAL DESKTOP PLATFORM USED IN RECENT PUBLICATIONS, E.G. [23], AND MOBILE DEVICES (TEGRA K1). NOTE THAT THE PERFORMANCE AND FILL-RATE FIGURES ARE IMPERFECT, AS THERE ARE NO STANDARDIZED ACQUISITION PROCESSES

	Desktop Platform	Mobile Platform
CPU Capacity: • CPU/#cores	Intel i7-4790 / 4	A15 ARM / 4
GPU Capacity: • GPU	NV GeForce GTX 980 Ti	NV Kepler
• # Cores / RAM	2816 cores / 6 GB	192 cores / 4 GB
• Perform./Fillrate	5.6 TFLOPS / 176 GTexel/s	0.365 TFLOPS / 7.6 GTexel/s
3D Camera: • Type	Kinect One (ToF)	picoflexx
• Spatial/temp.	512x424 px @ 30 Hz	224x171 px @ 15Hz
• Point Diameter@1.2 m dist.	2.7 mm	6.2 mm
• Noise/Variance@1.2m dist.	1.2 mm	2.8mm

In this paper we propose a lightweight solution for real-time 3D reconstruction on mobile devices. In order to address the challenges arising from a low temporal and spatial depth resolution and the limited computational resolution, we propose a novel multi-sensor tracking, solely using IMU as additional sensory input. In particular, our approach comprises the following contributions:

- We adapt the point-based framework as proposed by Keller *et al.* [3] for mobile online scene reconstruction.
- For initialization of the ICP-based registration, we propose a novel EKF-based fusion approach that robustly estimates rotational and positional pose information using the IMU sensor in combination with extrapolated ICP pose estimations as virtual measurements.
- We demonstrate that our approach minimizes the risk of an ICP failure, particularly in case of a fast camera motion.

II. RELATED WORK

The integration of the inertial tracking and 2D imagery for 3D reconstruction on mobile devices has been investigated in several prior works. In the context of scene reconstruction based on motion stereo, inertial tracking has been used for scene scale estimation and bridging the temporal gap between the two camera frames which are asynchronously acquired [4] or as initial guess of the current pose within the scope of a photo-consistency-based tracking [5]. In [7] an online estimation of the relative rotation between two camera frames by integration the gyroscope data only, combined with the information from a 2D feature point tracker, is used for an offline structure-from-motion algorithm. Other approaches use visual inertial odometry (VIO) as filter-based fusion of inertial and feature point tracking data, e.g. VIO from the Tango library [6].

In the context of scene reconstruction from depth data, several approaches use additional sensory information, such as inertial tracking and visual tracking, in order to improve camera pose estimation.

Kähler *et al.* [17] estimate rotational and translational components of the camera pose separately from different tracking sources. The IMU data in connection with a fusion algorithm are used for orientation estimation whereby ICP and the colour-based tracker only optimize position. This way the rotational drift is reduced.

Klingensmith *et al.* [18] incorporate VIO from a Tango mobile device, which fuses inertial data with visual odometry, based on feature tracking using a fish-eye camera. The low depth frame rate of the range camera of $\approx 3 - 6$ Hz doesn't allow for standard ICP-based pose estimation. Thus, the ICP is initialized with the pose from Tango VIO and refined incrementally. The approach is designed for large scale scene reconstructions with low to moderate reconstruction resolution of 2–3 cm and yields camera drift of ≈ 5 m in the case of long trajectories (≈ 175 m). In [16] the authors further improved their approach by applying a corrective transformation to the Tango-based pose estimation for ICP initialization in order to compensate the VIO drift. The correction term is updated with each depth frame using the difference between the initial guess and the final ICP pose.

Huai *et al.* [19] combine inertial data with ICP and SIFT odometry on a heavy-weight mobile platform consisting of a notebook and a Microsoft Kinect range camera. The integration of IMU readings provides a predicted camera orientation, which is used for initialization and validity check of the ICP. In case ICP fails, SIFT odometry is used. If both ICP and SIFT odometry fail, the incremental motion from the inertial tracking is used as final pose estimation. An EKF is used to correct the IMU-based predictions, where ICP or SIFT odometry serve as measurements for position correction, whereas the states with a small acceleration are utilized for roll and pitch correction by means of the gravity direction from the accelerometer.

In the context of SLAM, Chow *et al.* [20] describe a 3D terrestrial LiDAR system that integrates a MEMS IMU and two Microsoft Kinect sensors to map indoor urban environments in a stop-and-go fashion. The pose estimation is achieved by an implicit iterative extended Kalman filter (IEKF) that predicts poses integrating IMU data and gets measurements from visual tracking. The latter uses alternatively ICP-based point cloud matching initialized by sparse 3D feature point matching (during the “going” state), LiDAR data (during the “stopped” state) or 5-point monocular visual odometry (VO) in highly textured areas or regions with few depth features. The IMU-based pose predictions are also used for initialization of visual tracking.

To the best of our knowledge, current approaches solely relying on IMU data as additional sensory input are restricted to non-lightweight platforms and non-integrated range cameras such as the Microsoft Kinect. Niessner *et al.* [15] combine the

IMU of a mobile device with a Kinect depth sensor on a laptop platform. The inertial tracking integrates IMU data without sensor fusion and provides an estimate of the transformation between last and current camera pose that is used for ICP initialization. The authors demonstrate that their method reduces the number of ICP iterations and makes the tracking more robust in scenarios such as fast camera motion or scanning of planar surfaces. Hervier *et al.* [21] propose a general framework for the fusion of ICP-based pose tracking with data from motion sensors by means of an invariant EKF. The approach uses ICP noise covariances estimates on the basis of the Fisher matrix which allows detection of unobservable directions and prevents that information along these directions flows from ICP “measurements” in a-posteriori pose estimates. The pose prediction from motion data is used to initialize the ICP. Based on this framework, an implementation with a Kinect camera and a tri-axial gyroscope is described and experimentally tested. Camurri *et al.* [22] combine ICP-based camera tracking and inertial tracking in order to solve the SLAM problem for legged robots in a three-folded manner: the pose predicted by means of IMU data is used for ICP initialization, for validity check of the ICP result and, finally, for correction of ICP pose estimates by replacing roll and pitch.

Refocusing on online 3D scene reconstruction, i.e. using additional sensory information for the initialization of the geometric registration, we can conclude that existing approaches either use integrated IMU data as ICP initialization (which requires high temporal depth resolution) or use Kalman filtered IMU for orientation estimation only. More sophisticated approaches use a second sensory input, e.g. a fisheye-based feature tracker, in order to obtain robust position and orientation estimates via Kalman filtering. In contrast to this prior work, our approach requires only IMU as additional sensor information in conjunction with a mobile ToF camera with low spatial and temporal resolution and applies a fusion algorithm including extrapolated ICP poses as virtual EKF measurements in order to obtain a robust ICP initialization.

III. ONLINE SCENE RECONSTRUCTION FROM RANGE MAPS

In this section, we first give a coarse overview of the standard 3D reconstruction pipeline (Sec. III-A) and some implementation details related to porting the standard point-based pipeline to a mobile device (Sec. III-B).

A. 3D Reconstruction Pipeline

The process of scene reconstruction from depth data can be considered as a pipeline whose main steps are depth map preprocessing, camera pose estimation and model update (see Fig. 1, the present structure representation is based on [3], [23], a more generic description can be found for instance in Kolb *et al.* [24].)

In the following we describe the single steps more in detail.

Depth Map Preprocessing: The incoming depth map \mathcal{D}_i is usually filtered for outlier removal and data smoothing.

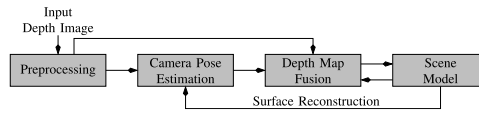


Fig. 1. 3D reconstruction pipeline.

The projection of the 3D point $(x, y, z)^T$ into the image plane of a depth camera is described with the pinhole camera model as

$$z(\mathbf{u}, 1)^T = K(x, y, z)^T \quad (1)$$

where $\mathbf{u} = (u_x, u_y)^T$ are pixel coordinates, $z = \mathcal{D}_i(\mathbf{u})$ the depth value and K is the intrinsic camera matrix. The standard approach for the calculation of 3D positions from depth images is the inverse transformation [2]

$$\mathcal{V}_i(\mathbf{u}) = \mathcal{D}_i(\mathbf{u})K^{-1}(\mathbf{u}^T, 1)^T \quad (2)$$

where \mathcal{V}_i is the vertex map. In addition to the positions, other vertex attributes such as normals, stored in a normal map \mathcal{N}_i , or 3D point sizes will be calculated, depending on the specific approach.

Camera Pose Estimation: The vertex map \mathcal{V}_i created in the first pipeline step contains 3D positions in camera coordinates. However, the fusion of the incoming points with the already reconstructed part of the scene requires a common coordinate system, i.e. the respective camera pose in a reference coordinate frame must be known. A widely used solution for camera pose estimation is ICP [25]. The ICP algorithm iteratively aligns two overlapping geometries estimating the rigid transformation between them. It will be achieved by performing the following steps:

- 1) transform the input geometry to the target coordinate system using the currently estimated transformation,
- 2) for each transformed input point seek for a corresponding (closest) point in the target geometry, and
- 3) for the set of corresponding points minimize the determined alignment error function by adjusting the transformation.

The above procedure is iterated until the alignment error is small enough.

In the context of 3D scene reconstruction, ICP is commonly used to align the incoming depth map \mathcal{D}_i to the so-far accumulated scene model, i.e. in a *frame-to-model* approach. Commonly, the *point-to-plane* alignment error function is used in order to estimate the differential transformation $T_{i \rightarrow (i-1)}$ between two successive camera poses:

$$E(T_{i \rightarrow (i-1)}) = \sum_{\mathbf{u}} ((T_{i \rightarrow (i-1)}\mathcal{V}_i(\mathbf{u}) - \mathcal{V}_{\mathcal{M}, i-1}(\hat{\mathbf{u}})) \cdot \mathcal{N}_{\mathcal{M}, i-1}(\hat{\mathbf{u}}))^2. \quad (3)$$

where $\mathcal{V}_{M,i-1}$ is the model vertex map in the camera coordinates of frame $i - 1$ and $\hat{\mathbf{u}}$ the estimated pixel coordinates of the model point corresponding to $\mathcal{V}_i(\mathbf{u})$. This error function measures the distance between the incoming point and the tangential plane of the model at the corresponding model point.

The ICP algorithm solves a highly non-linear optimization problem. Thus, its successful convergence heavily depends on a good initial guess for the camera pose. In the case of high temporal and spatial resolution range cameras such as the Microsoft Kinect, a simple initialization, such as the identity matrix [2], is fully sufficient. In our case, i.e. for a low temporal and spatial range camera resolution, this leads to misalignments, i.e. the optimization results in a local minimum, which yields significant geometric artifacts (e.g. ghost geometries).

In order to cope with faster motions, the ICP is commonly applied in a hierarchical manner [2]. By setting up an image pyramid, coarser version of the underlying matching problem are deduced. Solving this hierarchical optimization in a coarse-to-fine approach leads to a refinement of the camera pose which can overcome local minima to some extent.

Model Update: After camera pose estimation, the currently incoming range data is fused with the so-far accumulated model data. In the case of point-based geometry representation, as proposed by Keller *et al.* [3], the final correspondences are used and the following attributes are averaged into the model: position, normal, and point size. In order to cope with outliers and isolated points, the model points have a confidence attribute, which counts the number of merges, i.e. observations. Points are tagged as “unstable” or “stable”, if their confidence counter is below or above a given threshold, respectively. Furthermore, incoming points observed from larger distances than the corresponding model point are withdrawn from accumulation, as they are less reliable than the accumulated observations.

B. Implementation Details

In this paper, we basically use the point-based online scene reconstruction framework proposed by [3], [23]. Due to its memory efficiency, this approach does not impose specific modifications with respect to memory or algorithmic layout when porting to a mobile platform. Still, there are some adaptations required in order to use it in the case of range cameras with low temporal and spatial resolution, such as the picoflexx [9]. Highly integrated ToF cameras are designed for near-range and wide-field-of-view types of applications. The wide field of view leads to severe camera distortions which need to be accounted for by camera calibration and in the pre-processing of the range maps. Furthermore, wide field of view in combination with low spatial resolution leads to points with larger sizes and higher noise even at moderate camera-to-object distances and, in average, to fewer model point observations. This effect needs to be taken care of by adapting the system thresholds for, e.g., the confidence counter and the ICP convergence.

IV. MULTI-SENSOR CAMERA TRACKING WITH EKF

ICP is known to produce accurate estimations on the condition that transformations between successive camera poses are relatively small. However, larger transformations can lead to false results due to convergence in a local minimum, or even to a failure due to false correspondences. On a mobile system the probability of such critically large frame-to-frame transformations increases because of a lower range camera frame rate and longer processing times due to limited computational resources. The latter may lead to the situation that not all camera frames can be processed in real time, which results in frame dropping and, consequently, reduces the effective frame rate even more. Our camera tracking approach improves robustness by incorporating inertial sensor data in order to predict the camera pose that is used as ICP initialization. As a consequence, our approach speeds up convergence as it allows reducing the amount of image pyramid levels.

Since integrating noisy IMU data quickly leads to error accumulation, the results of inertial tracking are commonly fused with data from another tracking source, e.g. from a feature point-based visual odometry. An often used sensor fusion algorithm is EKF. In our approach, however, we deliberately do not incorporate any additional tracking device in order to keep hardware efforts as low as possible. Alternatively, we assume moderate change rates in human motion and take advantage of the available ICP camera pose estimations. In particular, we predict future camera poses by extrapolating ICP pose estimates from previous frames and take them as “virtual measurements” that are fused with the IMU integration results using an EKF with the objective to compensate the errors due to the high-frequency IMU noise.

Consequently, our EKF approach aims at estimating ICP results that is considered as true states. Since the true states corresponding to the camera frame i are known, it is preferable to reset the filter to these values and estimate the next frame-to-frame transformation $T_{(i+1) \rightarrow i}$ only, thus keeping the error accumulation as small as possible. However, due to a processing delay in the reconstruction pipeline the last ICP result T_i^{ICP} is not available at t_i , i.e., at the beginning of the next inter-frame interval $i \rightarrow (i + 1)$. We address this issue by splitting the rotational part of the camera pose R_{i+1} in the prior frame orientation R_i (base orientation) and the current frame-to-frame rotation $R_{(i+1) \rightarrow i}$, i.e., $R_{i+1} = R_i \cdot R_{(i+1) \rightarrow i}$. The EKF starts for each interval $i \rightarrow (i + 1)$ with zero frame-to-frame rotation and translation and with an estimate \tilde{R}_i without waiting for the ICP result. These states are propagated by integration of incoming IMU data within the interval. At t_{i+1} , i.e., at the end of the interval, the true previous orientation R_i^{ICP} is always available and can be used in the measurement vector that is passed to the EKF for correction of the predicted states. Due to the split representation of the orientation, it is possible to weight each of its components separately by setting appropriate values in the noise covariance matrices and, thus, to control their influence on the corrector. A generic scheme of the tracking workflow is shown in Fig. 2.

The (partial) filter reset after each inter-frame interval and the weighting of the orientation components according to their

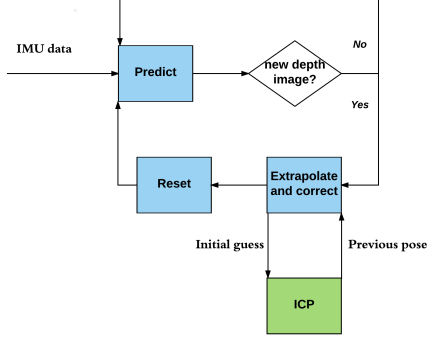


Fig. 2. Scheme of camera tracking. The components of the initial guess estimation module are highlighted in blue, the ICP-based pose estimation module is highlighted in green.

confidences reduces errors in the a-posteriori estimates despite multiple prediction steps without corrections. Furthermore, executing the more computationally expensive correction step only once per frame supports a high tracking performance. In the following we describe the EKF design in more detail.

A. System State and Propagation

In the following orientations are represented as unit quaternions $\mathbf{q} = (s, \mathbf{v}^T)$, where s is the scalar and \mathbf{v} the imagery vectorial component. Below, all quaternions are assumed to be normalized although we omit the normalization step for simplicity. Regarding the time steps, k corresponds to the IMU steps whereas the i refers to ICP frame index.

We define the state vector as

$$\mathbf{x} = [\mathbf{q} \quad \mathbf{v} \quad \mathbf{p} \quad \mathbf{q}_b] \quad (4)$$

where \mathbf{q} is the current frame-to-frame rotation, \mathbf{v} is the velocity, \mathbf{p} is the translation relating to the last camera position (both in global coordinates) and \mathbf{q}_b is the estimated last orientation (base orientation).

Let

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \quad (5)$$

be the discrete-time non-linear process function where \mathbf{u}_k is the control parameter vector and \mathbf{w}_k is the process white noise with normal distribution. \mathbf{u}_k consists of angular velocity $\boldsymbol{\omega}_k$, acceleration \mathbf{a}_k , gyroscope bias $\hat{\mathbf{b}}_g$ and accelerometer bias $\hat{\mathbf{b}}_a$, all expressed in local coordinates. The state propagation

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{f}(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, \mathbf{0}) \quad (6)$$

is specified with the following difference equations:

$$\hat{\mathbf{q}}_{k+1}^- = \frac{1}{2} \Delta t_k \hat{\mathbf{q}}_k^+ \otimes (0, (\boldsymbol{\omega}_k - \hat{\mathbf{b}}_g)^T) \quad (7)$$

$$\hat{\mathbf{v}}_{k+1}^- = \hat{\mathbf{v}}_k^+ + \Delta t_k (\text{qrot}(\hat{\mathbf{q}}_{b,k}^+ \otimes \hat{\mathbf{q}}_k^+, \mathbf{a}_k - \hat{\mathbf{b}}_a) - \mathbf{g}) \quad (8)$$

$$\hat{\mathbf{p}}_{k+1}^- = \hat{\mathbf{p}}_k^+ + \Delta t_k \hat{\mathbf{v}}_k^+ \quad (9)$$

$$\hat{\mathbf{q}}_{b,k+1}^- = \hat{\mathbf{q}}_{b,k}^+, \quad (10)$$

where \cdot^- refers to the a-priori and \cdot^+ to the a-posteriori estimate, Δt_k is the current sample step, \mathbf{g} is the acceleration due to gravity, \otimes represents quaternion multiplication, and $\text{qrot}(\mathbf{r}, \mathbf{s})$ describes rotation of vector \mathbf{s} by unit quaternion \mathbf{r} .

In order to reduce the filter complexity and to increase performance, we assume the gyroscope and accelerometer biases to be constant over time and do not include them in the state vector. Periodical filter resets allow avoiding large error accumulation despite this simplifying assumption. During a short steady phase at the beginning of each experiment we estimate $\hat{\mathbf{b}}_g$ averaging angular velocity measurements over this time interval. Analogously, the averaging across accelerometer readings produces an estimate of the acceleration due to gravity affected by the accelerometer bias in the reference (start) frame $\hat{\mathbf{g}} = \mathbf{g} + \hat{\mathbf{b}}_a^{\text{ref}}$. Thus we can approximate the object acceleration in global coordinates rotating the measured local acceleration \mathbf{a}_k by means of the current estimated device orientation $\hat{\mathbf{q}}_{b,k}^+ \otimes \hat{\mathbf{q}}_k^+$ and subtracting $\hat{\mathbf{g}}$. Thus, the right part of the Eq. (8) can be rewritten as:

$$\begin{aligned} \text{qrot}(\hat{\mathbf{q}}_{b,k}^+ \otimes \hat{\mathbf{q}}_k^+, \mathbf{a}_k - \hat{\mathbf{b}}_a) - \hat{\mathbf{g}} &= \text{qrot}(\hat{\mathbf{q}}_{b,k}^+ \otimes \hat{\mathbf{q}}_k^+, \mathbf{a}_k) - \hat{\mathbf{b}}_a^{\text{ref}} - \mathbf{g} \\ &= \text{qrot}(\hat{\mathbf{q}}_{b,k}^+ \otimes \hat{\mathbf{q}}_k^+, \mathbf{a}_k) - \hat{\mathbf{g}}. \end{aligned}$$

The propagation of the error covariance matrix

$$P_k = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \quad (11)$$

is described according to the standard EKF formulation as

$$P_{k+1}^- = J_f(\hat{\mathbf{x}}_k^+) P_k^+ J_f(\hat{\mathbf{x}}_k^+)^T + Q, \quad (12)$$

where

$$J_f(\hat{\mathbf{x}}_k^+) = \left. \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}_k^+} \right|_{\mathbf{u}_k} \quad (13)$$

is the Jacobian of $\mathbf{f}(\hat{\mathbf{x}}_k^+)$ and Q the process noise covariance matrix.

At t_0 (the beginning of scene reconstruction) the filter is initialized with zero velocity, translation and orientations. In our experiments we set the error covariance matrix to

$$P_0 = 10^{-4} \cdot I_{d_{14}}. \quad (14)$$

As soon as new IMU values are available, the system state vector (Eq. (7)-(10)) and the error covariance matrix (Eq. (12)) are propagated. Since the filter correction is executed only for the last sample of each estimated interval, the intermediate steps are given as:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- \quad \text{and} \quad P_k^+ = P_k^-.$$

B. Measurement Model

The measurement vector is defined as:

$$\mathbf{z} = [\mathbf{q}^e \quad \mathbf{v}^e \quad \mathbf{p}^e \quad \mathbf{q}_b^{\text{ICP}}] \quad (15)$$

where \cdot^e represents extrapolated equivalents of the state vector components, while $\mathbf{q}_b^{\text{ICP}}$ is this previous ICP camera orientation.

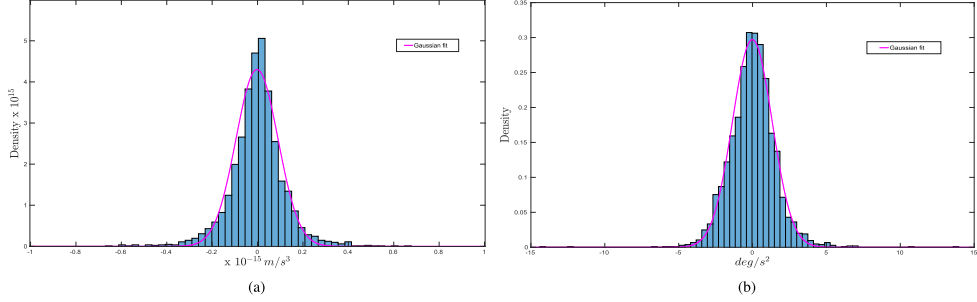


Fig. 3. Examples of estimated pdf of virtual noise. (a) $\delta \mathbf{a}.x$. (b) $\delta \boldsymbol{\omega}.x$.

In order to calculate \mathbf{z}_i , first the velocity $\tilde{\mathbf{v}}_i$, the acceleration $\tilde{\mathbf{a}}_i$ and the angular velocity $\tilde{\boldsymbol{\omega}}_i$ (all in global coordinates) are estimated with a backward Euler method.

$$\tilde{\mathbf{v}}_i = \begin{cases} (\mathbf{p}_i^{\text{ICP}} - \mathbf{p}_{i-1}^{\text{ICP}}) / \Delta t_{i-1}, & \text{if } i > 0 \\ \mathbf{0}, & \text{else,} \end{cases} \quad (16)$$

$$\tilde{\mathbf{a}}_i = \begin{cases} (\tilde{\mathbf{v}}_i - \tilde{\mathbf{v}}_{i-1}) / \Delta t_{i-1}, & \text{if } i > 1 \\ \mathbf{0}, & \text{else,} \end{cases} \quad (17)$$

$$\tilde{\boldsymbol{\omega}}_{\omega,i} = \begin{cases} \frac{2}{\Delta t_{i-1}} (\mathbf{q}_i^{\text{ICP}} - \mathbf{q}_{i-1}^{\text{ICP}}) \otimes \mathbf{q}_{i-1}^{\text{ICP}}, & \text{if } i > 0 \\ (1, \mathbf{0}^T), & \text{else,} \end{cases} \quad (18)$$

where \mathbf{p}^{ICP} , \mathbf{q}^{ICP} are the translational and rotational part of the respective ICP camera pose estimates. Δt_i is the time between ICP frames i and $i+1$, and $\tilde{\boldsymbol{\omega}} = (0, \tilde{\boldsymbol{\omega}})$ contains the angular velocity $\tilde{\boldsymbol{\omega}}$ as its imagery vector component.

After that, the measurements are calculated with an explicit Euler method:

$$\mathbf{q}_{i+1}^e = \mathbf{q}_i^{\text{ICP}} \otimes \left(\frac{1}{2} \Delta t_i \tilde{\boldsymbol{\omega}}_{\omega,i} \otimes \mathbf{q}_i^{\text{ICP}} \right), \quad (19)$$

$$\mathbf{v}_{i+1}^e = \tilde{\mathbf{v}}_i + \Delta t_i \tilde{\mathbf{a}}_i, \quad (20)$$

$$\mathbf{p}_{i+1}^e = \mathbf{p}_i^{\text{ICP}} + \Delta t_i \tilde{\mathbf{v}}_i. \quad (21)$$

The measurement function $\mathbf{h}(\cdot)$ is linear and only copies the a-priori estimates:

$$\mathbf{h}(\hat{\mathbf{x}}_{k+1}^-) = \hat{\mathbf{x}}_{k+1}^-. \quad (22)$$

C. Noise Modeling

Besides bias, whose treatment is explained in IV-A, the IMU measurements are perturbed by white noise \mathbf{w} , which is commonly modeled as Gaussian [26]. \mathbf{w} is considered in the process noise covariance matrix \mathbf{Q} .

The noise in the virtual measurements arises from uncertainties in the estimation of the motion parameters (Eq. (16)-(18)) used by the extrapolation. More precisely, the linear and angular velocities, estimated over an interval $(i-1) \rightarrow i$, and the acceleration over $(i-2) \rightarrow i$, are assumed to be constant over the next interval $i \rightarrow (i+1)$, which generally is an approximation and not the true state. Having the camera

pose T_{i+1}^{ICP} , we can retrospectively calculate the true values \mathbf{v}_i , \mathbf{a}_i and $\boldsymbol{\omega}_i$ substituting the left parts of the Eq. (19)-(21) with ICP results and solving them for the respective motion parameters. Then, the current deviations from the true values, normalized by the time intervals, can be obtained as

$$\Delta \mathbf{v}_i = (\tilde{\mathbf{v}}_i - \mathbf{v}_i) / \Delta t_i \quad (23)$$

$$\Delta \mathbf{a}_i = (\tilde{\mathbf{a}}_i - \mathbf{a}_i) / \Delta t_i \quad (24)$$

$$\Delta \boldsymbol{\omega}_i = (\tilde{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i) / \Delta t_i. \quad (25)$$

We applied the above calculations to the outcomes of our three scene reconstruction experiments in order to estimate the distributions of $\Delta \mathbf{v}$, $\Delta \mathbf{a}$ and $\Delta \boldsymbol{\omega}$ (see Fig. 3(a)-3(b)). The results demonstrate that they follow a Gaussian distribution with zero mean, which justifies modeling them as white Gaussian noise.

Even though, both noise covariance matrices are conceptually related to real-world physical noise measures, we treat them as filter tuning parameters by setting them as constant, diagonal matrices. The matrix values are determined experimentally by manual optimization with respect to filter response and noise rejection as described hereafter. This kind of simplification is repeatedly applied in the context of inertial navigation, e.g. [27], and it yields satisfactory precision at minimal computational costs, i.e. minimal time delays for the overall system (see Sec. V-B).

The values of the noise covariance matrix \mathbf{Q} are determined taking the following considerations into account. The integration of angular velocity (after bias subtraction) delivers rather precise orientation estimations. The velocity estimation, and, even worse, the position estimates, are less precise due to the high-frequency noise in the acceleration values and the double integration. After manual optimization, we set \mathbf{Q} 's diagonal matrix elements that correspond to \mathbf{q} , \mathbf{v} , \mathbf{p} and \mathbf{q}_b to 10^{-5} , 10^{-2} , 10^{-1} and 10^{-5} , respectively. The measurement noise covariance matrix \mathbf{R} is set in an analogous manner. Initially, we conducted some simulation experiments in order to get more insight into the precision of the ICP extrapolation values. We found a higher precision for positional and lower precision for rotational values. The base orientation from ICP, interpreted as true state, gets the highest confidence, i.e. the smallest noise value. After manual optimization, the \mathbf{R} 's values

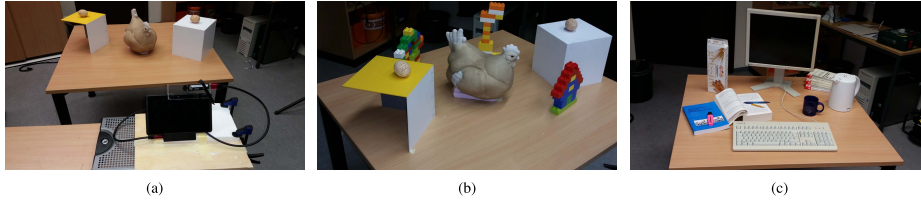


Fig. 4. The test scenes used for evaluation. (a) Hen scene and Tango tablet with mounted picoflex camboard in start position. (b) Hen Duplo scene. (c) Office scene.

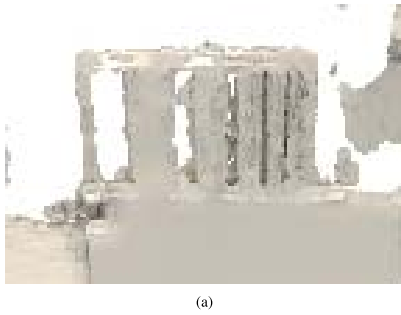


Fig. 5. Reconstruction of the geometric resolution target. Space widths left to right: 22 mm, 14.5 mm, 9 mm, 6 mm, 4 mm, 2.5 mm and 1.5 mm. (a) Reconstructed target.

that correspond to \mathbf{q}^e , \mathbf{v}^e , \mathbf{p}^e and $\mathbf{q}_b^{\text{ICP}}$ are set to 10^{-3} , 10^{-6} , 10^{-5} and 10^{-13} , respectively.

From the perspective of accuracy and precision, more sophisticated approaches to calculate Q and R incorporating, e.g., measured IMU noise values and dynamic adaptation, are promising. However, besides the additional work load and resulting loss in temporal performance, the asymmetrical design of our EKF, i.e. the unequal number of propagation and correction steps, is a major challenge for conceptual design and implementation of these kinds of filters. For instance, a commonly used adaptive approach applies noise covariance matching by means of innovation or residual covariances during each correction step [28], [29]. These covariances are estimated within a sliding window of the n latest measurement updates, and it is assumed that the error covariance matrix P (Eq. 11) is constant within this temporal window [30]. This assumption, however, is unjustifiable in our case, as ≈ 10 system propagations that update P (see Eq. 12) occur between two consecutive measurement updates. In future work, we will investigate more sophisticated, adaptive filter approaches that lead to improved filter response and noise rejection at reasonable computational costs.

D. Correction and Filter Reset

Once a new camera frame $i + 1$ initiates a new cycle of the reconstruction pipeline (see Fig. 1), the estimation of the pose transformation for the interval $i \rightarrow (i + 1)$ is terminated and

the result $\hat{T}_{(i+1) \rightarrow i}^{\text{EKF}}$ is available for the ICP pipeline module. At t_{i+1} , i.e. at the end of the interval, the virtual measurements are computed via ICP extrapolation (Eq. (19)-(21)) and the EKF correction is executed. Due to the linearity of $\mathbf{h}(\cdot)$ the Kalman gain equation can be written in a simplified form:

$$K_{k+1} = P_{k+1}^- (P_{k+1}^- + R)^{-1}. \quad (26)$$

Knowing the Kalman gain, the a-posteriori estimates of the system states and of the error covariance matrix are computed:

$$\hat{\mathbf{x}}_{m_i}^+ = \hat{\mathbf{x}}_{m_i}^- + K_{m_i} (\mathbf{z}_{m_i} - \hat{\mathbf{x}}_{m_i}^-), \quad (27)$$

$$P_{m_i}^+ = (I - K_{m_i}) P_{m_i}^-, \quad (28)$$

where m_i is the last IMU sample index in the current inter-frame interval $i \rightarrow (i + 1)$.

Then the current camera pose estimate is determined as

$$\hat{\mathbf{p}}_{i+1}^{\text{Init}} = \mathbf{p}_i^{\text{ICP}} + \hat{\mathbf{p}}_{m_i}^+, \quad (29)$$

$$\hat{\mathbf{q}}_{i+1}^{\text{Init}} = \mathbf{q}_i^{\text{ICP}} \otimes \hat{\mathbf{q}}_{m_i}^+. \quad (30)$$

After the submission of the pose frame-to-frame component to the ICP module, the EKF is reset for the next inter-frame interval $(i + 1) \rightarrow (i + 2)$. Let

$$\hat{\mathbf{x}}_{0_{i+1}} = \mathbf{g}(\hat{\mathbf{x}}_{m_i}^-, \mathbf{q}_i^{\text{ICP}}) \quad (31)$$

be the reset function that re-initializes the state vector for the interval $(i + 1) \rightarrow (i + 2)$ and operates as:

$$\hat{\mathbf{q}}_{0_{i+1}} = (1, \mathbf{0}^T) \quad (32)$$

$$\hat{\mathbf{v}}_{0_{i+1}} = \hat{\mathbf{v}}_{m_i}^+ \quad (33)$$

$$\hat{\mathbf{p}}_{0_{i+1}} = \mathbf{0} \quad (34)$$

$$\hat{\mathbf{q}}_{b,0_{i+1}} = \mathbf{q}_i^{\text{ICP}} \otimes \hat{\mathbf{q}}_{m_i}^+ \quad (35)$$

Consequently, the errors in the frame-to-frame translation and rotation are not propagated to the next interval and the base orientation includes only an error from the last frame-to-frame rotation estimate. Finally, the error covariance matrix P is re-initialized:

$$P_0 = J_g(\hat{\mathbf{x}}_{m_i}^+) P_{m_i}^+ J_g(\hat{\mathbf{x}}_{m_i}^+)^T, \quad (36)$$

where

$$J_g(\hat{\mathbf{x}}_{m_i}^+) = \left. \frac{\partial \mathbf{g}}{\partial \hat{\mathbf{x}}_{m_i}^+} \right|_{\mathbf{q}_i^{\text{ICP}}} \quad (37)$$

is the Jacobian of $\mathbf{g}(\hat{\mathbf{x}}_{m_i}^+)$. Since the part which relates to the frame-to-frame translation and rotation is cleared to zero, we reinitialize the corresponding diagonal values like in Eq. (14).

TABLE II
HEN SCENARIO: THE ROTATIONAL AND POSITIONAL DRIFT ERRORS, AS WELL AS THE TOTAL NUMBER OF ITERATIONS FOR THE ICP AND THE ICP + IMU METHOD FOR DIFFERENT FRAME DROPPING (ALL, 1/2, 1/4) AND DIFFERENT PYRAMID LEVEL (l = 3, l = 2, l = 1). \emptyset INDICATES ICP FAILURE

Frames	Pyr. lvl.	ICP	ICP	ICP	ICP+IMU	ICP+IMU	ICP+IMU
		l=3	l=2	l=1	l=3	l=2	l=1
All	Rot. drift, °	(176.389)	(37.729)	\emptyset	2.689	2.724	\emptyset
	Pos. drift, m	(1.3352)	(0.6740)	\emptyset	0.0122	0.0126	\emptyset
	Iterations	(9.14±2.76)	(6.56±2.21)	\emptyset	8.34±1.74	5.79±1.52	\emptyset
1/2	Rot. drift, °	\emptyset	\emptyset	\emptyset	2.847	2.836	2.826
	Pos. drift, m	\emptyset	\emptyset	\emptyset	0.0129	0.0127	0.0126
	Iterations	\emptyset	\emptyset	\emptyset	8.59±1.75	6.13±1.68	3.72±1.10
1/4	Rot. drift, °	\emptyset	\emptyset	\emptyset	2.719	\emptyset	\emptyset
	Pos. drift, m	\emptyset	\emptyset	\emptyset	0.0127	\emptyset	\emptyset
	Iterations	\emptyset	\emptyset	\emptyset	9.24±2.16	\emptyset	\emptyset

TABLE III
HEN_DUPLO SCENARIO: THE ROTATIONAL AND POSITIONAL DRIFT ERRORS, AS WELL AS THE TOTAL NUMBER OF ITERATIONS FOR THE ICP AND THE ICP + IMU METHOD FOR DIFFERENT FRAME DROPPING (ALL, 1/2) AND DIFFERENT PYRAMID LEVEL (l = 3, l = 2, l = 1). \emptyset INDICATES ICP FAILURE

Frames	Pyr. lvl.	ICP	ICP	ICP	ICP+IMU	ICP+IMU	ICP+IMU
		l=3	l=2	l=1	l=3	l=2	l=1
All	Rot. drift, °	8.948	\emptyset	\emptyset	8.944	8.954	8.963
	Pos. drift, m	0.1106	\emptyset	\emptyset	0.1106	0.1107	0.1107
	Iterations	8.61±1.74	\emptyset	\emptyset	8.46±1.74	5.74±1.39	3.34±1.01
1/2	Rot. drift, °	\emptyset	\emptyset	\emptyset	8.744	8.742	8.737
	Pos. drift, m	\emptyset	\emptyset	\emptyset	0.1088	0.1088	0.1087
	Iterations	\emptyset	\emptyset	\emptyset	8.96±2.00	6.10±1.69	3.62±1.70

TABLE IV
OFFICE SCENARIO: THE ROTATIONAL AND POSITIONAL DRIFT ERRORS, AS WELL AS THE TOTAL NUMBER OF ITERATIONS FOR THE ICP AND THE ICP + IMU METHOD FOR DIFFERENT FRAME DROPPING (ALL, 1/2) AND DIFFERENT PYRAMID LEVEL (l = 3, l = 2, l = 1). \emptyset INDICATES ICP FAILURE

Frames	Pyr. lvl.	ICP	ICP	ICP	ICP+IMU	ICP+IMU	ICP+IMU
		l=3	l=2	l=1	l=3	l=2	l=1
All	Rot. drift, °	(44.835)	(36.385)	(38.623)	1.089	1.097	0.814
	Pos. drift, m	(0.6259)	(0.4438)	(0.4921)	0.0179	0.0178	0.0114
	Iterations	(9.63±2.33)	(6.85±2.13)	(4.13±1.6)	8.93±1.61	5.95±1.55	3.31±1.05
1/2	Rot. drift, °	\emptyset	\emptyset	\emptyset	1.268	1.270	1.260
	Pos. drift, m	\emptyset	\emptyset	\emptyset	0.0185	0.0186	0.0185
	Iterations	\emptyset	\emptyset	\emptyset	8.97±1.50	6±1.35	3.44±0.89

E. Synchronization

Although incoming depth images and IMU data are provided with timestamps, the respective clocks are unsynchronized. In our implementation we use a simple synchronization approach considering relative times between samples. The EKF is initialized with the first incoming depth image. The arrivals of subsequent depth images serve as synchronization events. At each such synchronization event, we compare the time Δt_D between two last depth image timestamps (camera inter-frame time) and the integration time Δt_{EKF} of

the EKF, accumulated between the events provoked by the corresponding images.

If $\Delta t_{EKF} > \Delta t_D$, the number of integration steps is cut off to match Δt_D . If $\Delta t_{EKF} < \Delta t_D$, we cut off the corresponding IMU raw data at the beginning of the next interval.

V. RESULTS

A. Experimental Setup and Evaluation Criteria

The system described above is implemented on a Tango Yellowstone Tablet with a NVIDIA Tegra K1 processor.

TABLE V
HEN_SLOW SCENARIO: THE ROTATIONAL AND POSITIONAL DRIFT ERRORS, AS WELL AS THE TOTAL NUMBER OF ITERATIONS FOR THE **ICP** AND THE **ICP + IMU** METHOD FOR DIFFERENT FRAME DROPPING (**ALL**, **1/2**, **1/4**) AND DIFFERENT PYRAMID LEVEL (**l = 3**, **l = 2**, **l = 1**). \emptyset INDICATES ICP FAILURE

Frames	Pyr. lvl.	ICP	ICP	ICP	ICP+IMU	ICP+IMU	ICP+IMU
		l=3	l=2	l=1	l=3	l=2	l=1
All	Rot. drift, °	2.286	2.408	2.301	2.348	2.361	2.309
	Pos. drift, m	0.0162	0.0178	0.0166	0.0169	0.0173	0.0167
	Iterations	8.44±1.60	5.82±1.38	3.44±0.99	8.35±1.70	5.55±1.30	3.23±1.47
1/2	Rot. drift, °	2.255	2.289	2.275	2.262	2.254	2.268
	Pos. drift, m	0.0146	0.0150	0.0151	0.0147	0.0147	0.0148
	Iterations	8.63±1.62	6.19±1.52	3.87±1.15	8.58±1.80	5.71±1.24	3.42±1.00
1/4	Rot. drift, °	2.323	(37.044)	(36.609)	2.262	2.254	2.313
	Pos. drift, m	0.0165	(0.4694)	(0.4721)	0.0147	0.0147	0.0160
	Iterations	9.47±2.49	(7.51±2.38)	(5.12±2.06)	8.57±1.80	5.71±1.24	4.13±1.76

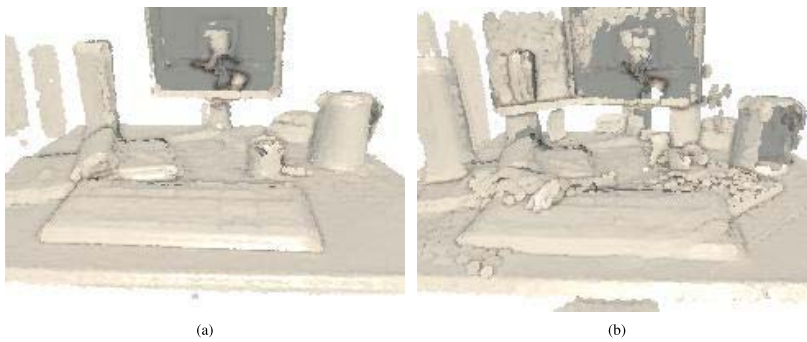


Fig. 6. The **Office** scene reconstruction results. (a) reconstructed with **ICP + IMU** method, **1/2** frames, **l = 1**. (b) shows ghost artifacts by the reconstruction with **ICP** method, **All** frames, **l = 3**.

The range data are provided by an external CamBoard picoflexx fixed on the tablet and connected via USB. The picoflexx provides ≈ 15 FPS at a resolution of 224×171 px. For a comparison of principle characteristics of the equipments used in our handheld system and in a recent desktop solution [23], they are summarized in the Tab. I.

In order to evaluate the achievable accuracy with the given system, we designed a geometric resolution target consisting of bars and spaces of different widths (see Fig. 5(a)). The width decreases from left to right: 22 mm, 14.5 mm, 9 mm, 6 mm, 4 mm, 2.5 mm and 1.5 mm. We capture the target from a distance of 0.5 m, which corresponds to a theoretical point diameter of ≈ 2.5 mm (see Tab. I). Fig. 5(a) shows the result of the target reconstruction. The two largest spaces are clearly recognisable, the next two are partially closed and the space of 4 mm is only distinguishable as a groove without holes. The space smaller than 4 mm cannot be recognized. This result is expected, as the imaging system's optics, e.g. its point-spread function, limits the system resolution.

The depth camera is calibrated against the integrated RGB camera of the tablet using standard calibration methods [31].

In order to calculate the complete calibration matrix we use the transformation between RGB camera and IMU from the Tango library.

We demonstrate an improvement of tracking robustness by our method applying extreme scenarios with fast handheld camera motions and abrupt directional changes. In the experiments we use three different scenes (see Fig. 4(a)-4(c)), for each of them we acquired a data set consisting of range and IMU data. The first scene, with relatively sparse geometric details, is composed of a gypsum hen figure and two cubes with gypsum eggs thereon (**Hen**). For the second data set **Hen** was extended with some Lego Duplo figures (**Hen_Duplo**). The third scene shows an office desk (**Office**). The data was acquired by moving the camera around the respective scene with loop closure. In addition, we recorded the (**Hen**) scene with slower, smoother camera movements (**Hen_Slow**) in order to evaluate the system under moderate motion conditions.

We compare the result of our tracking method (**ICP + IMU**) with the ICP initialized with identity matrix (**ICP**) in consideration of the following aspects.

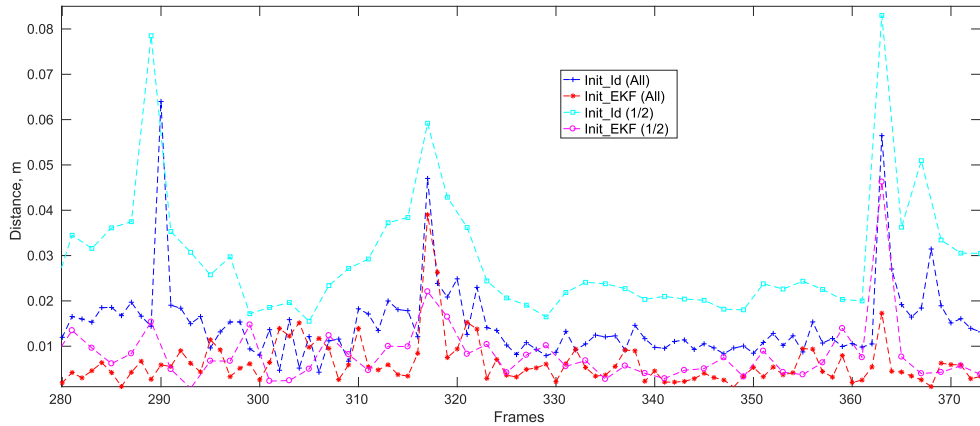


Fig. 7. Distance quality of EKF estimate **Init_EKF** comparing with the standard initialization approach **Init_Id** for the **Office** scenario with frame dropping **All**, **1/2**.

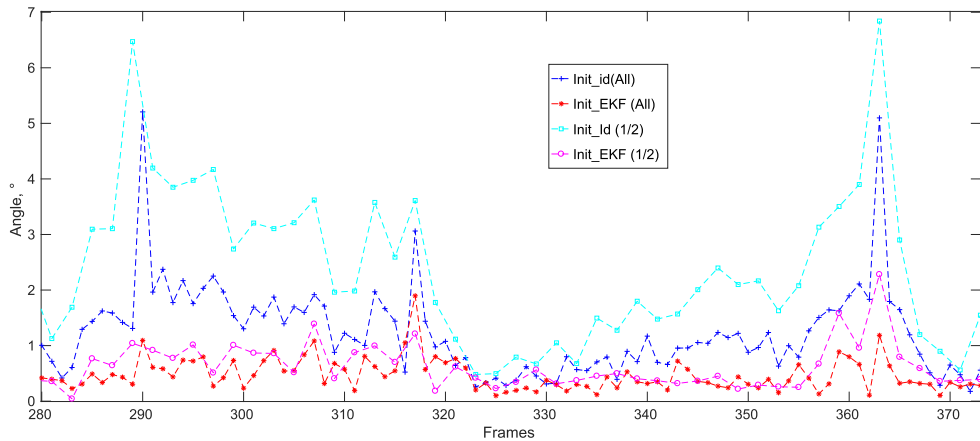


Fig. 8. Angular quality of EKF estimate **Init_EKF** comparing with the standard initialization approach **Init_Id** for the **Office** scenario with frame dropping **All**, **1/2**.

Levels of Image Pyramid: Executing the ICP on several levels of the input image stabilizes the camera pose estimation; see Sec. III-A. However, this hierarchical approach requires additional computations and memory. We demonstrate that due to the enhanced initial guess by **ICP + IMU**, we obtain good correspondence pairs already at the full image resolution, thus decreasing the overall computation and memory requirements. In our experiments we use $l = 3$, $l = 2$ and $l = 1$ pyramid levels.

Frame Dropping: The throughput of the reconstruction pipeline on the given mobile hardware is ≈ 8 FPS while the picoflexx delivers ≈ 15 depth frames per second.

Thus, in real time the pipeline is unable to process all data and has to drop some frames. Such an online frame dropping depends on different internal factors of the operating system and is therefore unpredictable. In order to achieve reproducible results, we used a controlled frame dropping approach in our experiments. This approach processes a predefined set of depth images, independently of the processing time of the mobile device, leading to test sequences **All**, **1/2** and **1/4** that contain all, each second and each fourth depth frame, respectively. The frame dropping enlarges pose transformation between consecutive processed frames and leads to similar inputs as capturing the same motion at a higher speed.

In order to evaluate the overall tracking precision and global drift errors, we setup the camera path as loop-closures, i.e. with the same start and end pose. Thus, we can calculate the positional and rotational drift for each test sequence and pose estimation method as absolute differences between the first and final estimated position and orientation respectively (see Tab. II-V). Large drift errors indicate unreliable pose estimations: in cases where they exceed the specified thresholds (0.15 m for positional and 10° for rotational error), we place the respective results in parentheses, even though ICP itself did not fail, i.e. no matrix singularity occurred.

A reduction of image pyramid does not automatically lead to a smaller total number of ICP iterations since a bad initial guess can slow down the convergence on lower pyramid levels. In order to demonstrate the impact of pyramid reduction on the computational effort, we consider for each experiment the mean and the standard deviation of the total iteration number.

Finally, we evaluate the enhancement of the initial guess by our method considering the difference in position and orientation between the EKF estimate, i.e., the ICP's initial pose, and the final pose after ICP (**Init_EKF**) and comparing it with the respective difference between the last and current ICP pose (**Init_Id**), which corresponds to the common ICP initialization with the identity matrix.

Besides the demonstration of an improvement over the standard ICP initialization, we also compare our method with the Tango VIO-based initialization approach proposed in [16]. For this purpose, we saved the respective pose from Tango tracking for each incoming depth image during the **Office** scene capturing. After the scene is reconstructed, we calculate the pose predictions offline, using the stored Tango data and the ICP results.

Again, we obtain differences between Tango-based predictions and ICP poses (**Init_Tango**).

B. Evaluation

The experimental results in Tab. II-IV show a considerable improvement of tracking robustness by **ICP + IMU**. Considering fast motion scenarios, **ICP** delivers reliable results only for one configuration, i.e. the **Hen_Duplo** scenario (Tab. III) with all pyramid levels and all frames (which is de-facto not achievable in online mode), whereas **ICP + IMU** yields valid reconstruction in almost all scenarios and configurations. Fig. 6(b) shows visual artifacts due to a large drift error in the **Office** scene reconstructed by **ICP** using **All** frames and **l = 3**. The comparison Fig. 6(a) shows the more precise reconstruction results achieved with **ICP + IMU** using fewer frames (**1/2**) and pyramid level (**l = 1**). Note that the artifacts on the computer screen are attributable to the general weakness of ToF cameras in capturing strongly light absorbing, i.e. dark, surfaces.

For the frame dropping rate **1/2** that is close to the online frame dropping due to the hardware limitations, **ICP + IMU** works successfully in all experiments, which demonstrate real-time capability of our method. Furthermore, **ICP + IMU** preserves tracking stability with fewer pyramid levels in most cases. In particular, in all tests with the “online-like” frame

dropping **1/2** the pose estimation on the basis of original image resolution (**l = 1**) was possible.

On the other hand, in the experiment with a slow camera motion (Tab. V), suitable for scene reconstruction with **ICP**, the application of **ICP + IMU** doesn't deteriorate tracking results and exhibits higher robustness under the highest frame dropping (**1/4**) with a reduced image pyramid (**l = 2, l = 1**).

As can be seen in Tab. III and V, in those experiments where both methods have delivered reliable results, the mean iteration number in **ICP + IMU** is only marginally below the one of **ICP**. In general, however, **ICP + IMU** achieves a robust result with considerable fewer overall iterations when fewer pyramid levels are used. As expected, the drift errors are comparable in cases where **ICP** and **ICP + IMU** deliver reliable results.

Concerning the difference between the initial guess and final pose, Fig. 7 and 8 show the absolute errors for a sequence of ≈ 130 frames of the **Office** scenario with **l = 1** for frame dropping **All** and **1/2**. In the vast majority of the frames there is a significant improvement by the EKF prediction. The overall relative error **Init_EKF/Init_Id** for the range image sequence of **All** frames is 0.5392 ± 0.6965 for the mean relative positional and 0.4556 ± 0.3697 for the mean relative rotational error. Dropping half of the frames (**1/2**), we get a mean relative position error of 0.4348 ± 0.4546 and a mean relative rotational error of 0.3283 ± 0.2876 . Thus, for faster motion we get relatively better initializations using our method.

As a comparison, the Tango-based initialization according to [16] produces with the frame sequence **1/2** a mean relative error **Init_Tango/Init_Id** of 0.2926 ± 0.3147 in position and 0.4349 ± 0.4659 in orientation. The results demonstrate that our method is only slightly less precise in position prediction, and at the same time has a slightly higher precision in orientation prediction. In summary, we can obtain an initial guess precision comparable with the above VIO method without requiring a fisheye camera with large opening angle and high temporal and spatial resolution.

The processing of an individual input depth frame according to Fig. 1 requires ≈ 23 ms for the preprocessing stage, 20 – 25 ms and ≈ 11 ms for the pose estimation stage in the **l = 3** and the **l = 1** case, respectively, 41 – 51 ms for the depth map fusion, and 35 – 57 ms for the surface reconstruction.

C. Limitations

1) *Gap in Sensor Sampling Rate*: A higher sampling frequency of IMU allows to bridge longer intervals between depth images. However, the integration of raw IMU data leads to a considerable sensor drift due to error accumulation (see discussion in Sec. IV). Although our method partially corrects the integration errors by the EKF at the end of each inter-frame interval, the error increases for longer temporal gaps between successive processed depth images, causing the a-posteriori estimates of the EKF be less precise. Furthermore, larger inter-frame intervals lead to a decreased reliability of the ICP pose extrapolations, particularly in the case of abrupt changes of motion direction. Thus, the difference between initial guess and final pose in our method increases by higher frame dropping rates albeit, as a general tendency, remaining

TABLE VI
 OVERVIEW OF THE FUNDAMENTAL CHARACTERISTICS OF MOBILE APPROACHES FOR SCENE RECONSTRUCTION FROM RANGE DATA:
 REQUIRED SENSORS, APPLICATION OF SENSOR FUSION, INTERNAL MODEL REPRESENTATION AND
 SYSTEM PERFORMANCE (VALUE IS N/A IF NO DATA ARE AVAILABLE)

	Ours	Dryanovski et al. [16]	Kähler et al. [17]
Sensors: • Range camera resolution / FPS • IMU • Additional sensors	224x171 / 15 + -	n/a / 5 + RGB Cam	320x240 / 60 + -
Sensor fusion: • Estimated parameter • Input sources • Target	Position+orientation IMU+ICP ICP initial guess	Position+orientation IMU+RGB Cam ICP initial guess	Orientation IMU Rotational part of end pose
Scene representation / performance: • Type of representation • Object space resolution • Model extent • FPS	Sparse points 2.5 mm Unlimited 9	Spatially hashed grid 2 cm Unlimited ≥ 5	Spatially hashed grid n/a Unlimited 47

below this parameter in the standard approach. For example, only one “fast” data set (**Hen**) can be processed with the dropping rate $1/4$ (see Tab. II).

The only way to counteract this problem is to reduce the inter-frame gap by faster depth frame processing, which, in turn, requires novel reconstruction approaches.

2) *Noise Estimation*: A further aspect that can limit the prediction precision is low accuracy of the noise estimation. As described in Sec. IV, we tuned the process noise covariance matrix experimentally, however, a measurement-based noise estimation, for instance using Allan variance [32] might provide more accurate results. Concerning the noise of ICP extrapolations, an online calculation depending on the length of the inter-frame interval may lead to a more appropriate noise estimation model.

3) *ICP Failure Handling*: A good initial guess of the current camera pose can prevent ICP errors, such as convergence in local minima or false correspondence association, related to large frame-to-frame transformations. However, a lack of geometric features, e.g. when sliding along planar geometries, also results in ICP stability problems. In these cases the ICP and EKF estimations diverge. If this constellation appears in several successive frames, tracking failure may occur. The experiment **Hen, All, 1/2, 1=1** (Tab. II) shows that in this particular case even discarding frame prevents tracking failure.

A possible solution for the above problem is to discard ICP results for such frames and continuing inertial-based estimation until new reliable ICP estimates are available. This requires sophisticated ICP error detection [15], [21]. This improvement is orthogonal to our proposed method and will be integrate in our system as part of the future work.

VI. CONCLUSIONS

In this paper we presented a novel, lightweight solution for real-time 3D reconstruction on mobile devices that uses IMU data as the only additional sensory input. Our pose estimation incorporates a novel EKF-based fusion of inertial tracking data with extrapolated ICP camera poses in order to initialize the ICP pose estimation. We demonstrate considerable enhancement of the tracking robustness in comparison with

the common initialization approach. On the one hand, our method shows a higher stability in the case of fast camera motion. On the other hand, our approach reliably deals with the low temporal resolution of highly integrated depth cameras such as the picoflexx. In addition, the robust ICP initialization allows to reduce the number of ICP image pyramid levels and, consequently, to achieve a higher depth frame throughput. All in all, our approach minimizes the negative impact of hardware limitations existing on a mobile system and allows 3D point-based scene reconstruction for fast camera motion. Tab. VI summarizes some distinctive characteristics of the approaches for scene reconstruction from range data on mobile devices, illustrating the most important conceptual differences between our method and existing solutions.¹

In future work we will improve the robustness of our tracking method in scenarios with sparse geometric features by integrating a corresponding ICP failure detection logic. Thus, also ICP failures due to scenes with low geometric feature can be handled more robustly. Furthermore, we aim to enhance the accuracy of EKF by means of an adaptive noise estimation.

ACKNOWLEDGMENTS

We would like to thank Dr. Holger Nies for his time in discussing details of the Kalman-filtering.

REFERENCES

- [1] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, “Real-time 3D model acquisition,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 438–446, 2002.
- [2] R. A. Newcombe et al., “KinectFusion: Real-time dense surface mapping and tracking,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.
- [3] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, “Real-time 3D reconstruction in dynamic scenes using point-based fusion,” in *Proc. Int. Conf. 3D Vis. (3DV)*, 2013, pp. 1–8.
- [4] P. Tanskanen, K. Kolev, L. Meier, F. Camposco, O. Saurer, and M. Pollefeys, “Live metric 3D reconstruction on mobile phones,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 65–72.

¹The parameter “Model extent” in this table refers to the extent limitations due to the system design. Note that the real model size is also limited by available memory.

- [5] P. Ondrůška, P. Kohli, and S. Izadi, "MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 11, pp. 1251–1258, Nov. 2015.
- [6] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys, "3D modeling on the go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *Proc. Int. Conf. 3D Vis.*, Oct. 2015, pp. 291–299.
- [7] O. Muratov, Y. Slynko, V. Chernov, M. Lyubimtseva, A. Shamsuarov, and V. Bucha, "3DCapture: 3D reconstruction for a smartphone," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPR)*, Jun./Jul. 2016, pp. 291–299.
- [8] Infineon. (2015). *Real3 (TM) Image Sensor Family—3D Depth Sensing Based on Time-of-Flight*. [Online]. Available: http://www.infineon.com/sgdl/Infineon-REAL3%20Image%20Sensor%20Family-PB-v01_00-EN.PDF?fileId=5546d462518ffd850151a0afc2302a58
- [9] PMD Technologies GmbH. (2015). *Reference Design Brief CamBoard Pico Flexx*. [Online]. Available: http://pmdtec.com/picoflexx/downloads/PMD_RD_Brief_CB_pico_flexx_V0201.pdf
- [10] S. May, D. Droschel, D. Holz, C. Wiesen, and S. Fuchs, "3D pose estimation and mapping with time-of-flight cameras," in *Proc. IROS Workshop 3D Mapping*, 2008, pp. 1–6.
- [11] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, 2015.
- [12] G. Burel and H. Henoco, "Determination of the orientation of 3D objects using spherical harmonics," *Graph. Models Image Process.*, vol. 57, no. 5, pp. 400–408, 1995.
- [13] R. L. Larkins, M. J. Cree, and A. A. Dorrington, "Analysis of binning of normals for spherical harmonic cross-correlation," *Proc. SPIE*, vol. 8290, p. 82900L, Jan. 2012.
- [14] J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," *Image Vis. Comput.*, vol. 25, no. 5, pp. 578–596, 2007.
- [15] M. Nießner, A. Dai, and M. Fisher, "Combining inertial navigation and ICP for real-time 3D surface reconstruction," in *Proc. EUROGRAPHICS*, 2014, pp. 13–16.
- [16] I. Dryanovski, M. Klingensmith, S. S. Srinivasa, and J. Xiao, "Large-scale, real-time 3D scene reconstruction on a mobile device," *Auto. Robots*, vol. 41, no. 6, pp. 1423–1445, 2017.
- [17] O. Köhler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 11, pp. 1241–1250, Nov. 2015.
- [18] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "CHISEL: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields," *Robot., Sci. Syst.*, vol. 4, Jul. 2015.
- [19] J. Huai, Y. Zhang, and A. Yilmaz, "Real-time large scale 3D reconstruction by fusing Kinect and IMU data," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. II-3/W5, pp. 491–496, Aug. 2015.
- [20] J. C. K. Chow, D. D. Lichti, J. D. Hol, G. Bellusci, and H. Luinge, "IMU and multiple RGB-D camera fusion for assisting indoor stop-and-go 3D terrestrial laser scanning," *Robotics*, vol. 3, no. 3, pp. 247–280, 2014.
- [21] T. Hervier, S. Bonnabel, and F. Goulette, "Accurate 3D maps from depth images and motion sensors via nonlinear Kalman filtering," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5291–5297.
- [22] M. Camurri, S. Bazeille, D. G. Caldwell, and C. Semini, "Real-time depth and inertial fusion for local SLAM on dynamic legged robots," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst. (MFI)*, Sep. 2015, pp. 259–264.
- [23] D. Lefloch, M. Kluge, H. Sarbolandi, T. Weyrich, and A. Kolb, "Comprehensive use of curvature for robust and accurate online surface reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2349–2365, Dec. 2017.
- [24] A. Kolb, J. Zhu, and R. Yang, "Sensor fusion," in *Digital Representations of the Real World*. Boca Raton, FL, USA: CRC Press, 2015, ch. 9, pp. 133–150.
- [25] P. J. Besl and D. N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
- [26] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. Stevenage, U.K.: IET, 2004.
- [27] A. Cavallo *et al.*, "Experimental comparison of sensor fusion algorithms for attitude estimation," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 7585–7591, 2014.
- [28] R. K. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Trans. Autom. Control*, vol. 15, no. 2, pp. 175–184, Apr. 1970.
- [29] J. Wang, "Stochastic modeling for real-time kinematic GPS/GLONASS positioning," *Navigation*, vol. 46, no. 4, pp. 297–305, 1999.
- [30] F. Aghili and C.-Y. Su, "Robust relative navigation by integration of ICP and adaptive Kalman filter using laser scanner and IMU," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 4, pp. 2015–2026, Aug. 2016.
- [31] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.
- [32] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008.


Dmitri Presnov received the M.Sc. degree in computer science from the University of Siegen, Germany, in 2017. He is currently a Researcher with the Computer Graphics and Multimedia Systems Group, University of Siegen. His research interests are 3-D scene reconstruction on mobile environments and medical visualization.

Martin Lambers received the Diploma degree in mathematics from the University of Münster, Germany, in 2006, and the Dr.Ing. degree in computer science from the University of Siegen, Germany, in 2011. He is currently a Senior Researcher with the Computer Graphics and Multimedia Systems Group, University of Siegen. His research interests include simulation, processing, analysis, and visualization of multimodal sensor data.

Andreas Kolb received the Ph.D. degree from the University of Erlangen, Germany, in 1995. He is currently the Head of the Computer Graphics and Multimedia Systems Group, University of Siegen, Germany. He is also the Spokesman of the Research Training Group *Imaging New Modalities*, funded by the German Research Foundation. His research interests include computer graphics and computer vision, including particle-based simulation and visualization, light-fields, real-time simulation, processing, and visualization of sensor data.

Article

Quantified, Interactive Simulation of AMCW ToF Camera Including Multipath Effects

David Bulczak * , Martin Lambers and Andreas Kolb

Computer Graphics Group, Institute for Vision and Graphics, University of Siegen, 57076 Siegen, Germany; martin.lambers@uni-siegen.de (M.L.); andreas.kolb@uni-siegen.de (A.K.)

* Correspondence: david.bulczak@uni-siegen.de; Tel.: +49-271-740-2603

Received: 27 October 2017; Accepted: 14 December 2017; Published: 22 December 2017

Abstract: In the last decade, Time-of-Flight (ToF) range cameras have gained increasing popularity in robotics, automotive industry, and home entertainment. Despite technological developments, ToF cameras still suffer from error sources such as multipath interference or motion artifacts. Thus, simulation of ToF cameras, including these artifacts, is important to improve camera and algorithm development. This paper presents a physically-based, interactive simulation technique for amplitude modulated continuous wave (AMCW) ToF cameras, which, among other error sources, includes single bounce indirect multipath interference based on an enhanced image-space approach. The simulation accounts for physical units down to the charge level accumulated in sensor pixels. Furthermore, we present the first quantified comparison for ToF camera simulators. We present bidirectional reference distribution function (BRDF) measurements for selected, purchasable materials in the near-infrared (NIR) range, craft real and synthetic scenes out of these materials and quantitatively compare the range sensor data.

Keywords: time-of-flight; sensor simulation; BRDF

1. Introduction

Amplitude modulated continuous wave Time-of-Flight (AMCW-ToF) depth sensors provide per-pixel distance information by estimating the phase shift of a received amplitude modulated light signal that has been emitted by an active light source using a reference signal [1]. This phase shift is proportional to the time light traveled from the light source to the sensor. Despite recent rapid development progress, AMCW ToF cameras still suffer from several error sources. Some of the major effects that severely influence AMCW ToF range measurements are motion artifacts, flying pixels and multipath interference (MPI).

The simulation of AMCW ToF cameras including reproduction of the major sensor effects benefits the development of new sensors by allowing tests of variations to the sensor design [2], as well as the development of down-stream data processing algorithms by providing ground truth and test data [3,4]. AMCW ToF simulation requires modeling the illumination, the light propagation in the scene, and the individual sensor pixel behavior. Furthermore, computationally efficient approaches are of great importance for simulating dynamic scenes and/or for parameter studies in hardware layout and algorithm design [2,5].

Up to now, there has only been very little research in multipath interference (MPI) simulation for AMCW ToF cameras. Meister et al. simulate MPI using *non-interactive* global illumination schemes, which implicate very high computational costs of approximately 2 h per depth image, including simple analytic bidirectional reflectance distribution functions (BRDFs) [4]. Furthermore, there is very little research in providing *quantitative* comparison to *real-world* measurements, which is indispensable to reliably predict the behavior of prospective AMCW ToF cameras and their application.

Such a comparison needs to take real scene material properties into account, typically provided by BRDF measurements. We are not aware of any work that uses such measurements at the relevant near-infrared wavelength for AMCW ToF simulation.

In this paper, we present a physically based simulation method for AMCW ToF cameras that runs on interactive frame rates. Our approach is based on Lambers et al. [2], which already accounts for physical units. Our simulation approach is fully GPU-based and comprises the following contributions:

- Enhancement of the *Reflective Shadow Map (RSM)* algorithm [6] for GPU-based, *interactive, single-bounce, image-space, multipath interference* simulation.
- *BRDF-based reflection simulation* for measured real-world materials.
- Extension of the simulation model to include realistic electronic and optical shot noise.

Furthermore, this paper presents evaluation approaches for ToF simulations based on data captured by real cameras with the following contributions:

- Measurement of isotropic BRDF at 850 nm wavelength for several specified materials that can be purchased worldwide and thus can be used to reproduce scenes reliably.
- Quantitative evaluations of the proposed simulator based on AMCW ToF camera acquisition of real-world reference scenes. We clearly see the improved ToF simulation results of our single-bounce approach over direct simulation in terms of quality, and over higher-order global illumination simulation [4] in terms of computational performance.
- Publicly available simulator, BRDF data including references to material vendors, geometry of the reference scenes, and real AMCW ToF camera measurements, in order to promote further activities in quantitative evaluation of AMCW ToF simulation.

2. Related Work

Keller and Kolb [5] present a GPU-based AMCW ToF simulation approach that computes light propagation in real-time by using basic rasterization techniques, including local illumination with a Lambertian reflection model. Their simulation approach generates so-called *phase images*, i.e., the raw images acquired by a AMCW-ToF camera, and can reproduce spatio-temporal artifacts such as flying pixels and motion artifacts.

Meister et al. [4] propose an AMCW ToF simulation method that adopts a global illumination technique, i.e., bidirectional path tracing, to simulate multipath interference. This approach is computationally very expensive and only suitable to simulate static scenes. They provide a visual comparison with real data on range image basis for two scenes (“corner” and “box”) as well as limited quantitative comparison of simulated data with real data captured with a PMDTec CamCube 3 (pmd technologies ag, Siegen, Germany). Neither real material properties are acquired nor used within their simulator.

Alternative AMCW ToF sensor simulation approaches have a stronger focus on the sensor hardware. Schmidt and Jähne [7] model optical excitation and target response to simulate the conversion of photons to electrical charges. Their approach does not simulate light propagation and illumination.

Lambers et al. [2] introduce a realistic sensor model to simulate both the photometric relations in the scene including light propagation and illumination, and physically correct charges at a sensor pixel’s readout circuit level, that result from incoming photons. Their simulation is limited to scene materials that are Lambertian reflectors. They provide quantitative comparison of AMCW ToF camera simulation data with real captured data, but their evaluation is limited to sensor pixel based plausibility checks of their simulation model and ignores characteristic error sources.

So far, none of the existing AMCW ToF simulation approaches is capable of simulating MPI effects at interactive rates, none take realistic scene material properties into account in the light propagation simulation, and no quantitative evaluation is available at range image level.

Ritschel et al. [8] give an overview of interactive global illumination methods. These approximations of global illumination are more suitable to simulate MPI effects at interactive rates than the more general, but computationally much more expensive methods, such as the bidirectional path tracing used by Meister. Image-space approximations of global illumination such as Reflective Shadow Maps [6] are especially efficient and sufficient for our use case despite their limitations.

In addition to global illumination approximation, realistic scene materials need to be taken into account to allow comparisons of simulated results and measurements. Most existing BRDF databases for material properties, such as the widely known MERL [9] and CURET [10] databases, focus on visible wavelengths and are typically limited to RGB channels.

Lacking near-infrared BRDF measurements, Mutny et al. [11] use an Oren–Nayar BRDF model with parameters fitted from the CURET database for AMCW ToF simulation using Meister’s simulator to create a database of scenes for correcting multipath artifacts based on a machine learning approach.

Choe et al. [10] recently published the first BRDF database for near-infrared wavelengths, demonstrating that material properties at these wavelengths may differ significantly from those at visible wavelengths. They focus on finely-structured materials such as fabrics. In contrast, in this work, we focus on standardized materials that are available for purchase worldwide, in order to allow reliable reproduction of scenes with defined material properties.

3. Time-of-Flight Simulation

Our simulation model is based on Lambers et al. [2] that consists of two parts, the direct light propagation (Section 3.1) and sensor pixel behaviour (Section 3.2). We describe our extension of this model with respect to BRDF-based materials (Section 3.1), multipath effect simulation via global illumination approximation (Section 3.3), and a realistic noise model (Section 3.4). For further information on the AMCW ToF principle, we refer the reader to, e.g., [1,2,12].

3.1. Direct Light Propagation

The model described by Lambers et al. [2] assumes Lambertian reflectors only. We directly extend this model to use BRDFs. Starting with the power $P_L [W]$ of the camera light source L , we can deduce the radiant intensity $I(\theta_{L \rightarrow P})$ from L to a surface point P . This radiant intensity can be assumed constant for an isotropic light source model, or taken from a vendor-provided intensity table depending on the angle $\theta_{L \rightarrow P}$ between the main light direction \vec{n}_L and $P - L$; see Figure 1b.

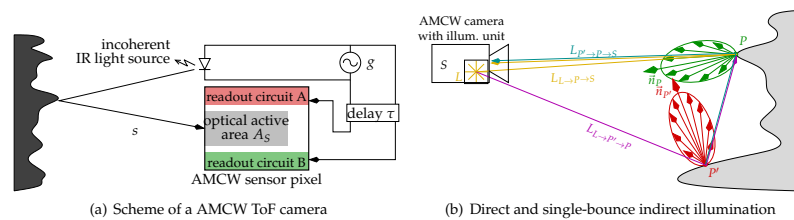


Figure 1. Scheme of an AMCW ToF camera including the pixel layout and the optically active pixel area A_S (gray) with the two readout circuits A (red) and B (green) (a); direct illumination and one path of single-bounce indirect illumination in the AMCW ToF simulation (b).

The irradiance $E_{L \rightarrow P} [W/m^2]$ of P resulting from direct illumination is:

$$E_{L \rightarrow P} = I_L(\theta_{L \rightarrow P}) \frac{\cos \theta_{P \rightarrow L}}{\|P - L\|^2}, \quad (1)$$

where $\theta_{P \rightarrow L}$ is the angle between $L - P$ and the surface normal \vec{n}_P at P . This notation will be used throughout the following derivations.

The resulting direct-illumination radiance $L_{L \rightarrow P \rightarrow S}$ from surface point P to sensor S depends on the material of the surface, described by its BRDF $f_{L \rightarrow P \rightarrow S}$ at P for incoming light direction $L \rightarrow P$ and outgoing light direction $P \rightarrow S$:

$$L_{L \rightarrow P \rightarrow S} = E_{L \rightarrow P} \cdot f_{L \rightarrow P \rightarrow S}. \quad (2)$$

3.2. Sensor Pixel Model

The AMCW ToF sensor consists of an array of $w \times h$ sensor pixels. Each sensor pixel accumulates charges in the readout circuits, A and B , depending on its irradiation, see Figure 1a. The irradiance E_S of the pixel's photosensitive area A_S resulting from direct illumination is given as

$$E_{L \rightarrow P \rightarrow S} = L_{L \rightarrow P \rightarrow S} \cdot \cos \theta_{S \rightarrow P}. \quad (3)$$

$E_{L \rightarrow P \rightarrow S}$ determines the optical power $P_S [W]$ and from that the energy $W_S [J]$ that is accumulated in one pixel over the integration time T for a common AMCW duty cycle of 50%:

$$P_S = E_S \cdot A_S, \quad W_S = P_S \cdot T \cdot 0.5. \quad (4)$$

The conversion into electron-hole pairs in the pixel depends on the quantum efficiency ν_q , which describes how many electrons are generated per incoming photon, and the wavelength λ :

$$N_{\text{tot}} = \frac{W_S}{\nu_q \cdot \frac{q \cdot \lambda}{h \cdot c}}, \quad (5)$$

where h is the Planck-constant, c is the speed of light, and q is the value of elementary charge.

This total charge $N_{\text{tot}} = N_A + N_B$ is accumulated in the two circuits A and B depending on the phase shift ϕ , the internal phase delay τ , and the achievable demodulation contrast $D \in [0, 1]$:

$$N_A = \frac{N_{\text{tot}}}{2} (1 + D \cdot f(\tau, \phi)), \quad N_B = \frac{N_{\text{tot}}}{2} (1 - D \cdot f(\tau, \phi)). \quad (6)$$

Here, f is the *correlation function* resulting from the mixing of the optical signal s with the delayed reference signal g ; see Figure 1a. Commonly, it is assumed that g and s are ideal cosine shaped functions, which results in a cosine shaped correlation function $f(\tau, \phi) = \cos(\tau + \phi)$.

3.3. Multipath Simulation

The simulation model of Lambers et al. [2] is restricted to direct illumination, i.e., light paths $L \rightarrow P \rightarrow S$. Here, we describe an approximation of the total radiance $L_{P \rightarrow S}$ reaching the sensor S from point P based on the rendering equation (see Figure 1b), which in our notation is

$$L_{P \rightarrow S} = \int_{P' \in \text{scene}} f_{P' \rightarrow P \rightarrow S} \cdot L_{P' \rightarrow P} \cdot \cos \theta_{P \rightarrow P'} \cdot V_{P, P'} \cdot dP', \quad (7)$$

where $f_{P' \rightarrow P \rightarrow S}$ is the BRDF at P for incoming light direction $P' \rightarrow P$ and outgoing light direction $P \rightarrow S$, and $V_{P, P'} = 1$ if P and P' are mutually visible, otherwise 0.

We adapt the ideas of instant radiosity [13] and its implementation via Reflective Shadow Maps (RSMs) [6] for our purposes. First, we separate the direct illumination path $L_{L \rightarrow P \rightarrow S}$ from the single-bounce indirect illumination paths $L_{P' \rightarrow P \rightarrow S}$. Second, we consider a discrete set of points P' in the scene that are directly illuminated by L . This set of virtual point lights (VPLs) is generated by rendering the scene from the point of view of the light source into a two-dimensional map (called reflective shadow map, RSM). Each pixel in this RSM describes one VPL. Note that the RSM

approach ignores multi-bounce indirect illumination paths and realizes single-bounce multipath reflections only.

Considering the separated direct and single-bounce reflections using RSM, the rendering equation then simplifies to

$$L_{P \rightarrow S} = L_{L \rightarrow P \rightarrow S} + \sum_{P' \in \text{RSM}} f_{P' \rightarrow P \rightarrow S} \cdot L_{L \rightarrow P' \rightarrow P} \cdot \cos \theta_{P \rightarrow P'} \cdot V_{P', P} \cdot W_{P', P} \quad (8)$$

with (analogous to the direct illumination path)

$$L_{L \rightarrow P' \rightarrow P} = E_{L \rightarrow P'} \cdot f_{L \rightarrow P' \rightarrow P}, \quad E_{L \rightarrow P'} = I_L(\theta_{L \rightarrow P'}) \frac{\cos \theta_{P' \rightarrow L}}{\|P' - L\|^2} \quad (9)$$

and the VPL weight factor

$$W_{P', P} = A_{P'} \frac{\cos \theta_{P \rightarrow P'}}{\|P' - P\|^2}. \quad (10)$$

$W_{P', P}$ describes the steradian of the VLP P' (considered as area light source) at surface point P . In order to provide all necessary information, each VPL in the RSM stores the irradiance $E_{L \rightarrow P'}$ and the VPL's area $A_{P'}$.

Furthermore, for the image-space RSM approach, visibility tests between P and P' are impractical, thus we simplify this term by suppressing light directions below the horizon, i.e.,

$$V_{P, P'} = \begin{cases} 1, & \text{if } \theta_{P' \rightarrow P} < \frac{\pi}{2} \wedge \theta_{P \rightarrow P'} < \frac{\pi}{2}, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

As the results will show, the limitations of the RSM approach are acceptable in our use case. Its benefits as a purely image-space approach are its simplicity and efficiency, especially for GPU-based implementations.

In contrast to the original RSM approach that samples VLPs only in the vicinity of P , we sample the entire RSM when computing the incident radiance at P .

In the sensor pixel, the charges N_A and N_B now result from a superimposed signal from the direct and multiple single-bounce indirect light paths. We convert radiance incident to the sensor pixel from indirect illumination paths and $L_{L \rightarrow P' \rightarrow P \rightarrow S}$ into irradiance

$$E_{L \rightarrow P' \rightarrow P \rightarrow S} = \cos \theta_{S \rightarrow P} \cdot L_{P \rightarrow S} \quad (12)$$

and can then deduce electron pair counts $N_{\text{tot}, L \rightarrow P' \rightarrow P \rightarrow S}$ for each indirect path using Equations (4) and (5).

Denoting the phase shift along the direct and a single-bounce indirect light path as $\phi_{L \rightarrow P \rightarrow S}$ and $\phi_{L \rightarrow P' \rightarrow P \rightarrow S}$, respectively, we can compute the total charges N_A and N_B by combining Equations (6) and (8), yielding

$$N_A = \frac{N_{\text{tot}, L \rightarrow P \rightarrow S}}{2} (1 + D \cos(\tau + \phi_{L \rightarrow P \rightarrow S})) + \sum_{P' \in \text{RSM}} \frac{N_{\text{tot}, L \rightarrow P' \rightarrow P \rightarrow S}}{2} (1 + D \cos(\tau + \phi_{L \rightarrow P' \rightarrow P \rightarrow S})), \quad (13)$$

$$N_B = \frac{N_{\text{tot}, L \rightarrow P \rightarrow S}}{2} (1 - D \cos(\tau + \phi_{L \rightarrow P \rightarrow S})) + \sum_{P' \in \text{RSM}} \frac{N_{\text{tot}, L \rightarrow P' \rightarrow P \rightarrow S}}{2} (1 - D \cos(\tau + \phi_{L \rightarrow P' \rightarrow P \rightarrow S})). \quad (14)$$

3.4. Noise Model

Electronic and optical shot noise plays the dominant role for ToF cameras [12], which is especially the case for low light situations [14]. In contrast to other noise sources, shot noise cannot be reduced by signal processing methods but has an impact on range resolution $\Delta L = \frac{L}{360^\circ} \cdot \Delta \phi$ with non-ambiguity

range L and phase error $\Delta\phi$ [12]. Shot noise is Poisson distributed, but fitting an explicit Poisson model to it is known to be numerically unstable. A common approach to handle this instability is to use variance stabilization transformations [15]. We apply a Freeman–Tukey (FT) transform [16], which transforms the Poisson distributed charge values into an approximate standard normal distribution. Applying a Gaussian fitting to the FT-transformed mean and variance values yields the distribution model in FT space.

More precisely, we acquire 1000 images of the ToF camera, a PMD CamCube 3.0 in our case, for 80 different integration times. Correcting each raw image by subtracting the dark image (fix pattern noise) yields the raw data that is transformed into FT space. Selecting random pixels in each batch of 1000×4 phase images, mean and variance values are computed (blue dots in Figure 2). In order to deduce the final noise model, we apply a polynomial fitting of degree 9 over the the mean-variance measurements (see Figure 2). The resulting fit error is (sums of squares error) $SSE = 0.0391$ and (root mean square error) $RMSE = 0.0112$. Applying the model is done by transforming the charge values from Equations (13) and (14) into the FT domain, computing the Gaussian parameters by evaluating the variance curve for the given intensity, generating the noise value using a random number and the variance, and, finally, back-transforming this value in the original domain of the charge values. This noise value is then added to the noise-free charge value in order to get the final charge value.

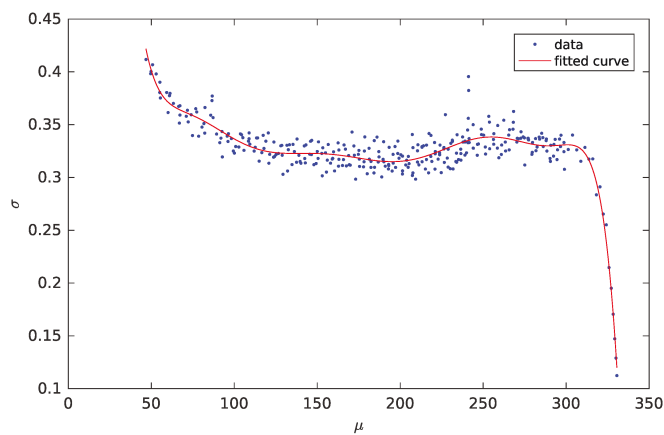


Figure 2. Plot of noise model that describes the Gaussian variance as function of the mean value (transformed intensity) in Freeman–Tukey space with $SSE = 0.0391$ and $RMSE = 0.0112$.

4. NIR BRDF Measurements

In addition to multipath and noise effects in the simulator (Section 3), a quantified comparison between simulated and real AMCW ToF data requires a realistic model of materials in the scene in the form of BRDFs. Analytic BRDF models such as Cook-Torrance [17] offer only limited capabilities to represent real world material. Databases of measured BRDFs, such as MERL [9] and CURET [10], do not provide data for the operating wavelength of AMCW ToF camera, i.e., around 870 nm, and/or do not contain standardized, purchasable materials that can be used to craft real-world reference scenes.

In this section, we briefly describe the measurement and data processing procedures to acquire isotropic BRDF at 850 nm. We apply this procedure to standard material that can be purchased worldwide (see Section 5).

4.1. Measurement Setup

Even though there are various potential approaches to acquire BRDFs [18], similar to Li et al. [19], we opt for a rather simple approach using a three-axis gonioreflectometer. The gonioreflectometer controls the elevation angles of the incoming and outgoing light and their relative azimuth; see Figure 3 left.

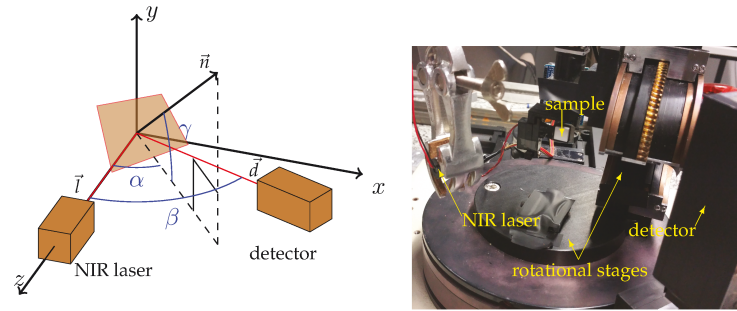


Figure 3. The schematic gonioreflectometer measurement setup (left) and a photo of the real setup (right).

Our measurement setup is parameterized using the angles $\alpha = \angle(\vec{n}', \vec{l})$, $\beta = \angle(\vec{d}, \vec{l})$ and the elevation angle γ of the sample's normal \vec{n} . Here, \vec{n}' is the projection of \vec{n} onto the \vec{l} - \vec{d} -plane, $\vec{l} = (0, 0, 1)^T$ the light direction towards the NIR laser, and \vec{d} the direction towards the detector; see Figure 3, right. Using a simple vector calculus, we get $\vec{n} = (\sin \alpha \cdot \cos \gamma, -\sin \alpha \cdot \sin \gamma, \cos \alpha)^T$ and $\vec{d} = (\sin \beta, 0, \cos \beta)^T$. This simply transfers to the angles θ_i, θ_o between the surface normal and the light and detector direction, respectively, and the azimuthal difference ϕ_d with respect to the sample's coordinates:

$$(\theta_i, \theta_o, \phi_d)^T = (\arccos(\vec{n} \cdot \vec{z}), \arccos(\vec{n} \cdot \vec{d}), \arccos(\vec{z}' \cdot \vec{d}'))^T, \quad (15)$$

where \vec{z}' and \vec{d}' are the projections of \vec{z} and \vec{d} onto the plane perpendicular to \vec{n} .

4.2. Extrapolating BRDF Measurements

Due to physical limitations, the angular regions $\beta \in [-10^\circ, 10^\circ]$ and $\theta_i, \theta_o > 80^\circ$ can not be acquired. Inspired by the work of Panzer and Ponteggia [20], who have a similar problem in sampling directional acoustic reflection values, we tested two approaches: inverse distance weighting (IDW), which is basically a Shepard's method, and spherical harmonic interpolation. In our experiments, we found that an extended version of the first method delivers the most reliable results.

Given the BRDF acquisition values $f(\rho')$ for sampling angular parameters $\vec{\rho}' = (\theta'_i, \theta'_o, \phi'_d)$, the IDW approach computes the BRDF $f(\vec{\rho})$ for an unmeasured parameter vector $\vec{\rho}$ by taking the measurements in its neighborhood $N(\rho)$ into account:

$$f(\vec{\rho}) = \begin{cases} f(\rho'), & \text{if } \rho \rightarrow \rho' \\ \frac{1}{W} \sum_{\vec{\rho}' \in N(\vec{\rho})} w(\vec{\rho}, \vec{\rho}') \cdot f(\vec{\rho}'), & \text{else} \end{cases}, \quad \text{with } W = \sum_{\rho' \in N(\rho)} w(\rho, \rho'). \quad (16)$$

Here, $w(\vec{\rho}, \vec{\rho}')$ is a distance measure, or weight, between the two parameter vectors. In our case, we define w as

$$w(\vec{\rho}_1, \vec{\rho}_2) = (\arccos(\vec{l}_1 \cdot \vec{l}_2) + \arccos(\vec{d}_1 \cdot \vec{d}_2))^{-u} \quad (17)$$

with incoming \vec{l}_1, \vec{l}_2 and outgoing directions \vec{d}_1, \vec{d}_2 corresponding to $\vec{\rho}_1$ and $\vec{\rho}_2$, respectively. This weight definition accounts for angular differences between both the incident and the outgoing angles. Thus, measured BRDF values have a large weight in the BRDF estimation of a set of unobservable angular parameters, if their angular parameters are similar. We use $u = 5$, which has been determined experimentally.

5. Results

In this section, we present the results related to the BRDF measurement for standard materials (Section 5.1) and the evaluation of the multipath interference using the proposed simulation technique (Section 5.2).

5.1. BRDF Measurement

The measurement setup has been equipped with a 850 nm NIR-laser as a light source (LDM850/5LT, Roithner Lasertechnik GmbH, Vienna, Austria) and a photo diode as detector. The chosen sampling stepsizes are $\Delta\alpha = \Delta\beta = \Delta\gamma = 2^\circ$ for specular material and $\Delta\alpha = \Delta\beta = \Delta\gamma = 5^\circ$ for diffuse materials. To reduce noise, we averaged 1000 measurements for each parameter set $\vec{\rho} = (\alpha, \beta, \gamma)$. Finally, we normalized the measured reflection values by the maximum energy arriving directly from laser to detector and the cosine of the incident angle θ_i in order to get the final BRDF values.

We have selected materials with rather diffuse and rather glossy reflection properties. Since we consider reflections but not scattering, transmission, and other properties, we have chosen opaque materials so that almost no light transmission occurs. As diffuse materials, we have chosen guttagliss PVC rigid foam variants [21], as materials with glossy specular reflection we have chosen PLEXIGLAS® [22]; see Table 1.

Table 1. Materials used for BRDF measurement. The PLEXIGLAS® provides more specular reflection, and the PVC rigid foam is rather diffuse.

Mat. No.	PLEXIGLAS® (Glossy)	Mat. No.	Guttagliss PVC (Diffuse)
1	XT (allround), White WN297 GT	5	Rigid Foam, White
2	XT (allround), Red 3N570 GT	6	Rigid Foam, Red
3	XT (allround), Green 6N570 GT	7	Rigid Foam, Green
4	XT (allround), Blue 5N870 GT	8	Rigid Foam, Blue
		9	Rigid Foam, Yellow
		10	Rigid Foam, Gray

Figure 4 shows the polar plots of the raw measured and the resulting interpolated BRDFs; see Section 4. Plots of the raw measurement visualize the missing measurements around the incoming direction for $\beta \in [-10^\circ, 10^\circ]$. The IDW interpolation closes the gap and slightly smooths the measurements.

Open Science

Upon acceptance of this paper, we will make the full BRDF data publicly available, thus other researchers can purchase the respective materials in order to setup their own test scenes for quantitative evaluations.

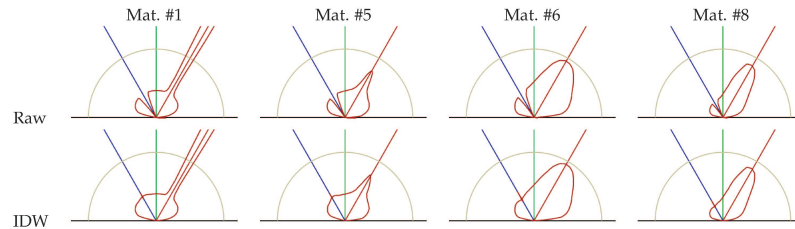


Figure 4. BRDF raw data and IDW interpolation results for materials #1, #5, #6 and #8 acquired for an incident light angle $\theta_i = 30^\circ$. The blue ray indicates the incident light direction, the red ray the ideally reflected incident light direction and the red curve the BRDF value related to the corresponding ray from the center to a point of this curve.

5.2. Simulator Evaluation

In our evaluation of the simulator, we have used three different scene geometries. All geometries are variants of a corner, for which multipath effects are to be expected (see Figure 5a): **Corner** is a simple corner scene without an additional cube, in the **CornerCube** scene an additional cube is placed directly in the corner, and in the **CornerCubeShift** the cube is shifted by 10 cm from each corner wall. We have setup the three scenes with **Material #1** and **Material #5**.

For real world data acquisition, we have used a PMD CamCube 3.0 ToF camera that captures depth images at 200×200 px resolution. The CamCube's driver delivers data that is already corrected for the so-called wiggling error, which is a systematic error occurring due to imperfections in the signal shape of the real-world camera with respect to the theoretically assumed cosine function; see Equation (6). Its active light source operates at 870 nm wavelength. The ToF camera is positioned symmetrically towards the corner. Even though there are sophisticated approaches that utilize range and intensity data to determine intrinsic and extrinsic ToF camera parameters [23], we use OpenCV's implementation Zhang's simple checkerboard method [24] to estimate these parameters (see Figure 5c). Lindner [25] reports a distance error of less than 0.9–1.9 cm at 1.2–2.2 m using this method (see Table 2.2 in [25]). The range measurements of the PMD CamCube are denoted as **CamCube**, while the simulation results are labeled as **SimDirect** (only direct reflection is simulated) and **SimSingle** (additional single-bounce indirect reflections are simulated). Furthermore, we add the ground truth depth information for comparison (**GroundTruth**). Our evaluation indirectly compares to Lambers et al. [2], as **SimDirect** essentially is the approach in Lambers et al. [2] enhanced with the noise model described in Section 3.4 and BRDF-based reflection (instead of Lambertian).

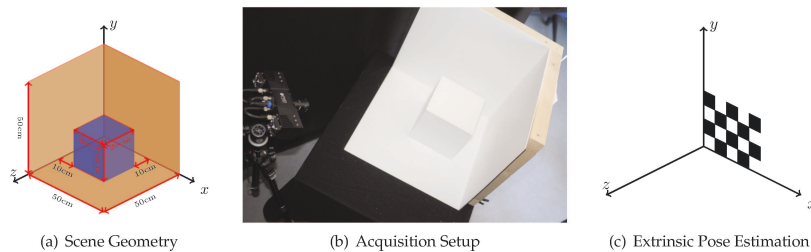


Figure 5. The geometry of our box scene (light brown) with the additional cube (blue) (a); a photo of the AMCW ToF measurement setup (b); and the positioning of the calibration pattern for ToF camera pose estimation (c).

Figure 6 shows the depth images for all three test-scenes for **GroundTruth**, the **CamCube** measurements, and the **SimDirect** and **SimSingle** simulations. Figures 7 and 8 give additional insight into range simulation results by showing the signed differences between the simulation and the **CamCube** measurements for the scenes **CornerCube**, **CornerCubeShift** and the explicit range values along row 100 for all three scenes, respectively. As expected, the real ToF data exhibits significant multipath effects in all three scenes. Considering the simulation without multipath component (**SimDirect**), the resulting range values are close to the ground truth depth. This is consistent with the ToF measurement principle, which explicitly considers direct reflection only. Table 2 states all error values for all scenes and material with respect to the measured **CamCube** data. Especially for scenes **Corner** and **CornerCube**, our approach outperforms **SimDirect** since it captures multipath effects in the corners. In **CornerCubeShifted**, our approach still decreases the errors by more than 50%. In summary, we find that adding single-bounce indirect reflections (**SimSingle**) significantly improves the simulation results with respect to the **CamCube** measurements. This is especially the case for the **Corner** and the **CornerCube** scenes. For the **CornerCubeShift** scene, however, the deviation between the ToF measurement **CamCube** and the simulation including single-bounce reflections **SimSingle** still deviate, mainly in the visual corners between the base corner and the inserted, shifted cube.

Table 2. Evaluation of error for all simulation methods and scenes with respect to the measured **CamCube** data. For each method you can see the the mean-absolute-error (MAE), mean-squared-error, (MSE) and the root-mean-squared-error (RSME).

	Corner		CornerCube		CornerCubeShifted	
	Material #1	Material #5	Material #1	Material #5	Material #1	Material #5
GroundTruth						
MAE	0.1001	0.0998	0.0831	0.0790	0.0969	0.0897
MSE	0.0103	0.0105	0.0071	0.0068	0.0106	0.0091
RMSE	0.1017	0.1025	0.0853	0.0823	0.1029	0.0956
SimDirect						
MAE	0.0885	0.0884	0.0728	0.0688	0.0846	0.0777
MSE	0.0084	0.0085	0.0079	0.0069	0.0086	0.0074
RMSE	0.0916	0.0924	0.0888	0.0829	0.0927	0.0859
SimSingle						
MAE	0.0238	0.0200	0.0194	0.0138	0.0396	0.0342
MSE	0.0007	0.0005	0.0006	0.0003	0.0023	0.0017
RMSE	0.0270	0.0226	0.0233	0.0171	0.0476	0.0417

Figure 9 shows the limitations of our simulation method. We have used a variant of the **CornerCube** scene, where we have placed an aluminum cube with an edge length of 5 cm into the glossy corner (**Material #1**). In our simulation, we have used a Cook-Torrance BRDF to model the reflection behavior of the aluminum cube. In this highly reflective scenario, the multipath effects have a strong influence on the **CamCube** measurements (see range values in row 108, Figure 9 right). Here, the cube nearly vanishes in the distance measurements between pixels 90 and 110. **SimSingle**, in comparison, cannot capture this camera behavior, as no higher order multipath effects are simulated.

Regarding the noise model, we observe that the noise level of the **SimSingle** simulations and the real ToF measurements **CamCube** are comparable, whereas the noise level for the direct simulation **SimDirect** is higher. This is due to the fact that the total amount of charge is lower and that, in this case, the additive Poisson noise is with a fixed amplitude. Thus, the relative impact of the noise is higher in the case of the lower overall charge in the **SimDirect** simulation. This effect gets very apparent for flat incident angle with respect to the direct light–surface impact.

Some final notes on the performance of the simulator: The computation of RSMs takes approx. 1.5 ms. Since we sample the whole RSM, the final accumulation step takes approx. 80 ms. Thus,

our simulation operates at interactive frame rates. Meister et al. [4], the only other AMCW simulation method accounting for multipath effects, requires computation times in the order of hours.

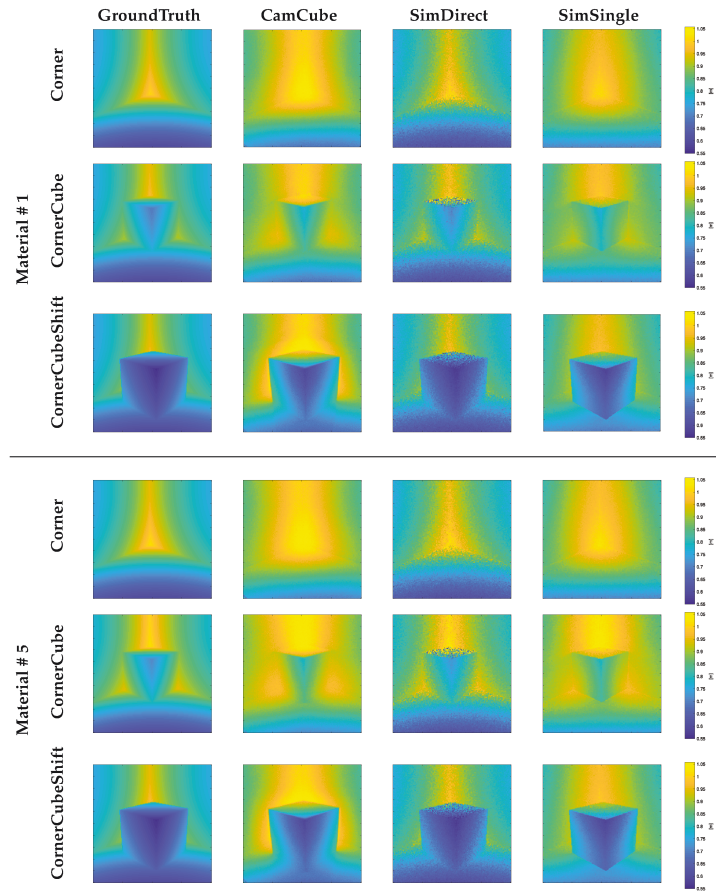


Figure 6. Range image comparison: **GroundTruth** (left) and **CamCube** (mid-left) compared with simulation using direct illumination (**SimDirect**, mid-right) and single bounce reflection (**SimSingle**, right) for the three test scenes **Corner** (rows 1,4), **CornerCube** (rows 2,5) and **CornerCubeShift** (rows 3,6) for **Material # 1** and **Material # 5**.

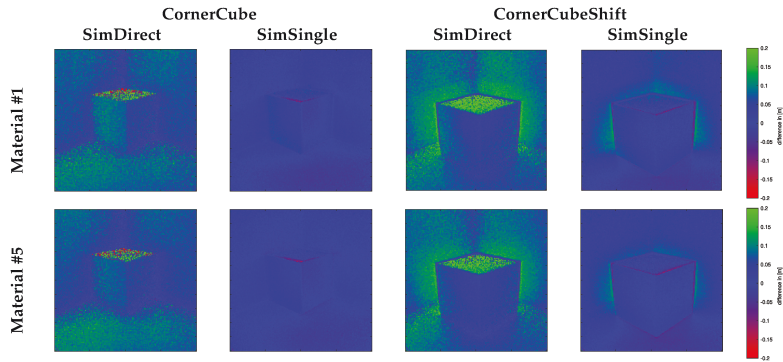


Figure 7. Signed difference images to the CamCube measurement for CornerCube and CornerCubeShift: GroundTruth (Left) and the simulation with direct illumination (SimDirect, middle) and with single bounce reflection (SimSingle, Right) for Material #1 (top row) and Material #5 (bottom).

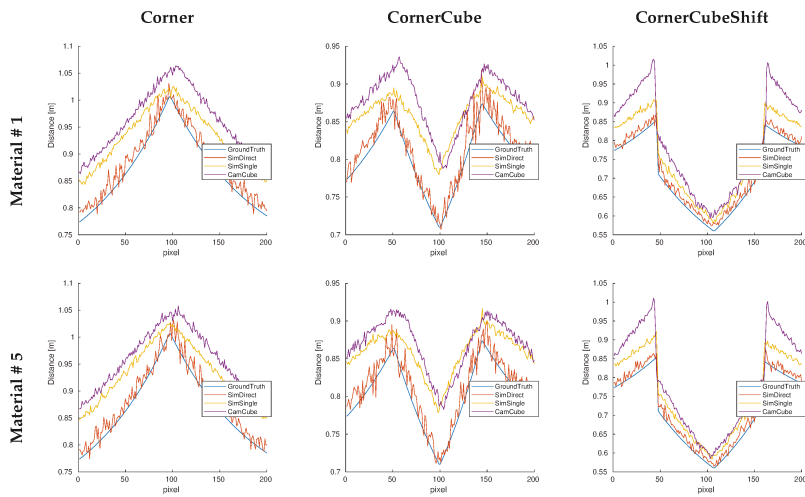


Figure 8. Range comparison for scan lines 100 for the three test scenes Corner (Left), CornerCube (Middle) and CornerCubeShift (Right) for Material #1 and Material #5.

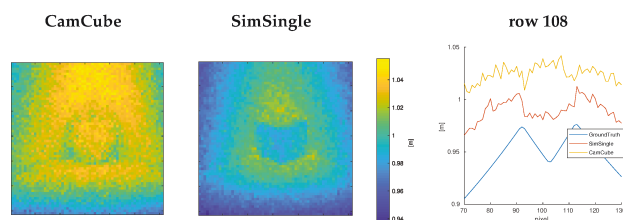


Figure 9. Evaluation of the corner scene with an aluminum cube (extraction of pixel regions $[70, 130] \times [70, 130]$): this scene comprises a significantly larger amount of multipath effects that cannot be fully covered by our single-bounce simulation method.

6. Conclusions

We present an enhancement of a physically-based simulation technique for AMCW ToF cameras with respect to multi-path effects and shot noise, while maintaining interactive frame rates in a GPU-based implementation. We further provide a database of BRDF measurements in the near-infrared range for a selection of purchasable materials. This database enables researchers to build up their own real-world and virtual reference scenes out of materials with known reflection properties. This allows for quantitative comparison between corresponding real-world and virtual scenes and, thus, allows for quantitative evaluation of ToF cameras.

The comparison of simulated and measured depth data in Section 5 demonstrates that the combination of BRDF measurements, multi-path effects, and noise enables realistic simulation results that closely match real measurements of reference scenes, even though our model contains single-bounce indirect reflections only.

In future work, it would be interesting to adapt our simulation technique to other types of sensors, e.g., pulse-based ToF cameras, which are also affected by multi-path effects. These cameras operate at similar wavelengths so that our BRDF database again facilitates quantitative comparisons of simulated and real data.

Acknowledgments: The work is partially funded by the German Research Foundation (DFG), grants Ko-2960-12/1 and GRK-1564/2.

Author Contributions: David Bulczak, Martin Lambers, Andreas Kolb conceived and designed the experiments; David Bulczak performed the experiments; analyzed the data and contributed reagents/materials/analysis tools; David Bulczak, Martin Lambers, Andreas Kolb wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kolb, A.; Barth, E.; Koch, R.; Larsen, R. Time-of-Flight cameras in computer graphics. *Comput. Graph. Forum* **2010**, *29*, 141–159.
2. Lambers, M.; Hoberg, S.; Kolb, A. Simulation of Time-of-Flight sensors for evaluation of chip layout variants. *IEEE Sens.* **2015**, *15*, 4019–4026.
3. Nair, R.; Meister, S.; Lambers, M.; Balda, M.; Hofmann, H.; Kolb, A.; Kondermann, D.; Jähne, B. Ground truth for evaluating time of flight imaging. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 52–74.
4. Meister, S.; Nair, R.; Kondermann, D. Simulation of Time-of-Flight sensors using global illumination. In *Vision, Modeling & Visualization*; The Eurographics Association: Geneva, Switzerland, 2013.
5. Keller, M.; Kolb, A. Real-time simulation of time-of-flight sensors. *Simul. Model. Pract. Theory* **2009**, *17*, 967–978.
6. Dachsbacher, C.; Stamminger, M. Reflective shadow maps. In Proceedings of the Symposium on Interactive 3D Graphics and Games, Washington, DC, USA, 3–6 April 2005; pp. 203–231.

7. Schmidt, M.; Jähne, B. A physical model of time-of-flight 3D imaging systems, including suppression of ambient light. In *Dynamic 3D Imaging*; Springer: Berlin, Germany, 2009; pp. 1–15.
8. Ritschel, T.; Dachsbacher, C.; Grosch, T.; Kautz, J. The state of the art in interactive global illumination. *Comput. Graph. Forum* **2012**, *31*, 160–188.
9. Matusik, W.; Pfister, H.; Brand, M.; McMillan, L. A Data-Driven Reflectance Model. *ACM Trans. Graph.* **2003**, *22*, 759–769.
10. Choe, G.; Narasimhan, S.G.; So Kweon, I. Simultaneous estimation of near IR BRDF and fine-scale surface geometry. In Proceedings of the IEEE 2016 IEEE Conference on Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2452–2460.
11. Mutny, M.; Nair, R.; Gottfried, J.M. Learning the correction for Multi-Path deviations in Time-of-Flight Cameras. *arXiv* **2015**, arXiv:1512.04077.
12. Lange, R.; Seitz, P. Solid-state time-of-flight range camera. *IEEE J. Quantum Electron.* **2001**, *37*, 390–397.
13. Keller, A. Instant radiosity. In Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), Los Angeles, CA, USA, 3–8 August 1997; pp. 49–56.
14. Conde, M.H.; Zhang, B.; Kagawa, K.; Loffeld, O. Low-light image enhancement for multiaperture and multitap systems. *IEEE Photonics J.* **2016**, *8*, 1–25.
15. White, G.C.; Bennetts, R.E. Analysis of frequency count data using the negative binomial distribution. *Ecology* **1996**, *77*, 2549–2557.
16. Freeman, M.F.; Tukey, J.W. Transformations related to the angular and the square root. *Ann. Math. Stat.* **1950**, *21*, 607–611.
17. Schlick, C. An Inexpensive BRDF Model for Physically-based Rendering. *Comput. Graph. Forum* **1994**, *13*, 233–246.
18. Achutha, S. BRDF Acquisition with Basis Illumination; Chapter 2.2 “BRDF Acquisition”. Ph.D. Thesis, The University of British Columbia, Vancouver, BC, Canada, 2006.
19. Li, H.; Foo, S.C.; Torrance, K.E.; Westin, S.H. Automated three-axis gonioreflectometer for computer graphics applications. *Opt. Eng.* **2006**, *45*, 043605, doi:10.1117/1.2192787.
20. Panzer, J.; Ponteggia, D. *Inverse Distance Weighting for Extrapolating Balloon-Directivity-Plots*; AES Convention 13; Audio Engineering Society: New York, NY, USA, 2011.
21. All-Foam Products Company. Available online: <http://products.allfoam.com/> (accessed on 20 December 2017).
22. Evonik Cyro LCC. Available online: <http://www.plexiglas-shop.com/> (accessed on 20 December 2017).
23. Lindner, M.; Schiller, I.; Kolb, A.; Koch, R. Time-of-Flight Sensor Calibration for Accurate Range Sensing. *Comput. Vis. Image Underst.* **2010**, *114*, 1318–1328.
24. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)* **2000**, *22*, 1330–1334.
25. Lindner, M. Calibration and Realtime Processing of Time-of-Flight Range Data. Ph.D. Thesis, University of Siegen, Computer Graphics Group, Siegen, Germany, 2010.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

Mappings between Sphere, Disc, and Square

Martin Lambers
University of Siegen, Germany

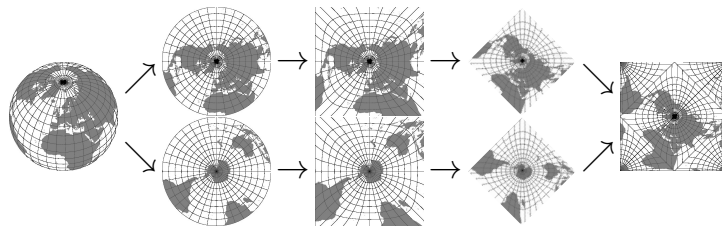


Figure 1. A mapping between a sphere and a square, composed of a mapping between a hemisphere and a disc, a mapping between a disc and a square, and an arrangement of two squares in a new square.

Abstract

A variety of mappings between a sphere and a disc and between a disc and a square, as well as combinations of both, are used in computer graphics applications, resulting in mappings between spheres and squares. Many options exist for each type of mapping; to pick the right methods for a given application requires knowledge about the nature and magnitude of mapping distortions.

This paper provides an overview of forward and inverse mappings between a unit sphere, a unit disc, and a unit square. Quality measurements relevant for computer graphics applications are derived from tools used in the field of map projection, and a comparative analysis of the mapping methods is given.

1. Introduction and Background

Mappings between spheres, discs, and squares are useful tools in many areas of computer graphics. Examples include panoramic imaging [German et al. 2007; Fong 2014], environment mapping [Greene 1986; Heidrich and Seidel 1998], and generating points on a disc or sphere for sampling purposes [Shirley and Chiu 1997; Sloan et al. 2005].

Important properties of such mappings include the angle and area distortions that they introduce [Floater and Hormann 2005]. Some applications require equal-area mappings that preserve area ratios, and others require conformal mappings that preserve angles locally. No mapping can be both equal-area and conformal at the same time, and preserving one quality often results in strong distortions in the other. Many applications, in particular in computer graphics, require mappings that allow efficient sampling of maps [Snyder and Mitchell 2001] without introducing artifacts. A balanced mapping for this purpose provides both small area distortions and small angle distortions, although neither distortion has to be zero.

This paper gives an overview of mappings between a unit sphere, a unit disc, and a unit square, with formulas for forward and inverse transformation. Furthermore, we derive two quality measurements relevant for computer graphics applications based on established tools from the field of map projection, and we compare the mappings based on these measurements.

Section 2 starts with mappings between discs and squares, Section 3 describes map projections between spheres (or hemispheres) and discs, and Section 4 details methods to combine methods from both categories to produce mappings between spheres and squares. Section 5 provides numerical analysis results for all mappings based on the derived quality measurements. The supplementary material consists of C++ source code that implements all mapping and analysis methods as well as all tools necessary to recreate the figures and results presented in this paper.

2. Mappings between Disc and Square

This section provides an overview of mappings between the closed unit disc D and the closed unit square R . We identify points on D using either polar coordinates, with radius r and angle φ , $0 \leq r \leq 1$, $-\pi < \varphi \leq \pi$, or Cartesian coordinates, u and v with $r = \sqrt{u^2 + v^2}$ and $\varphi = \text{atan2}(v, u)$. Points on R are identified using Cartesian coordinates x and y , $-1 \leq x \leq +1$, $-1 \leq y \leq +1$. The mappings are summarized and compared in Table 1.

2.1. Radial Stretching

Perhaps, the most direct method of mapping a disc to a square is to adjust the radius of a point on the disc according to its angle.

For $(x, y) \in R$, we have a distance of $t = \sqrt{x^2 + y^2}$ to the origin and a polar angle of $\varphi = \text{atan2}(y, x)$. The polar radius of $(r, \varphi) \in D$ can then be set to $r = t \cos \varphi$ for $\varphi \in [-\pi/4, +\pi/4]$ (and accordingly for the remaining ranges of φ). Using $\cos \varphi = x/t$, $\sin \varphi = y/t$, $\tan \varphi = v/u$, and a method applied by Cline to Shirley's equal-area mapping [Shirley and Cline 2011] to reduce the number of cases, this leads to the following simple equations (see the derivation by Fong for details [Fong 2015]):

	Stretching	Shirley	Squirele	Elliptical	Conformal
	D_A				
					D_I

Table 1. Overview of mappings between a disc and a square.

Disc to square mapping:

$$r = \sqrt{u^2 + v^2}$$

$$(x, y) = \begin{cases} (0, 0) & \text{if } r = 0 \\ (\text{sgn}(u) \cdot r, \text{sgn}(v) \cdot rv/u) & \text{if } r > 0 \text{ and } u^2 \geq v^2 \\ (\text{sgn}(u) \cdot ru/v, \text{sgn}(v) \cdot r) & \text{if } r > 0 \text{ and } u^2 < v^2 \end{cases}$$

Square to disc mapping:

$$t = \sqrt{x^2 + y^2}$$

$$(u, v) = \begin{cases} (0, 0) & \text{if } t = 0 \\ (\text{sgn}(x) \cdot x^2/t, \text{sgn}(y) \cdot xy/t) & \text{if } t > 0 \text{ and } x^2 \geq y^2 \\ (\text{sgn}(x) \cdot xy/t, \text{sgn}(y) \cdot y^2/t) & \text{if } t > 0 \text{ and } x^2 < y^2 \end{cases}$$

This mapping is neither conformal nor equal-area. It suffers from strong angular and area distortions.

2.2. Shirley's Equal-Area Mapping

Shirley constructs an equal-area map between a disc and a square by mapping concentric disc strings to concentric square strips [Shirley and Chiu 1997]. Note that Roşca derived an equivalent mapping (except for a scale factor) using a different approach [Roşca 2010]. Both prove the equal-area property by showing that the Jacobian is constant. The following formulas for the disc-to-square mapping are based on Shirley's code samples. The formulas for the inverse mapping are based on Cline's method for reducing the number of cases [Shirley and Cline 2011].

Disc to square mapping:

$$r = \sqrt{u^2 + v^2}$$

$$\varphi = \begin{cases} \text{atan2}(v, u) & \text{if } \text{atan2}(v, u) \geq -\pi/4 \\ \text{atan2}(v, u) + 2\pi, & \text{otherwise} \end{cases}$$

$$(x, y) = \begin{cases} (r, \frac{4}{\pi}r\varphi) & \text{if } \varphi < \pi/4 \\ (-\frac{4}{\pi}r(\varphi - \pi/2), r) & \text{if } \varphi < 3\pi/4 \\ (-r, -\frac{4}{\pi}r(\varphi - \pi)) & \text{if } \varphi < 5\pi/4 \\ (\frac{4}{\pi}r(\varphi - 3\pi/2), -r), & \text{otherwise} \end{cases}$$

Square to disc mapping:

$$(r, \varphi) = \begin{cases} (x, \frac{\pi}{4}y/x) & \text{if } x^2 > y^2 \\ (y, \frac{\pi}{2} - \frac{\pi}{4}x/y) & \text{if } x^2 \leq y^2 \text{ and } y^2 > 0 \\ (0, 0), & \text{otherwise} \end{cases}$$

Note that by writing the square to disc mapping of the radial stretching method in terms of r, φ , using $\cos \varphi = x/t$ and $\sin \varphi = y/t$, one can see that Shirley's mapping uses the same stretching of the radius but modifies the angle, φ .

This mapping is equal-area. It suffers from strong angular distortions, especially around the discontinuities at the diagonals of the square.

2.3. Fernández-Guasti's Squircle Mapping

Fernández-Guasti introduced a geometric form that can be varied between square and circle using a "squareness" parameter [Fernández-Guasti 1992]; he later called this form squiracle. Note that the term squiracle is sometimes used to refer to a special case of a superellipse. To avoid ambiguities, we therefore use the term "Fernández-Guasti's squiracle".

In terms of Cartesian coordinates, the equation that describes Fernández-Guasti's squiracle is $\frac{s^2}{k^4}x^2y^2 - \frac{x^2+y^2}{k^2} + 1 = 0$, where s is the squareness parameter and k is the radius of the circle (for $s = 0$) or half the side length of the square (for $s = 1$). For values of s between 0 and 1, the geometry resembles both square and circle.

Fong proposes to map concentric circles to concentric squiracles to construct a mapping between a unit disc and a unit square [Fong 2014]. For this purpose, he sets $s = k$ and varies s from 0 to 1, taking $r = s = \sqrt{x^2 + y^2 - x^2y^2}$ as disc radius.

Disc to square mapping:

$$w = \frac{\text{sgn}(uv)}{\sqrt{2}} \sqrt{u^2 + v^2 - \sqrt{(u^2 + v^2)(u^2 + v^2 - 4u^2v^2)}}$$

$$(x, y) = \begin{cases} (w/v, w/u) & \text{if } |w| > 0 \\ (u, v), & \text{otherwise} \end{cases}$$

Square to disc mapping:

$$u = x \frac{\sqrt{x^2 + y^2 - x^2y^2}}{\sqrt{x^2 + y^2}}$$

$$v = y \frac{\sqrt{x^2 + y^2 - x^2y^2}}{\sqrt{x^2 + y^2}}$$

Note that $(0, 0)$ must be mapped to $(0, 0)$ as a special case.

This mapping is neither equal-area nor conformal.

2.4. Elliptical Arc Mapping

Nowell derived a method to map a square to a disc by mapping lines of constant x and lines of constant y in the square to ellipses in the disc [Nowell 2005]. Cigolle et al. mention an elliptical mapping between a disc and a square [Cigolle et al. 2014]. Their paper omits details, but their implementation in the supplementary material

shows that they derived equivalent formulas for the mapping from square to disc and additional formulas for the inverse mapping. The simpler formulas for the inverse mapping given below were derived by Fong [Fong 2015].

Disc to square mapping:

$$x = \frac{1}{2}\sqrt{2 + u^2 - v^2 + 2\sqrt{2}u} - \frac{1}{2}\sqrt{2 + u^2 - v^2 - 2\sqrt{2}u}$$
$$y = \frac{1}{2}\sqrt{2 - u^2 + v^2 + 2\sqrt{2}v} - \frac{1}{2}\sqrt{2 - u^2 + v^2 - 2\sqrt{2}v}$$

Square to disc mapping:

$$u = x\sqrt{1 - \frac{y^2}{2}}$$
$$v = y\sqrt{1 - \frac{x^2}{2}}$$

This mapping is neither equal-area nor conformal.

2.5. Conformal Mapping

In complex analysis, the Schwarz-Christoffel transformation provides a way to construct conformal transformations between simple polygons and the upper half of the complex plane. Since there exists a conformal transformation between the upper half of the complex plane and the open unit disc, one can construct a conformal mapping between a square and a disc. Indeed, this special case was used as illustration and motivation in Schwarz' original publication on the topic. However, formulas for forward and inverse mappings suitable for implementation on computers were not available for some time.

Conformal mapping of a disc onto a rotated square using the Schwarz-Christoffel transformation is a core element of the Peirce quincuncial map projection [Peirce 1879]; see also Section 4. The formulas given below are based on Lee's analysis of Peirce's projection [Lee 1976].

The mapping from disc to square first rotates the disc by 45° . This step is not necessary, but it makes this mapping consistent with the others with regard to the orientation of the square content. Then, the rotated disc is conformally mapped to a rotated square with corners $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$. This square is then rotated and scaled to fit the unit square. The conformal mapping is based on the incomplete elliptic Legendre integral, F , with a modulus of $k = 1/\sqrt{2}$. This integral must be computed using iterative methods. Numerical libraries usually provide the necessary methods; our supplementary material includes an implementation based on the Landen transformation.

The mapping from square to disc first rotates and scales the unit square back into the square with corners $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$. It then uses the complex-valued Jacobian elliptic function cn with modulus $k = 1/\sqrt{2}$ to conformally map

that square to the unit circle. Afterwards, the inverse of the optional rotation is applied. This mapping requires the value $K = F(\pi/2) \approx 1.854$, which is the complete elliptic integral of the first kind with modulus $k = 1/\sqrt{2}$. The function cn must again be computed using iterative methods. Numerical libraries often provide real-valued implementations that compute the related functions cn , sn , and dn at the same time; our supplementary material includes an implementation based on the arithmetic-geometric mean. For the special case of $k = 1/\sqrt{2}$, the complementary elliptic modulus $k' = \sqrt{1 - k^2}$ is identical to k , and the complex cn function can be computed from real-valued cn , sn , and dn as follows:

$$\text{cn}(x + iy) = \frac{\text{cn}(x) \text{cn}(y)}{1 - \text{dn}^2(x) \text{sn}^2(y)} - i \frac{\text{sn}(x) \text{dn}(x) \text{sn}(y) \text{dn}(y)}{1 - \text{dn}^2(x) \text{sn}^2(y)}$$

Note that Stark derives optimized numerical methods for the special case of conformal mappings between unit sphere and unit disc [Stark 2009].

Disc to square mapping:

$$\begin{aligned} u' &= (u - v)/\sqrt{2} \\ v' &= (u + v)/\sqrt{2} \\ A &= u'^2 + v'^2 \\ B &= u'^2 - v'^2 \\ T &= \sqrt{(1 + A^2)^2 - 4B^2} \\ U &= 1 + 2B - A^2 \\ \alpha &= \text{acos}((2A - T)/U) \\ \beta &= \text{acos}(U/(2A + T)) \\ x' &= \text{sgn}(u')(1 - F(\alpha)/2K) \\ y' &= \text{sgn}(v')(F(\beta)/2K) \\ x &= x' + y' \\ y &= y' - x' \end{aligned}$$

Square to disc mapping:

$$\begin{aligned} x' &= x/2 - y/2 \\ y' &= x/2 + y/2 \\ w &= \text{cn}(K(1 - x') - iKy') \\ u &= (\Re(w) + \Im(w))/\sqrt{2} \\ v &= (\Im(w) - \Re(w))/\sqrt{2} \end{aligned}$$

This mapping is conformal, except for the four singular points located on the corners of the square. Area deformation is substantial, especially near the singular points.

3. Mappings between Sphere and Disc

This section covers mappings between the unit sphere, S , and the unit disc, D . We identify points on S using the longitude λ and the colatitude θ , $-\pi < \lambda \leq \pi$, $0 \leq \theta \leq \pi$. The colatitude measures the angle to the north pole (by convention $(0, 0, 1)$ in Cartesian coordinates). We use colatitude instead of latitude (which measures the angle to the equatorial plane), because the resulting formulas are shorter. The longitude measures the angle in the xy -plane. Points on D are again identified using polar coordinates with radius r and angle φ , $0 \leq r \leq 1$, $-\pi < \varphi \leq \pi$.

The problem of mapping the sphere onto the disc is one of the classical areas of map projection, and the methods discussed in this section are well known in that field. In the following sections, all methods map the north pole to the center of the unit disc, set the polar angle $\varphi = \lambda - \pi/2$, and compute radius r as a function of the colatitude θ . We will discuss only this radius function for each method.

Only a subset of the mappings covered here can map the whole sphere to the unit disc; others are restricted to a hemisphere. By shifting the north pole of the sphere to another point and suitably recomputing longitude and colatitude, other projection centers can be chosen. In Section 4, we will use the south pole as an additional projection center to cover both north and south hemispheres. For this case, the new colatitude is simply $\pi - \theta$, and the longitude remains unchanged.

The mappings are summarized and compared in Table 2.

3.1. Equal-Area Projection (Lambert Azimuthal)

Lambert designed several important map projections [Snyder 1987], among them the azimuthal equal-area map projection.

Sphere to disc mapping:

$$r = \sin(\theta/2)$$

Disc to sphere mapping:

$$\theta = 2 \arcsin(r)$$

When projecting just one hemisphere instead of the whole sphere, an additional scale factor of $\sqrt{2}$ is applied to the radius so that the unit disc is filled.

This mapping can map the whole sphere in a disc. It is an equal-area mapping. Angular distortions increase with distance to the pole.

3.2. Conformal Projection (Stereographic)

The stereographic map projection is a conformal projection that was already known in ancient Greece [Snyder 1987].

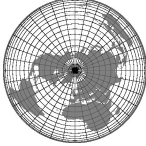
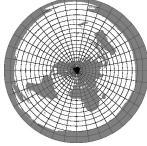
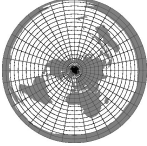
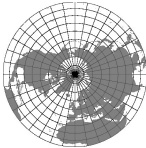
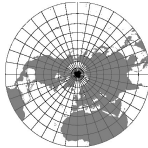
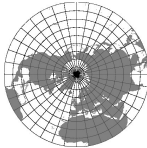
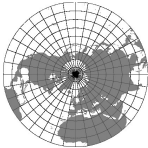
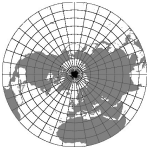

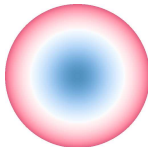
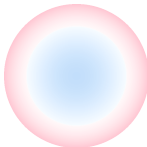
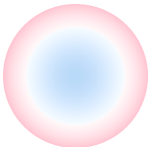
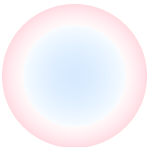


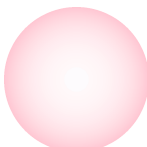
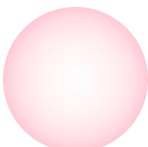
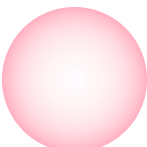
	Equal-Area	Conformal	Harmonic Mean	Mixture	Equidistant
Sphere to disc					
Hemisphere to disc					
D_A (hemisphere)					
D_I (hemisphere)					

Table 2. Overview of mappings between a sphere and a disc.

Hemisphere to disc mapping:

$$r = \tan(\theta/2)$$

Disc to hemisphere mapping:

$$\theta = 2 \operatorname{atan}(r)$$

This mapping cannot be used to map a complete sphere in a disc. It is a conformal mapping. Scale increases with distance from the pole.

3.3. Harmonic Mean of Equal-Area and Conformal Projection (Breusing)

Many attempts have been made to balance the area-preserving qualities of the Lambert azimuthal equal-area projection with the angle-preserving qualities of the stereographic projection, with the goal of arriving at a map projection with only moderate area and angle distortions throughout. Breusing is credited with the idea of using the geometric mean of both projections; Young preferred the harmonic mean over the geometric and arithmetic means and stated that it leads to simpler formulas than the alternatives while being an error-minimizing projection in some sense [Young 1920].

Sphere to disc mapping:

$$r = \tan(\theta/4)$$

Disc to sphere mapping:

$$\theta = 4 \operatorname{atan}(r)$$

When projecting just one hemisphere instead of the whole sphere, an additional scale factor of $\frac{1}{\sqrt{2}-1}$ is applied to the radius so that the unit disc is filled.

This mapping can map the whole sphere in a disc. It is neither equal-area nor conformal, but both area and angular distortions are moderate.

3.4. Mixture of Equal-Area and Conformal Projection

Instead of using a fixed relation between equal-area and conformal projection, Fong proposes to use a parameterized mixture of both [Fong 2014]. The tradeoff between area and angular distortions can be chosen using a parameter $\beta \in [0, 1]$, with $\beta = 0$ choosing the stereographic projection and $\beta = 1$ choosing the Lambert azimuthal equal-area projection. Fong applied this idea to the whole sphere, which lead to difficulties since the stereographic projection requires an infinite plane to map the complete sphere; in his version, the parameter β can only approach zero, but not become zero. We restrict the mixed projection to the hemisphere instead and can, therefore, use simpler formulas that furthermore do not impose a restriction on β .

Hemisphere to disc mapping:

$$r = \frac{\sqrt{1 + \beta} \tan(\theta/2)}{\sqrt{1 + \beta \tan^2(\theta/2)}}$$

Disc to hemisphere mapping:

$$\theta = 2 \operatorname{atan} \left(\frac{r}{\sqrt{1 + \beta(1 - r^2)}} \right)$$

For $\beta = 0$, $r = \tan(\theta/2)$ and $\theta = 2 \operatorname{atan}(r)$, thus this mapping becomes the stereographic mapping. For $\beta = 1$, $r = \frac{\sqrt{2} \tan(\theta/2)}{\sqrt{1 + \tan^2(\theta/2)}} = \sqrt{2} \sin(\theta/2)$ and $\theta = 2 \operatorname{atan}(r/\sqrt{2 - r^2}) = 2 \operatorname{asin}(r/\sqrt{2})$, thus this mapping becomes the Lambert azimuthal equal-area mapping in its hemisphere variant.

This mapping cannot map the whole sphere in a disc. It can be equal-area (for $\beta = 1$) or conformal (for $\beta = 0$) and balances area against angular distortions for $0 < \beta < 1$.

3.5. Equidistant Projection

The equidistant map projection has been known for many centuries; its origin cannot be clearly identified [Snyder 1987]. Its main features are its simplicity and the preservation of distances measured from the center of the projection. Its area and angular distortions fall between those of the equal-area and conformal projections, thus making this projection another candidate for a compromise between both.

Sphere to disc mapping:

$$r = \theta/\pi$$

Disc to sphere mapping:

$$\theta = r\pi$$

When projecting just one hemisphere, instead of the whole sphere, an additional scale factor of 2 is applied to the radius so that the unit disc is filled.

This projection can map the whole sphere in a disc. It is neither conformal nor equal-area, but balances area and angular distortions. Distances to the center are preserved.

4. Mappings between Sphere and Square

For projecting a sphere onto a square, there are essentially two layout options, described in the following sections.

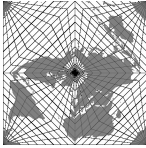
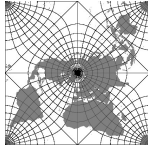
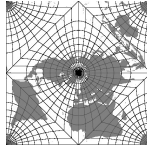
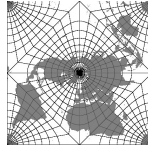
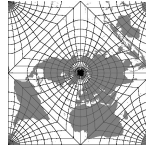
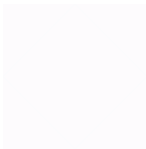
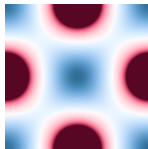

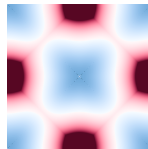

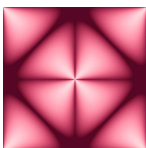

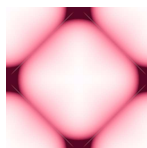
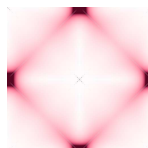
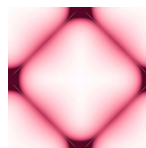
Hemisphere to disc mapping	Lambert	Stereographic	Harmonic Mean	Mixture	Equidistant
Disc to square mapping	Shirley	Conformal	Elliptical	Squirele	Elliptical
Example map					
D_A					
D_I					

Table 3. Selected mappings between a sphere and a square in quincuncial layout.

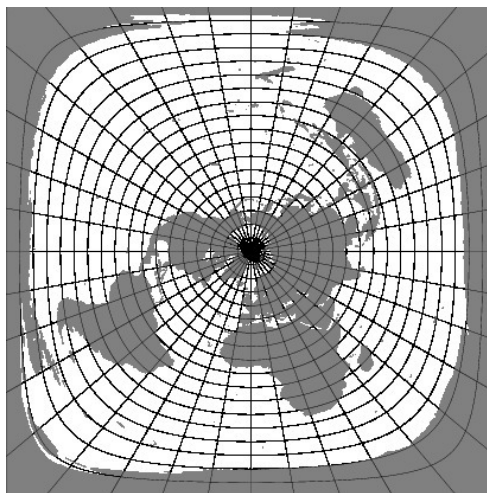


Figure 2. Example map with the south pole of the sphere mapped to the square border. Here the equidistant azimuthal projection was combined with the disc-to-square mapping based on Fernández-Guasti's squircle.

4.1. Pole-at-border Layout

The first layout option projects the whole sphere onto a disc using one of the methods from Section 3 that is capable of this, and then applies one of the methods from Section 2 to map that disc to a square.

In this layout, the sphere point opposite the projection center is mapped to the border of the square. In our examples, the projection center is the north pole, and the south pole is spread across the border. Figure 2 shows an example map.

Obviously, this layout leads to very strong distortions in the region around the point opposite the projection center. This limits its usefulness in applications that require acceptable sampling quality throughout the map, but there are still use cases for this layout, for example in panoramic imaging [Fong 2014].

4.2. Quincuncial Layout

Peirce's quincuncial map projection [Peirce 1879] shown in Figure 3 introduced this layout. It projects each hemisphere onto its own disc, maps these discs to two squares, and then arranges the squares into a single square as depicted in Figure 4. The center part and the four corner parts form a quincunx.

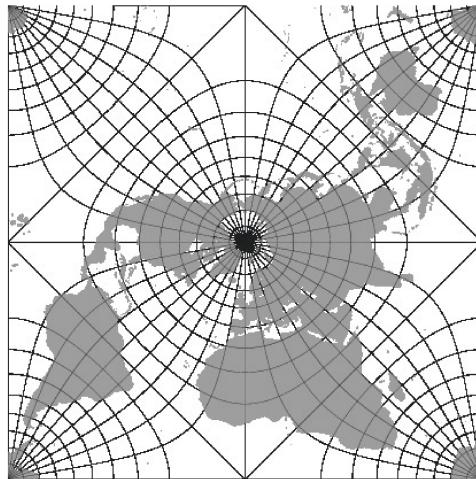


Figure 3. The original quincuncial map by Peirce combines stereographic conformal projection from each hemisphere to a disc, conformal mapping of these discs to two squares, and arrangement of the squares in a quincuncial layout.

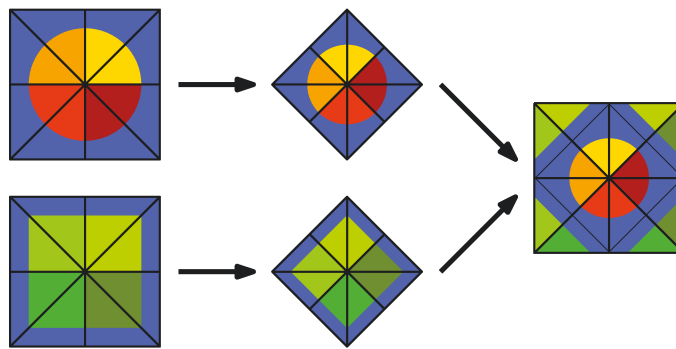


Figure 4. Arrangement of two squares into a new square in quincuncial layout. The second square is mirrored along its borders to fit into the new square.

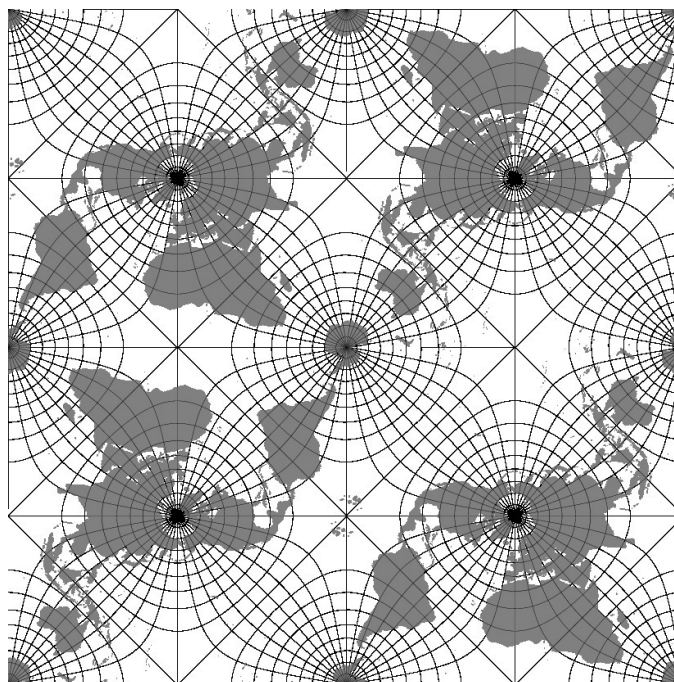


Figure 5. Tiling of the quincuncial layout. The upper-left and lower-right quarters are the original map, the upper-right and lower-left quarters are rotated by 180° .

The sphere point opposite the projection center is mapped to the four corners of the square. Distortions are much smaller than in the opposite-point-at-border layout. Additionally, the quincuncial layout has the nice property that maps can be tiled, as shown in Figure 5: when leaving the map at any border point except the four corners, in any direction, there is a continuation point where we can re-enter the map without disruption.

The quincuncial layout has been used in computer graphics to represent an octahedron inside a square for the purpose of vector representation [Meyer et al. 2010; Cigolle et al. 2014] or sphere parametrization [Praun and Hoppe 2003] and for the display of panoramic images [German et al. 2007].

To arrange two unit squares into a unit square in quincuncial layout, both must first be rotated by -45° and scaled by $1/\sqrt{2}$. Each quadrant of the second input square R_2 is then mirrored along its border in that quadrant; see Figure 4.

Selected mappings in quincuncial layout are summarized and compared in Table 3. Note that there are methods that map directly between sphere and octahedron or square, without using the disc as an intermediate step. These include Snyder's equal-area map projection for polyhedral globes [Snyder 1992] and Gringorten's square equal-area world map [Gringorten 1972]. Furthermore, Clarberg describes an optimized SIMD implementation of a mapping between sphere and square using Shirley's equal-area mapping and an octahedral layout [Clarberg 2008]. These methods are not discussed in this paper.

5. Analysis

In this section, we derive quality measurements and provide numerical analysis results for all mappings. Note that all results were obtained using IEEE double precision floating-point numbers and computations.

5.1. Distortion Measurements

To evaluate the mapping methods discussed in the preceding sections, we use a standard tool from the field of map projection: Tissot's indicatrix [Snyder 1987].

The idea of the indicatrix is that any map projection maps an infinitesimal circle on the sphere onto an infinitesimal ellipse on the map. This ellipse describes the local characteristics of the map projection. For example, a conformal map projection preserves angles, and the local ellipse will therefore be a circle, but generally not of the same size as the original circle. An equal-area map projection preserves area, and the size ratio of the local ellipse to the original circle will be constant throughout the map, but the axes of the ellipse will have varying orientation and length.

In general, for any pair of lines that intersect at a given point on the sphere, the angle at which they intersect on the map will not be identical (unless the map projection is conformal). The greatest deviation from the correct angle at a given point is called the maximum angular deformation ω .

Both the original circle and the mapped ellipse are infinitesimal, but Tissot's indicatrix allows to compute the ratio between corresponding properties. The most important ellipse properties are the semi-major axis a and the semi-minor axis b . Both are measured relative to the original circle: the identity mapping will result in $a = b = 1$.

For applications in computer graphics, two values derived from a and b are of special interest, since they determine the sampling quality of a map projection:

- *The local area distortion D_A .* The determinant of the Jacobian matrix of a mapping gives the local scale factor s , which can also be determined numerically as $s = ab$. In a strict sense, s must be 1 everywhere on the map for the mapping to be truly area-preserving, but usually source and destination have different areas, and an equal-area mapping will have a constant scale factor representing

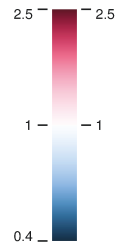


Figure 6. Color map for D_A and D_I in Tables 1, 2, 3 and Figure 7. D_A (left scale) can deviate upward or downward from the ideal value 1, with direction of the deviation encoded in hue. D_I (right scale) can only deviate upward from 1, where it uses the same color encoding as D_A .

the ratio R of destination and source areas, e.g. $s \equiv 4/\pi$ for equal-area mappings from disc to square. We therefore use $D_A = ab/R$ as a measurement of local area distortion.

- *The local isotropy distortion D_I .* The ratio of a and b is a measurement for the isotropy distortion: $D_I = a/b$. The ideal value is 1. Larger anisotropy typically leads to degraded sampling quality in computer graphics applications. Note that conformal maps have $D_I \equiv 1$, but $D_I \equiv 1$ does not imply that a map is conformal: ω might still be larger than zero locally.

These two distortion measurements are color-coded using a color map generated with the methods described by Wijffelaars et al. [Wijffelaars et al. 2008] as shown in Figure 6; they are displayed for all relevant mappings in Tables 1, 2, and 3.

Table 1 shows that for mappings between disc and square, D_A tends to increase strongly in the corners of the square, except for the radial stretching method and, of course, Shirley’s equal-area mapping. In fact, D_A grows indefinitely in the corners for Fernández-Guasti’s squircle method, elliptical arc mapping, and conformal mapping, with conformal mapping showing the largest errors and affected areas. D_I is strongest at the square diagonals for the radial stretching mapping and Shirley’s equal-area mapping, but it does not grow indefinitely for these methods. Fernández-Guasti’s squircle method and elliptical arc mapping show indefinitely growing D_I values in the corners of the square. For conformal mapping, $D_I \equiv 0$ as expected.

Interestingly, Shirley’s equal-area mapping exhibits the lowest D_I measurements of the non-conformal mappings. It is therefore a good candidate if its discontinuity at the square diagonals is acceptable for the application. On the other hand, Fernández-Guasti’s squircle method and elliptical arc mapping provide small D_A and D_I values for the largest part of the square, as can also be seen by looking at the mapped teapot shapes; if an application is mainly interested in the interior of the square and not its corners, these methods are good candidates.

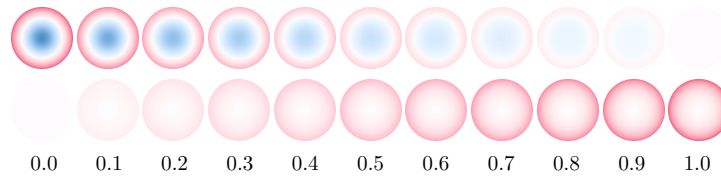


Figure 7. D_A (top) and D_I (bottom) of the method described in Section 3.4, for varying β .

Table 2 shows that the harmonic mean, mixture, and equidistant mappings all provide good compromises in terms of D_A and D_I when compared to the equal-area and conformal mappings. We used $\beta = 0.4$ for the mixture method since it delivers the best compromise; see Figure 7. Still, the harmonic mean method might be preferable if an adjustable error tradeoff is not required.

Table 3 shows only a small subset of the many possible quincuncial combinations of methods. The distortions introduced by sphere/disc and disc/square mappings combine. Compromises between equal-area and conformal mappings still exhibit relatively large values for D_A and D_I at the corners of the inner square, mainly caused by the disc/square mapping. For some applications, it is possible to position these four points in regions of low interest, e.g. oceans for applications that visualize Earth landmass data.

5.2. Precision Measurements

We tested the numerical precision of our implementation by placing a uniform grid on the square map, applying inverse mapping to each grid point to obtain coordinates in the original domain (disc or sphere), and then mapping these coordinates first to the map and then back again. For each obtained point in the original domain, we then have a distance to the point resulting from forward and inverse transformation. The maximum distance is the error measurement.

The mappings between disc and square all have very low errors. Measured on Earth's equatorial disc, the distances are in the micrometer range or below, with two exceptions: Fernández-Guasti's squircle mapping has an error in the millimeter range, and conformal mapping has an error in the meter range. Of course, these are implementation dependent, but we did not arrive at lower distances for the conformal mapping when using different numerical libraries for the computation of F and cn .

The errors of mappings between sphere and square are dominated by the errors of the disc/square mappings used. On an Earth-sized sphere, they are again mostly in the micrometer range, except when Fernández-Guasti's squircle mapping or conformal mapping are used, in which case errors in the millimeter (squircle) or meter range (conformal) occur.

6. Conclusion

This paper gives an overview of mappings between a disc and a square, between a sphere and a disc, and (by composition) between a sphere and a square. It provides ready-to-implement formulas for both the forward and inverse mappings, and it analyzes all mappings using distortion measurements that are relevant for applications in computer graphics.

Since requirements in terms of mapping properties strongly depend on the application area, general recommendations cannot be made. Instead, this paper provides analysis results intended to help pick the right mapping for a given application.

Acknowledgements

The polygonal world map data used in maps throughout this paper is provided by Bjorn Sandvik, http://thematicmapping.org/downloads/world_borders.php, license CC BY-SA 3.0.

References

- CIGOLLE, Z. H., DONOW, S., EVANGELAKOS, D., MARA, M., MCGUIRE, M., AND MEYER, Q. 2014. A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques* 3, 2 (April), 1–30. URL: <http://jcgt.org/published/0003/02/01/>. 5, 15
- CLARBERG, P. 2008. Fast equal-area mapping of the (hemi)sphere using SIMD. *Journal of Graphics, GPU, and Game Tools* 13, 3, 53–68. URL: <http://dx.doi.org/10.1080/2151237X.2008.10129263>. 16
- FERNÁNDEZ-GUASTI, M. 1992. Classroom notes: Analytic geometry of some rectilinear figures. *Int. J. Mathematical Education in Science and Technology* 23, 6, 895–913. URL: <http://dx.doi.org/10.1080/0020739920230607>. 5
- FLOATER, M. S., AND HORMANN, K. 2005. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*. Springer, New York, 157–186. URL: http://dx.doi.org/10.1007/3-540-26808-1_9. 2
- FONG, C. 2014. An indoor alternative to stereographic spherical panoramas. In *Proc. Bridges 2014: Mathematics, Music, Art, Architecture, Culture*, 103–110. URL: <http://archive.bridgesmathart.org/2014/bridges2014-103.html>. 1, 5, 10, 13
- FONG, C., 2015. Analytical methods for squaring the disc. <http://arxiv.org/abs/1509.06344>. Accessed: 2015-10-06. 2, 6
- GERMAN, D. M., D'ANGELO, P., GROSS, M., AND POSTLE, B. 2007. New methods to project panoramas for practical and aesthetic purposes. In *Proc. Computational Aesthetics in Graphics, Visualization, and Imaging*, Eurographics Association, Aire-la-Ville, Switzerland. URL: <http://dx.doi.org/10.2312/COMPAESTH/COMPAESTH07/015-022>. 1, 15

- GREENE, N. 1986. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications* 6, 11 (Nov.), 21–29. URL: <http://dx.doi.org/10.1109/MCG.1986.276658>. 1
- GRINGORTEN, I. I. 1972. A square equal-area map of the world. *Journal of Applied Meteorology* 11, 5, 763–767. URL: [http://dx.doi.org/10.1175/1520-0450\(1972\)011<0763:ASEAMO>2.0.CO;2](http://dx.doi.org/10.1175/1520-0450(1972)011<0763:ASEAMO>2.0.CO;2). 16
- HEIDRICH, W., AND SEIDEL, H.-P. 1998. View-independent environment maps. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, ACM, New York, 39–ff. URL: <http://dx.doi.org/10.1145/285305.285310>. 1
- LEE, L. 1976. Conformal projections based on jacobian elliptic functions. *Cartographica* 13, 1, 67–101. URL: <http://dx.doi.org/10.3138/X687-1574-4325-WM62>. 6
- MEYER, Q., SÜSSMUTH, J., SUSSNER, G., STAMMINGER, M., AND GREINER, G. 2010. On floating-point normal vectors. *Computer Graphics Forum* 29, 4, 1405–1409. URL: <http://dx.doi.org/10.1111/j.1467-8659.2010.01737.x>. 15
- NOWELL, P., 2005. Mapping a square to a circle. <http://mathproofs.blogspot.com/2005/07/mapping-square-to-circle.html>. Accessed: 2015-10-06. 5
- PEIRCE, C. 1879. A quincuncial projection of the sphere. *American Journal of Mathematics* 2, 4, 394–396. URL: <http://dx.doi.org/10.2307/2369491>. 6, 13
- PRAUN, E., AND HOPPE, H. 2003. Spherical parametrization and remeshing. *ACM Trans. Graph.* 22, 3 (July), 340–349. URL: <http://dx.doi.org/10.1145/882262.882274>. 15
- ROŞCA, D. 2010. New uniform grids on the sphere. *Astronomy & Astrophysics* 520, A63. URL: <http://dx.doi.org/10.1051/0004-6361/201015278>. 4
- SHIRLEY, P., AND CHIU, K. 1997. A low distortion map between disk and square. *Journal of graphics tools* 2, 3, 45–52. URL: <http://dx.doi.org/10.1080/10867651.1997.10487479>. 1, 4
- SHIRLEY, P., AND CLINE, D., 2011. Improved code for concentric map. <http://psgraphics.blogspot.de/2011/01/improved-code-for-concentric-map.html>. Accessed: 2016-03-18. 2, 4
- SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. Graph.* 24, 3 (July), 1216–1224. URL: <http://dx.doi.org/10.1145/1073204.1073335>. 1
- SNYDER, J., AND MITCHELL, D., 2001. Sampling-efficient mapping of spherical images. http://research.microsoft.com/en-us/um/people/johnsny/papers/spheremap_tr.pdf. Microsoft Research Technical Report. 2
- SNYDER, J. 1987. *Map projections—a working manual*, vol. 1395 of *Professional Paper*. US Geological Survey. 8, 11, 16
- SNYDER, J. 1992. An equal-area map projection for polyhedral globes. *Cartographica* 29, 1, 10–21. URL: <http://dx.doi.org/10.3138/27H7-8K88-4882-1752>. 16

- STARK, M. M. 2009. Fast and stable conformal mapping between a disc and a square. *Journal of Graphics, GPU, and Game Tools 14*, 2, 1–23. URL: <http://dx.doi.org/10.1080/2151237X.2009.10129277>. 7
- WIJFFELAARS, M., VLIEGEN, R., VAN WIJK, J. J., AND VAN DER LINDEN, E.-J. 2008. Generating color palettes using intuitive parameters. *Computer Graphics Forum 27*, 3, 743–750. doi:10.1111/j.1467-8659.2008.01203.x. 17
- YOUNG, A. E. 1920. *Some Investigations in the Theory of Map Projections*. Royal Geographical Society, London. URL: <http://www.archive.org/details/cu31924029951401>. 10

Index of Supplemental Materials

The supplementary material found at <http://www.jcgt.org/published/0005/02/01/code.zip> consists of C++ source code implementing all mapping and analysis methods, licensed under the MIT/Expat license. See included README file for an overview.

Author Contact Information

Martin Lambers
Computer Graphics Group
University of Siegen
Hölderlinstraße 3
57076 Siegen
martin.lambers@uni-siegen.de

Martin Lambers, Mappings between Sphere, Disc, and Square, *Journal of Computer Graphics Techniques (JCGT)*, vol. 5, no. 2, 1–21, 2016
<http://jcgt.org/published/0005/02/01/>

Received: 2015-11-05
Recommended: 2016-03-15
Published: 2016-04-15

Corresponding Editor: Peter Shirley
Editor-in-Chief: Marc Olano

© 2016 Martin Lambers (the Authors).
The Authors provide this document (the Work) under the Creative Commons CC BY-ND 3.0 license available online at <http://creativecommons.org/licenses/by-nd/3.0/>. The Authors further grant permission for reuse of images and text from the first page of the Work, provided that the reuse is for the purpose of promoting and/or summarizing the Work in scholarly venues and that any reuse is accompanied by a scientific citation to the Work.



Lowering the entry barrier for students programming Virtual Reality applications

M. Lambers

Computer Graphics Group, University of Siegen, Germany

Abstract

In Computer Graphics, it is common practice to accompany lectures with hands-on tutorials and/or project assignments that allow students to write and run their own interactive graphics applications. In the special case of Virtual Reality courses, this approach is difficult to maintain since the software requirements pose a high entry barrier to students.

In this paper, we propose a technique to significantly simplify Virtual Reality application programming, and implement it in an easy-to-use framework that supports the full range of typical Virtual Reality hardware setups, from head-mounted displays to multi-node, multi-GPU render clusters. The framework lowers the entry barrier for students and allows them to focus on course goals instead of fighting software complexities.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computers and Education]: Computer and Information Science Education—Computer Science Education I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual and Augmented Reality

1. Introduction

Virtual Reality (VR) courses at universities often target postgraduate students because the entry barrier regarding hardware and software has traditionally been high. Recently, the hardware situation improved significantly with the availability of affordable sensor, display, and interaction devices. Sousa Santos et al. [SDM15] documented a course taking advantage of this development.

However, software requirements for VR application programming are still complex, and finding the right software to base a course on is difficult [SDSS14, SDM15]. To cover the full range of common VR setups, applications must be able to handle multi-GPU and multi-node systems [EMP09]. The complexity of distributed graphics applications is usually handled by a specialized framework, but available frameworks come with their own complexities and steep learning curves.

Consistent with observations made by Boers et al. [BDHB08] and Anderson and Peters [AP10], our experience with past courses shows that students lose time and motivation fighting software complexities instead of focusing on course goals. Therefore, the hands-on tutorial part of our VR course was limited in comparison to other computer graphics courses, and student projects in VR typically had a difficult and time-consuming start phase.

In this paper, we focus on reducing the software-side complexity of VR courses, with the goals of increasing hands-on tutorial content and enabling more student projects. We define the following requirements for a VR software framework:

1. The framework must be free and open-source. We agree with Sousa Santos et al. [SDM15] that in academic contexts paying for VR software licenses is problematic, especially when students can use their own devices.
2. The framework must allow the use of external rendering software, but also plain OpenGL. High-level rendering engines are useful for advanced student projects and thesis works, but simple tutorial assignments should not require the students to learn about such an engine when plain OpenGL is sufficient.
3. The framework must support the full range of common VR graphics hardware, including head-mounted displays and multi-GPU/multi-host setups. Applications based on the framework must run in CAVE-like VR environments and other hardware available in a VR lab, and also on the students' own devices.
4. The framework must be easy to set up and work with. Students must be able to quickly reach a productive familiarity with the framework so that they can focus on course goals.

To arrive at a VR framework that fulfills these requirements, we propose a technique to simplify multi-window, multi-GPU, and multi-process handling in Sec. 3, and describe its implementation in Sec. 4. Sec. 5 shows example results, and Sec. 6 discusses the current state and future work.

2. Related Work

Software frameworks used in VR courses at universities include in-house frameworks that are not publicly available [Sta05] and/or are

```

while application is running do
  preRenderProcess ();
  foreach window w do
    preRenderWindow (w);
    for i ← 1 to stereoRenderPasses (w) do
       $M_i \leftarrow \text{viewMatrix}(w, i)$ ;
       $F_i \leftarrow \text{frustum}(w, i)$ ;
       $T_i \leftarrow \text{texture}(w, i)$ ;
      render ( $M_i, F_i, T_i$ );
    end
    postRenderWindow (w);
  end
  postRenderProcess ();
  foreach window w do
    updateDisplay (w,  $T_1, T_2$ );
    asyncBufferSwap (w);
  end
  processEvents ();
  updateApplication ();
  waitForBufferSwaps ();
end

```

Algorithm 1: The main loop in the single-context single-thread approach. Multi-process handling is omitted for brevity.

not actively developed [Ant09, DK11], frameworks that are coupled to a particular rendering, scene graph, or game engine [Tra99, vRKG*00, LCC*12, SDSS14] and scene graph engines with multi-GPU or multi-host support [Rei02], and web-oriented techniques such as VRML and WebGL [Zar06, SDM15]. None of the solutions from these categories fulfill the requirements defined in Sec. 1.

The only two frameworks we could find which fulfill the requirements 1–3 are VR Juggler [BJH*01] and Equalizer [EMP09]. Both fail to fulfill requirement 4. As noted by Sousa Santos [SDM15], VR Juggler is complex to install and set up, and support for recent hardware is absent. We also note that development seems to have slowed down significantly, or even stopped.

We have therefore based our VR courses, tutorials, and projects on Equalizer in the last years. While it is a very powerful and flexible framework, capable of much more than just VR application scenarios, in our experience it is also very hard for students to work with. We have observed the following major obstacles:

- Install-and-setup obstacle: Equalizer is split into many sub-libraries and requires several external libraries that a custom build script partly tries to download and install during the Equalizer build. This process fails regularly on relevant platforms.
- Hierarchy-level obstacle: Distributing program logic over the Equalizer hierarchy levels (see Sec. 3) runs counter to the students' previous experiences with object-oriented programming, which is to group program logic according to its purpose.
- Multi-context obstacle: Equalizer uses multiple OpenGL contexts, and contexts on the same GPU share objects such as textures. Students are typically not familiar with OpenGL contexts, context sharing, and context/thread binding. As a result, they struggle to understand which context is active at which time and in which thread, and how OpenGL objects should be managed across multiple contexts and/or multiple GPUs.
- Multi-thread obstacle: Equalizer uses multiple rendering threads

```

class VRApplication {
  /* Mandatory functions */
  // Update scene state (animations etc).
  // Called once before each new frame.
  void update (...) = 0;
  // Render the scene into texture T using
  // frustum F and view matrix M.
  void render (M, F, T) = 0;
  /* Optional functions */
  // Event handling, using Qt conventions
  void keyPressEvent (...) {}
  void mousePressEvent (...) {}
  void mouseMoveEvent (...) {}
  // Special per-process/per-window actions
  void preRenderProcess (...) {}
  void preRenderWindow (...) {}
  void postRenderWindow (...) {}
  void postRenderProcess (...) {}
  // Multi-process support
  void serializeDynamicData (...) const {}
  void deserializeDynamicData (...) {}
};

```

Figure 1: Summary of the interface that a VR application needs to implement. Note that most functions are optional.

within each process on multi-GPU systems. Many students struggle to grasp all consequences of this approach, especially when integrating third-party software, and consequently run into multithreading pitfalls that are notoriously hard to debug.

Like Sousa Santos et al. [SDM15], we prefer simple software that allows “actually understanding the whole process necessary to create a virtual environment”. Furthermore, we agree with Boers et al. [BDHB08] and Anderson and Peters [AP10] that students need to focus on course goals instead of losing time and motivation fighting with framework setup and usage complexities.

3. Simplified VR Application Programming

A central function of a VR framework is the handling of multi-GPU and multi-node render systems. Challenges associated with multi-GPU programming are listed in the Parallel OpenGL FAQ [Eih07]. The main aspects relevant to VR are the following:

- An OpenGL context can only be bound to one thread at a time, and a switch of that thread is expensive. Therefore, all rendering to a context should happen from only one thread.
- Access to a GPU is serialized by the driver. Rendering into different contexts on the same GPU therefore best happens in a serial manner, to avoid unnecessary costly context switches.
- The swapping of back and front buffers of a context is typically synchronized with the refresh rate of the connected display. The function that triggers the swap blocks the calling thread until that swap happens.
- OpenGL contexts can share objects such as textures if they live on the same GPU.
- Multi-GPU systems need to provide a way for an application to differentiate between GPUs. This is system dependant.

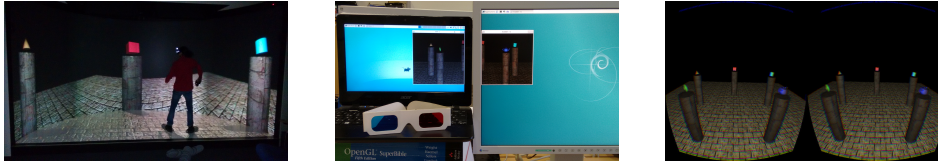


Figure 2: Example VR application running in a CAVE-like VR lab (left), across a laptop and a desktop PC (middle), and on the Oculus Rift DK2 head-mounted display (right).

The Equalizer framework handles these challenges using a hierarchy of processes, GPUs, windows, and channels. Each process can handle multiple GPUs, each GPU can drive multiple windows, and each window can be divided into multiple channels. Each window has its own context, and contexts on the same GPU share objects. Each GPU has a dedicated rendering thread.

While very flexible and powerful, this *multi-context multi-thread* approach introduces a lot of complexity: an application needs to split operations and data over four hierarchy levels, it has to manage multiple contexts that share objects only if they live on the same GPU, and it has to manage concurrent access to process-level resources by different rendering threads.

We propose the following simplified *single-context single-thread* approach. There are only two hierarchy levels: process and window. Each process handles only one GPU, and maintains a master OpenGL context on that GPU that is not connected to any visible window. This is the only context a VR application process sees. All windows of a process have their own private contexts that share objects with the master context. The windows are represented by textures in the master context. The private window contexts are driven by private rendering threads that display these textures and wait for the buffer swap, while the main thread is free for other work such as application scene state updates. The corresponding render logic is summarized in Alg. 1.

In this approach, the application programmer only handles a set of textures in a single OpenGL context, and renders different views into these textures sequentially from the main thread. The multi-context and multi-thread obstacles are eliminated.

Multiple processes, both for multi-GPU and multi-host setups, are handled by serializing relevant VR application data (mostly scene state) on the master process, writing it via local or network pipes to the slave processes, and deserializing it there. The application only needs to implement serialization and deserialization logic.

One potential drawback of our approach affects multi-GPU hosts. We use separate processes for each GPU, and inter-process communication via pipes. Equalizer uses separate threads, thus avoiding communication overhead. However, we found that the complexities and pitfalls of multi-threaded rendering often drove our students to switch to multi-process but single-threaded Equalizer configurations, thereby eliminating the advantage. Furthermore, we expect that the communication costs are tolerable when shared memory or similar techniques are used for processes on the same host.

4. A Simple VR Application Framework

We based our implementation of the single-context single-thread approach on the C++ language and the Qt library because our students are already familiar with both from other courses, and because both offer widely used and easy-to-setup development tools. This eliminates the install-and-setup obstacle.

The interface that a VR application has to implement is summarized in Fig. 1. Since only a single interface needs to be implemented, the hierarchy-level obstacle is removed. To allow students to quickly become familiar with the framework, the interface is kept minimal, and optional components have an empty default implementation.

The configuration file concept of Equalizer has been very useful in our experience, and we copy it in our framework. A configuration file defines the VR application processes, one for each GPU, and their properties. For each process, it defines windows with attributes such as stereo mode, size, position, and projection area geometry. When starting the VR application, the running process is assumed to be the master process defined first in the configuration file. Slave processes are started automatically by the framework. The application can run unmodified on different display hardware setups by using different configuration files.

5. Results

The simplifications proposed in Sec. 3 are made possible by graphics hardware and API capabilities that are now ubiquitous (render-to-texture via framebuffer objects, off-screen contexts, context object sharing). All platform dependent aspects are handled by Qt, keeping the code base of the framework small.

To demonstrate the versatility in terms of display hardware setups, we show a simple example application using different configuration files for different hardware setups in Fig. 2. The first setup (left) is a CAVE-like VR lab. Its render system has four GPUs, each connected to three projectors for a total of six passive stereo projection areas. The second setup (middle) demonstrates multi-host support using a laptop and a desktop PC. The third setup (right) is an Oculus Rift DK2 head-mounted display.

While the simple example application is written in plain OpenGL, more complex applications often require external rendering engines. The integration of such engines is simplified by our single-context single-thread approach since this corresponds well

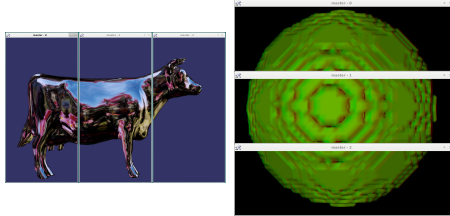


Figure 3: Left: OpenSceneGraph viewer running in a three-window configuration with monoscopic views. Right: a VTK visualization pipeline running in a three-window configuration with stereoscopic views for red-cyan anaglyph glasses.

to the common usage scenario of embedding a rendering engine into an application-managed graphics window.

The versatility in terms of rendering approaches is demonstrated in Fig. 3. The left side shows a scene rendered by a full-featured OpenSceneGraph viewer. The integration of OpenSceneGraph with our framework requires only few lines of code, while its integration with Equalizer (osgScaleViewer; included in the Equalizer source code) is much more complex and still does not provide full functionality. The right side of the figure shows a VTK example that renders an isosurface extracted from a voxelized sphere. The integration of VTK is equally simple and does not impose any restrictions on VTK visualization pipelines.

Full source code is available under the MIT/Expat license at <https://github.com/marlam/qvr>. This includes the framework, called QVR, and all examples shown in this section.

6. Discussion and Conclusions

The four complexity obstacles we observed while using Equalizer are removed: the single-context single-thread approach removes the multi-context and multi-threading obstacles, and the framework implementation removes the install-and-setup and hierarchy-level obstacles.

The framework handles distributed graphics only. Applications typically use third-party libraries for other VR-related tasks, e.g. VRPN [THS*01] for tracking devices. On the one hand, this requires teachers to provide additional software depending on their lab equipment and course topics. On the other hand, it keeps the framework lightweight and easy to set up, and lowers maintenance burden.

The framework is currently in a testing stage. Initial feedback from selected students is encouraging to the point that we decided to base our future VR courses on it. However, further tweaks will likely become necessary once the framework is in wider use. For example, we plan to measure the impact of the potential communication overhead in multi-GPU systems for real-world applications, and implement appropriate countermeasures such as the use of shared memory instead of pipes if required.

References

- [Ant09] ANTHES C.: *A Collaborative Interaction Framework for Networked Virtual Environments*. PhD thesis, Institute of Graphics and Parallel Processing at JKU Linz, Austria, August 2009. 2
- [API0] ANDERSON E. F., PETERS C. E.: No more reinventing the virtual wheel: Middleware for use in computer games and interactive computer graphics education. In *Eurographics - Education Papers* (2010). doi:10.2312/eged.20101013. 1, 2
- [BDHB08] BOERS J., DOBBE J., HUIJSER R., BIDARRA R.: From a light CG framework to a strong cannibal experience. In *Eurographics - Education Papers* (2008). doi:10.2312/eged.20081002. 1, 2
- [BJH*01] BIERBAUM A., JUST C., HARTLING P., MEINERT K., BAKER A., CRUZ-NEIRA C.: VR Juggler: a virtual platform for virtual reality application development. In *IEEE Proc. Virtual Reality* (March 2001), pp. 89–96. doi:10.1109/VR.2001.913774. 2
- [DK11] DOERR K.-U., KUESTER F.: CGLX: A scalable, high-performance visualization framework for networked display environments. *IEEE Trans. Visualization and Computer Graphics* 17, 3 (March 2011), 320–332. doi:10.1109/TVCG.2010.59. 2
- [Ei07] EILEMANN S.: Parallel OpenGL FAQ. <http://www.equalizergraphics.com/documentation/parallelOpenGLFAQ.html>, 2007. Accessed: 2015-12-20. 2
- [EMP09] EILEMANN S., MAKHINYA M., PAJAROLA R.: Equalizer: A scalable parallel rendering framework. *IEEE Trans. Visualization and Computer Graphics* 15, 3 (May 2009), 436–452. doi:10.1109/TVCG.2008.104. 1, 2
- [LCC*12] LUGRIN J.-L., CHARLES F., CAVAZZA M., LE RENARD M., FREEMAN J., LESSITER J.: CaveUDK: A VR game engine middleware. In *Proc. ACM Symp. on Virtual Reality Software and Technology* (2012), pp. 137–144. doi:10.1145/2407336.2407363. 2
- [Rei02] REINERS D.: *OpenSG: A scene graph system for flexible and efficient realtime rendering for virtual and augmented reality applications*. PhD thesis, Technische Universität Darmstadt, 2002. 2
- [SDM15] SANTOS B. S., DIAS P., MADEIRA J.: A virtual and augmented reality course based on inexpensive interaction devices and displays. In *Eurographics - Education Papers* (2015). doi:10.2312/eged.20151022. 1, 2
- [SDSS14] SOUZA D., DIAS P., SANTOS D., SANTOS B. S.: Platform for setting up interactive virtual environments. In *Proc. SPIE 9012, The Engineering Reality of Virtual Reality* (2014), pp. 901200–1–901200–9. doi:10.1117/12.2038668.7. 1, 2
- [Sta05] STANSFIELD S.: An introductory VR course for undergraduates incorporating foundation, experience and capstone. In *Proc. 36th SIGCSE Technical Symposium on Computer Science Education* (2005), SIGCSE, pp. 197–200. doi:10.1145/1047344.1047417. 1
- [THS*01] TAYLOR II R. M., HUDSON T. C., SEEGER A., WEBER H., JULIANO J., HELSER A. T.: VRPN: A device-independent, network-transparent VR peripheral system. In *Proc. ACM Symp. on Virtual Reality Software and Technology* (2001), pp. 55–61. doi:10.1145/505008.505019. 4
- [Tra99] TRAMBEREND H.: Avocado: a distributed virtual reality framework. In *IEEE Proc. Virtual Reality* (Mar 1999), pp. 14–21. doi:10.1109/VR.1999.756918. 2
- [vRKG*00] VAN REIMERSDAHL T., KUHLEN T., GERNDT A., HENRICHS J., BISCHOF C.: ViSTA: a multimodal, platform-independent VR-toolkit based on WTK, VTK, and MPI. In *Proc. 4th International Immersive Projection Technology Workshop* (2000). 2
- [Zar06] ZARA J.: Virtual reality course - a natural enrichment of computer graphics classes. *Computer Graphics Forum* 25, 1 (2006), 105–112. doi:10.1111/j.1467-8659.2006.00921.x. 2

**COMPARISON OF SPHERICAL CUBE MAP PROJECTIONS
USED IN PLANET-SIZED TERRAIN RENDERING**

Aleksandar M. Dimitrijević, Martin Lambers and Dejan D. Rančić

Abstract. A wide variety of projections from a planet surface to a two-dimensional map are known, and the correct choice of a particular projection for a given application area depends on many factors. In the computer graphics domain, in particular in the field of planet rendering systems, the importance of that choice has been neglected so far and inadequate criteria have been used to select a projection. In this paper, we derive evaluation criteria, based on texture distortion, suitable for this application domain, and apply them to a comprehensive list of spherical cube map projections to demonstrate their properties.

Keywords: Map projection, spherical cube, distortion, texturing, graphics

1. Introduction

Map projections have been used for centuries to represent the curved surface of the Earth with a two-dimensional map. A wide variety of map projections have been proposed, each with different properties. Of particular interest are scale variations and angular distortions introduced by map projections – since the spheroidal surface is not developable, a projection onto a plane cannot be both conformal (angle-preserving) and equal-area (constant-scale) at the same time. These two properties are usually analyzed using Tissot's indicatrix. An overview of map projections and an introduction to Tissot's indicatrix are given by Snyder [24].

In computer graphics, a map projection is a central part of systems that render planets or similar celestial bodies: the surface properties (photos, digital elevation models, radar imagery, thermal measurements, etc.) are stored in a map hierarchy in different resolutions. During the rendering, the data from this map hierarchy are sampled for display on a screen. Despite its central role, many systems do not pay much attention to the choice of projection for the map hierarchy. Often, a relatively straightforward approach is used, which leads to sampling problems both during the creation of the map hierarchy and during its sampling at rendering time.

Received February 19, 2015; accepted January 18, 2016
2010 *Mathematics Subject Classification.* Primary 68U05; Secondary 65D18, 68W25

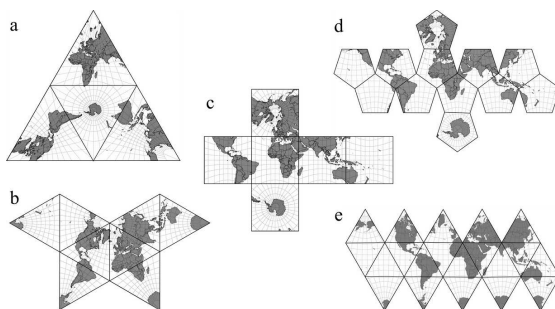


FIG. 1.1: Polyhedral projections based on Platonic solids: tetrahedron (a), octahedron (b), hexahedron (c), dodecahedron (d) and icosahedron (e).

This is particularly apparent with systems that use a single map to cover the whole planet surface. Such systems usually exhibit strong distortions and artifacts in the polar regions. Examples include the well-known commercial products like Google Earth [5] and NASA World Wind [15]. A few systems do care about the projection they use, but use insufficient evaluation criteria and/or evaluate too few alternatives to make a good choice [10, 13, 11].

Projecting a spheroidal surface to a single plane (flat or folded into a cylinder or a cone) always results in singularities [10, 24], therefore the first step for improvement is to subdivide the spheroidal surface into several regions, each of which is projected to a separate projection plane. The subdivision reduces map distortion, but increases the number of interrupts. The faces of an encompassing or inscribed polyhedron are very good candidates for the projection planes, hence the polyhedral projections have been used for centuries to represent the surface of the Earth. Fig. 1.1 displays unfolded polyhedral projections based on Platonic solids.

As it can be seen in Fig. 1.1, the increase of the polyhedral faces number reduces distortion and increases interrupts at the same time. The number of interrupts is also an important aspect of a map projection. For paper maps, interrupts make visual discontinuities, while for electronic maps (i.e. textures) they may require separate data sets for each region. The number of data sets may have a direct impact on the memory usage [4]. Therefore, it should be minimized if possible.

For the purposes of computer graphics, the projection to the faces of a cube (as a special form of a hexahedron) is of particular interest because each face is rectangular and thus allows straightforward storage of map data in common 2D file formats, as well as management of rendering data in common 2D texture formats. Also, cube based projections expose moderate distortion and number of interrupts.

In this paper, we derive a set of evaluation criteria, based on texture distortion,

and apply them to a comprehensive list of hexahedral map projections to demonstrate their properties. This list covers all spherical cube map projection known to be used in planet-sized terrain rendering. One of the projections (Outerra Spherical Cube Map) is published for the first time, thanks to its original implementer Brano Kemen [8]. We implemented all of these projections as well as map projection software and a set of evaluation tests.

The remainder of this paper is organized as follows. Sec. 2. gives an overview of related work in the field of planet rendering, with an emphasis on the choice of map projections. In Sec. 3., we derive the evaluation criteria that we apply to spherical cube map projections, reviewed in Sec. 4. The results of this evaluation are presented and discussed in Sec. 5. Finally, Sec. 6. concludes the paper.

2. Related Work

Map projections from a sphere surface to a plane have a long history, and a wide variety of methods have been developed, each with specific properties carefully chosen for specific tasks. An overview is given by Snyder [24].

A popular map projection for planet rendering systems, including the commercial offerings Google Earth [5] and NASA World Wind [15], is the equidistant cylindrical (or plate carrée) projection. Like all single-map projections, it suffers from singularities. In proximity to the poles, very small surface areas are mapped to many samples on the map, distributed over elongated areas. This causes significant storage and data access overhead in the renderer as well as a radial blur in the rendered image [10].

The problems associated with singularities can only be avoided by subdividing the sphere and using multiple maps. Kooima et al. use equidistant cylindrical projection for the equatorial part of the planet and two additional polar stereographic projections for the polar regions. Weighted averages are used for smooth transitions between the three regions [10].

Among polyhedral projections, the cube based approaches are very popular. They divide the spherical surface into six identical regions, as shown in Fig. 2.1. This allows using a single map projection (Fig. 2.2) that behaves consistently at cube face borders, thus eliminating the need for weighted averaging. Furthermore, the rectangular maps for the cube faces allow straightforward data storage using quadtree hierarchies and common file formats and straightforward data management in the rendering system using common and efficient rectangular textures.

The straightforward projection of the sphere to the cube faces is a gnomonic projection. The distortions introduced by gnomonic projection onto cube faces are significant. For this reason, Lerbour and et al. proposed an adjustment to the gnomonic projection [13] that reduces these distortions to some degree. Lambers and Kolb compared the gnomonic and adjusted gnomonic projections with the Quadrilateralized Spherical Cube (QSC) projection, and chose the latter [11]. As we will demonstrate in the next sections, their evaluation was too limited.

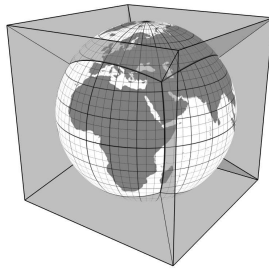


FIG. 2.1: A spherical planet model inscribed into a cube. The cube partitions the sphere surface into six equal areas.

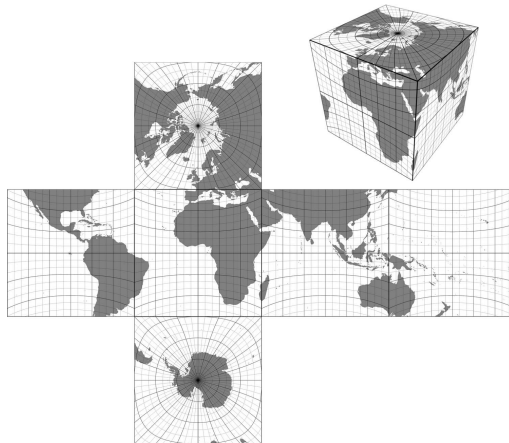


FIG. 2.2: The world mapped using a cube map projection.

In the application area of planet rendering, very few map projections have been considered for subdivisions of the planet. Of the cube-based subdivisions, only the gnomonic projection, an adjusted version of the gnomonic projection and the QSC projection are documented in the literature. In the next sections, we extend this list with a projection used in the Outerra rendering engine [8], an approximately equal-area projection based on the sphere representation in the Cartesian coordinates

[17] and a variant of the HEALPix projection [6]. All of these spherical cube map projections are explained in Sec. 4. and compared using the criteria derived in Sec. 3., according to the distortion they introduce in the texture application for planet-sized terrain rendering algorithms.

3. Evaluation Approach

In classical applications of map projections, the two-dimensional map is the final product and intended for direct use by the end user. In a planet rendering application, on the other hand, the map is just an intermediate data representation. Consequently, the projection is used in two steps: first, when mapping the original data to the cube-based hierarchical representation in a preprocessing step, and second, when sampling this representation during rendering to produce the end result. While a poor choice of map projection can have negative effects already during preprocessing, the crucial step for the quality of the end result is the rendering step.

We will, therefore, focus on rendering and sampling aspects of spherical cube map projections. To this end, we first examine, in Sec. 3.1., the way textures are applied to a rendered terrain. This discussion provides sufficient details to understand how texture filtering is performed and how effects of distortion can be reduced. In Sec. 3.2., we discuss how applied projections introduce a texture distortion, while Sec. 3.3. explains the methods used in the evaluation process. The main evaluation criterion for the projection comparison is texture distortion, but we also consider the precision and efficiency of the forward and inverse transformations, as well as the size of the applied textures.

3.1. Texture Application

Two-dimensional textures are image-overlays applied to geometrical objects to improve their fidelity without increasing their complexity. A texture application entails mapping from texture space to screen space. The mapping is done through two filtering schemes: *minification* and *magnification* [23]. When a texel (the smallest unit of a texture) is smaller than an area, it is applied to (one texel maps to multiple pixels on a screen) a magnification filter is used. Otherwise, multiple texels are mapped to a single pixel using a minification filter. In order to minimize the aliasing effect caused by minification, multiple levels of detail (the same texture in different resolutions) are used. That enables choosing the level where the texel-to-pixel ratio is near to one. Lance Williams proposed mipmaps as an efficient way to pack multiple levels of detail into a single texture [28]: each lower resolution level is constructed from the higher resolution level by downsampling with a factor of two in both horizontal and vertical directions.

Mipmaps have been used for decades as a very efficient way of texture mapping. As the size of the object being mapped increases, however, the application of high-fidelity mipmapped textures becomes untenable. The visualization of the planet

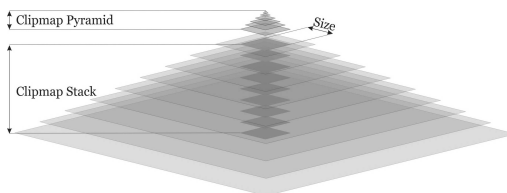


FIG. 3.1: A clipmap – an updatable partial mipmap. Clipmap levels are grouped into two sets: the clipmap pyramid (low-resolution static levels) and the clipmap stack (higher-resolution dynamic levels). The levels in the clipmap stack are centered on the focal point and toroidally updated as it moves.

Earth, with submeter accuracy, could require several petabytes of storage space for the full mipmapped texture; clearly this is too much to fit in the graphics or system memory. To solve this problem, several techniques have been developed. One of the most popular is known as *clipmapping* [26]. A clipmap is an updatable representation of a partial mipmap, in which each mipmap level is clipped to a specified size. Instead of the exponential growth of full mipmaps, a clipmap grows linearly with each new level of detail.

The appropriate level of clipmap to apply is chosen according to the texture scale-factors. The scale-factors of the applied texture are calculated using partial derivatives of the given functions $u(x, y)$ and $v(x, y)$ that map screen coordinates (x, y) to the two-dimensional texture coordinates s and t , respectively. Depending on the orientation of the surface, scale-factors along the horizontal and vertical screen axes (ρ_x and ρ_y , respectively) may differ.

$$\begin{aligned}
 \rho_x(x, y) &= \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2} \\
 \rho_y(x, y) &= \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} \\
 \rho_{max}(x, y) &= \max(\rho_x(x, y), \rho_y(x, y)) \\
 \rho_{min}(x, y) &= \min(\rho_x(x, y), \rho_y(x, y))
 \end{aligned}
 \tag{3.1}$$

Since derivatives may be computationally expensive and/or numerically unstable [9], they are usually approximated in the graphics hardware by computing forward/backward differences between neighboring pixels in a 2×2 block. Whether the forward or backward difference is used depends on the position of the pixel in the block. In standard texture filtering schemes, a proper level of detail (λ) is selected according to $\rho_{max}(x, y)$ which gives smoother results:

$$\lambda(x, y) = \log_2(\rho_{max}(x, y))
 \tag{3.2}$$

Further smoothing is achieved by linear interpolation of the texel values closest to the center of the displayed pixel. Bilinear filtering interpolates values of the texel from a single level only, selected by the rounded value of an integral part of λ . Trilinear filtering further improves the smoothness by combining two adjacent levels. The integral part of λ selects a more detailed level, while the fractional part defines blending factor with the next coarser one ($\lambda + 1$).

If the surface being viewed is at an oblique angle, trilinear filtering could result in a blurry display. The fidelity and sharpness of the applied texture, in that case, can be improved through anisotropic filtering [19]. Unlike the previous (isotropic) filtering schemes, where the footprint of the filter projection into texture space is a square, anisotropic filtering may have very narrow or long footprints. A higher degree of anisotropy may improve texture filtering quality, but at the same time reduce the texture filtering rate. Hence, the maximum degree of anisotropy (ϱ) is always limited, usually to 16. According to the OpenGL anisotropic texture filter specification [19], a proper texture level for anisotropic filtering should be selected using the following equations:

$$(3.3) \quad \begin{aligned} N(x, y) &= \min \left(\left\lceil \frac{\rho_{max}(x, y)}{\rho_{min}(x, y)} \right\rceil, \varrho \right) \\ \lambda(x, y) &= \log_2 \left(\frac{\rho_{max}(x, y)}{N(x, y)} \right) \end{aligned}$$

In the next section, we shall see how applied filtering is used to minimize the manifestation of texture distortion.

3.2. Distortion

Tissot's indicatrices are very useful in estimating the distortion of a projection of the Earth's surface to a planar map. They are used in cartography to evaluate the size and shape of the objects depicted on the map. However, in computer graphics the more important consideration is the distortion of the textures after application to a 3D model of the planet. In order to evaluate this texture application distortion we will introduce two measures of distortion:

- the texel aspect distortion and
- the texel area distortion.

The texel aspect distortion (δ_{aspect}) is defined as the texel width (Λ_x) to height (Λ_y) ratio after unprojecting to the surface of a planet:

$$(3.4) \quad \delta_{aspect} = \frac{\Lambda_x}{\Lambda_y}$$

In Eq. 3.4, Λ_x and Λ_y are calculated as distances on the spheroidal surface along directions aligned with the texture (i.e. projection) axes X and Y, respectively. The calculation is based on the central differences using the following equations:

$$\begin{aligned}
 (\varphi_l, \theta_l) &\leftarrow \text{inverse}(x - \Delta/2, y) \\
 (\varphi_r, \theta_r) &\leftarrow \text{inverse}(x + \Delta/2, y) \\
 (\varphi_b, \theta_b) &\leftarrow \text{inverse}(x, y - \Delta/2) \\
 (\varphi_t, \theta_t) &\leftarrow \text{inverse}(x, y + \Delta/2) \\
 \Lambda_x &= \sigma(\varphi_l, \theta_l, \varphi_r, \theta_r) \\
 \Lambda_y &= \sigma(\varphi_b, \theta_b, \varphi_t, \theta_t)
 \end{aligned}
 \tag{3.5}$$

In the previous equations, x and y are the coordinates of the point in texture space and Δ is a texel size. The function *inverse* depends on the chosen projection. Sec. 4. presents all spherical cube map projections with their forward transformation (from a sphere onto a plane, with normalized coordinates in the range [-1,1]) and inverse transformation (from the plane back to the sphere). A distance on the sphere between two points defined by their spherical coordinates (φ_1, θ_1) and (φ_2, θ_2) , is calculated with the function σ based on the following formulae:

$$\begin{aligned}
 \sigma(\varphi_1, \theta_1, \varphi_2, \theta_2) &= \\
 2R_e \arcsin \left(\sqrt{\sin^2 \frac{|\theta_1 - \theta_2|}{2} + \cos \theta_1 \cos \theta_2 \sin^2 \frac{|\varphi_1 - \varphi_2|}{2}} \right) \\
 R_e &= \frac{2 \cdot a + b}{3}
 \end{aligned}
 \tag{3.6}$$

Since the Earth is an oblate spheroid with very small flattening ($f = 1/298.257223563$), in order to simplify equations throughout this paper, we are using a spherical approximation with the same volume as the reference ellipsoid ($R_e = (a \cdot b)^{1/3} \approx (2 \cdot a + b)/3 = 6371km$). The parameters a and b in the previous equations refer to the semi-major and semi-minor axes of the WGS84 ellipsoid [16], respectively. For the ellipsoidal model of the planet, the distance between two points can be calculated more accurately using a Vincenty's inverse method [27]. However, the difference between the great circle distance formula (Eq. 3.6) and Vincenty's inverse formula in calculating distortion values is negligible. For example, in the case of QSC projection (Sec. 4.5.), the relative error of the maximum texel aspect distortion of the spherical approximation (compared to the WGS84 ellipsoidal model) is only about $5 \cdot 10^{-6}$. Therefore, the use of spherical approximations in the following discussion is justified.

Although the texel aspect distortion is usually neglected when a projection is chosen, the impact on the rendered surface can be significant and it manifests through:

- aliasing/blurring,
- additional anisotropy and
- a requirement for bigger textures.

The aliasing or blurring effects are consequences of shrinking or stretching of the texture over the applied surface. A texel distortion is always the combination of both aspect and area distortion. In order to simplify the analysis, without loss of generality, let us assume that distortion affects just a single direction (Fig. 3.2). If $\delta_{\text{aspect}} > 1$, stretching appears, while $\delta_{\text{aspect}} < 1$ causes texture shrinking. Depending on the texture scale-factors (Eq. 3.1), if the distortion is significant, even the current level of a clipmap (λ) may change. Without bilinear or trilinear filtering, the distortion results in a noticeable aliasing effect. Aliasing effects can be reduced by bilinear/trilinear filtering, but only to a certain degree and with an accompanying blur effect (Fig. 3.2).

Since aspect distortion introduces uneven texture sampling along different axes, a higher texture fidelity and sharpness can be achieved only by using anisotropic filtering (Fig. 3.2), but this introduces computational costs and is limited by the maximum anisotropy degree. As the surface is viewed from a more oblique angle, a higher degree of anisotropic filtering is required to provide sharpness. Because part of the available anisotropy is spent on aspect distortion correction, oblique surfaces may appear blurry.

Another reason for minimizing texel aspect distortion, even more important than the blur of oblique surfaces, is the need for bigger textures. Because of aspect distortion, after the application to a terrain, a texture changes its aspect and cover the different area along different directions, resulting in more details in the direction where shrinking occurs and fewer details in the direction where stretching occurs. If we select the texture level with enough details for the stretching direction while not taking into account aspect distortion, it will result in exceeding the size of a current clipmap level along the direction where shrinking occurs (black strips in Fig. 3.2). This issue can be solved by using texture levels bigger than their nominal size, for the factor greater or equal to the maximum texel distortion. Bigger textures induce higher memory consumption and longer update times. Choosing a coarser clipmap level, to avoid exceeding the size of the current level, leads to blurry rendering results.

The texel area distortion (δ_{area}) is defined as the ratio of the texel size at the current position ($\Lambda_x \cdot \Lambda_y$) and the texel size at the center of the cube face ($\Lambda_0 \cdot \Lambda_0$) after unprojecting to the surface of a planet:

$$(3.7) \quad \delta_{\text{area}} = \frac{\Lambda_x \Lambda_y}{\Lambda_0 \Lambda_0}$$

Unlike texel aspect distortion, the texel area distortion is important only if it changes across the surface of the current clipmap level λ . If δ_{area} is nearly constant, a distortion actually does not exist. The texel area distortion in non-equal-area

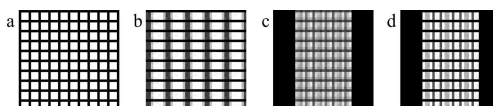


FIG. 3.2: Effects of the texel aspect distortion. From left to right: texture without distortion (a), horizontal stretching $\delta_{\text{aspect}} = 2$ (b), horizontal shrinking with trilinear filtering $\delta_{\text{aspect}} = 0.67$ (c), horizontal shrinking with anisotropic filtering $\delta_{\text{aspect}} = 0.67$ (d)

projection can be nearly constant only for the higher resolution levels. Such levels have smaller spatial extent, which prevents significant change in the value of area distortion. For the lower resolution levels, where it is not the case, area distortion has to be treated the same way as aspect distortion.

At the first glance, it seems that the texel area distortion does not require bigger textures, but only a modification of the clipmap level selection, defined by the following equation:

$$(3.8) \quad \lambda' = \lambda + \log_2(\sqrt{\delta_{\text{area}}})$$

However, since texel area distortion gradually changes across the surface and its value is usually not high enough to switch to the coarser clipmap level, texel area distortion also contributes to the need for larger textures. If the value of λ' is clamped to a higher integral value, a nominal size of the texture can be used, but the blurry outcome is inevitable. In order to preserve the ability to properly blend adjacent clipmap levels and gain required sharpness of the visualization, we have to provide levels big enough to contain nominal spatial extent, no matter where the viewer is located.

Considering both texel aspect and area distortion, the size of the storage space for a clipmap level should be bigger than its nominal size for the factor ε , where:

$$(3.9) \quad \varepsilon = \max(\delta_{\text{aspect}}^{\max}, \delta_{\text{area}}^{\max})$$

$\delta_{\text{aspect}}^{\max}$ and $\delta_{\text{area}}^{\max}$ are the maximum values of the texel aspect and area distortion, respectively.

In the conclusion, both texel aspect and area distortions result in higher texture storage demands and, hence, slightly slower update. The texel aspect distortion also spends a part of the available anisotropic filtering range. The higher aspect distortion the blurrier the display of a surface viewed from an oblique angle. The texel distortion elaborated in this section is one of the very important aspects that has to be evaluated when a proper projection for the terrain rendering system is chosen. The following section presents an evaluation method we used for comparing spherical cube map projections in this paper.

3.3. Evaluation Method

In order to examine their properties, we have implemented all spherical cube map projections that we could collect. Our development environment was the Microsoft Visual Studio 2013 and we used C/C++ compiler. All calculations are done on the CPU using double extended precision [7]. For each face of the spherical cube, the following tests were carried out:

- precision self-test,
- forward/inverse transformation time measurement,
- texel aspect distortion statistics, and
- texel area distortion statistics.

The precision self-test was used both for an internal implementation check and for checking the quality of the projection. For each of N^2 equally spaced points in the map space ($x, y \in [-1, 1]$), the inverse transformation is executed followed by forward transformations of its results. The final results should match the starting coordinates. The difference of the starting and final values projected to the surface of the Earth is used as a precision evaluation criterion.

Projection (forward transformation) and unprojection (inverse transformation) are required in data preparation and during terrain rendering, respectively, which is why the transformation execution time is an important property for evaluation. High-performance counters [14] are used for measuring the transformation of huge matrices with input coordinates. The measured time, besides the transformations themselves, includes iteration and matrix access time. Such overhead is required in order to prevent compiler optimizations from leading to incorrect results.

The texel aspect and area distortion are computed, using Eq. 3.4 through 3.7, for the points on the equally spaced grid in map space. The statistics discussed in this paper are computed for the set of up to 8000^2 points, with $\Delta = 1 \cdot 10^{-6}$.

In addition to the numerical tests, the same software was used to generate world maps in each of the projections, to reproject data from the equidistant cylindrical projection and to produce the images of texel aspect and area distribution over the faces of the cube. The results of the tests, as well as the generated images, are displayed and discussed in the following sections.

4. Spherical Cube Map Projections

This section presents a comprehensive set of Spherical Cube Map (SCM) projections. All projections are presented with their forward and inverse transformations, visual aspects of distortion shown through continents coastline and graticule skewing for the *front* and *top* faces, and the distribution of texel aspect and area distortions over the faces of the cube. The given formulae for forward and inverse

transformations apply to the cube face centered on $\varphi = 0$ and $\theta = 0$ (the *front* face). An exception is the HEALPix projection, where transformations differ for equatorial and polar regions, and, hence, have to be treated separately. Also, for each of the projections, the effects of the inverse transformation are visualized by reprojecting continent coastline and equally-spaced regular grid in the map space back to the globe.

4.1. Tangential Spherical Cube

Tangential Spherical Cube (TSC) is the simplest SCM projection. It uses the standard gnomonic projection to map the globe onto the six faces of a tangent cube. As a gnomonic projection, it is distortion-free only at the point where the tangent plane touches the surface. The distortion of shape, area and scale increases with the distance from that point. The gnomonic projection dates from Ancient Greece. Thales of Miletus (624-546 BC) used it for celestial maps. TSC was used for terrestrial maps in the beginning of the 19th century for the first time [24]. It is neither conformal nor equal-area projection.

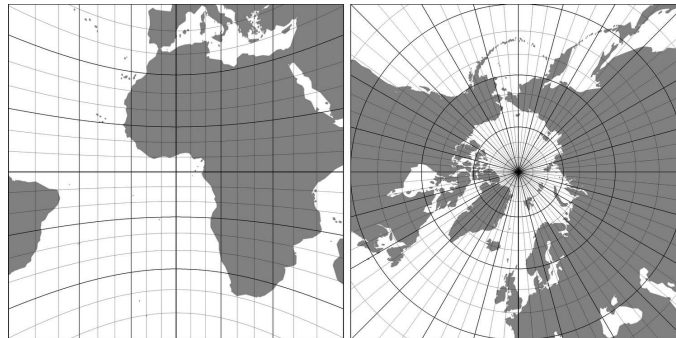


FIG. 4.1: Front and top faces of the TSC projection.

Forward transformation:

$$(4.1) \quad \begin{aligned} x &= \tan \varphi \\ y &= \frac{\tan \theta}{\cos \varphi} \end{aligned}$$

Inverse transformation:

$$(4.2) \quad \begin{aligned} \varphi &= \arctan x \\ \theta &= \arctan (y \cdot \cos \varphi) \end{aligned}$$

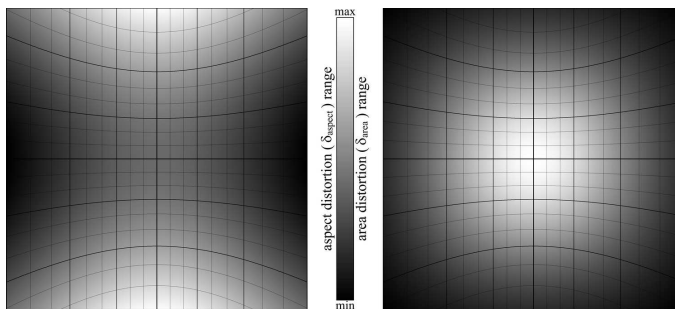


FIG. 4.2: The distribution of the texel aspect (left side) and area (right side) distortion over the face of the cube when TSC projection is used. The texel aspect distortion ranges from 0.707 to 1.414 ($\delta_{aspect}^{max} / \delta_{aspect}^{min} = 2.0$), while the texel area distortion ranges from 0.222 to 1.0 ($\delta_{area}^{max} / \delta_{area}^{min} = 4.5$).

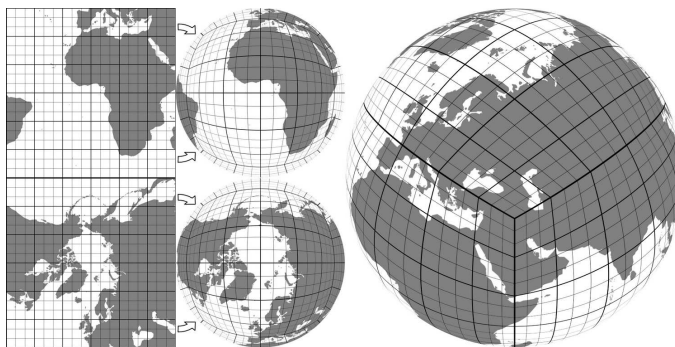


FIG. 4.3: The inverse transformation of the rectangular equidistant grid in TSC planar space and the continent coastlines to a globe surface.

Although it is simple, TSC is rarely used for visualization of the planet Earth, because of its significant distortions, both in aspect and area. Fig. 4.1 depicts a distortion through continent coastlines and graticule skewing, while Fig. 4.2 gives a spatial distribution of distortions over the faces of the cube through the shades of gray. Darker tones for the δ_{aspect} represent shrinking in the X-direction, while brighter tones represent shrinking in the Y-direction. Darker tones for δ_{area} repre-

sent an area shrinking. Texels at the corners of a face of the cube cover 4.5 times smaller surface than texels at the center. The effect of area shrinking can be explicitly visualized through the inverse transformation of the rectangular equidistant grid in the projection space to a globe surface (Fig. 4.3). While aspect distortion is slightly better than the equal-area or approximately equal-area SCM projections, area distortion is far worse.

4.2. Adjusted Spherical Cube

Adjusted Spherical Cube (ASC) modifies TSC in order to reduce area distortion. Instead of sampling the plane of projection, ASC samples the map directly in spherical coordinates with steps expressed in terms of angles [12, 13]. Thus, the forward transformation of TSC can be turned into ASC by simply calculating \arctan of the x and y coordinates and normalizing to ± 1 .

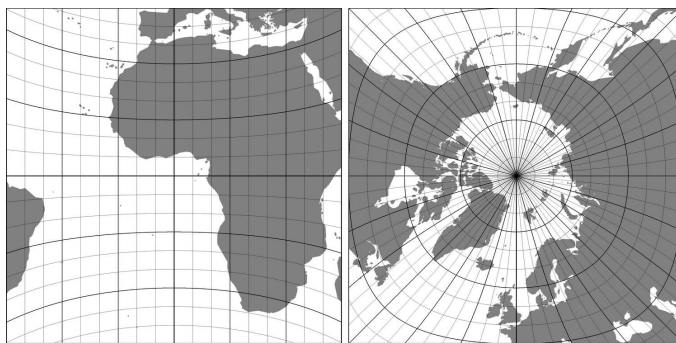


FIG. 4.4: Front and top faces of the ASC projection.

Forward transformation:

$$(4.3) \quad \begin{aligned} x &= \varphi \cdot \frac{4}{\pi} \\ y &= \arctan\left(\frac{\tan \theta}{\cos \varphi}\right) \cdot \frac{4}{\pi} \end{aligned}$$

Inverse transformation:

$$(4.4) \quad \begin{aligned} \varphi &= x \cdot \frac{\pi}{4} \\ \theta &= \arctan\left(\tan\left(\frac{\pi \cdot y}{4}\right) \cdot \cos \varphi\right) \end{aligned}$$

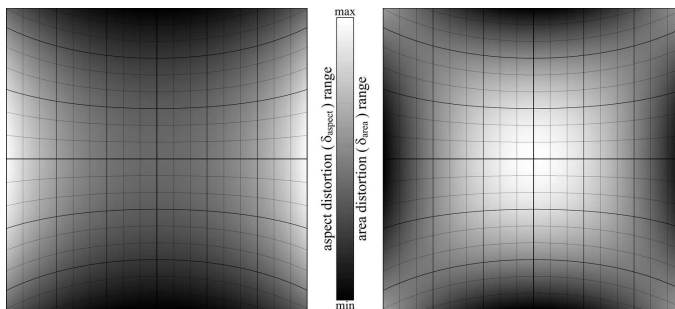


FIG. 4.5: The distribution of the aspect (left side) and area (right side) distortion over the face of the cube when ASC projection is used. The texel aspect distortion ranges from 0.707 to 1.414 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 2.0$), while the texel area distortion ranges from 0.707 to 1.0 ($\delta_{area}^{max}/\delta_{area}^{min} = 1.414$).

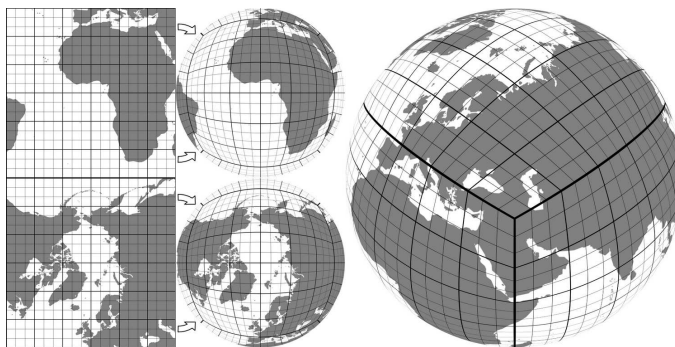


FIG. 4.6: The inverse transformation of the rectangular equidistant grid in ASC planar space and the continent coastlines to a globe surface.

ASC was published for the first time in 1996, with slightly different formulae based on colatitude [22]. It was reinvented and used for the planet-sized terrain rendering many years later, in 2009 [12]. Like TSC, ASC is neither conformal nor equal-area. However, the proposed adjustment of TSC effectively reduces area distortion. Graticule spacing increases toward the midpoints of the edges of the cube (Fig. 4.4), which indicates the texture area is shrinking. The distribution of

distortions is depicted in Fig. 4.5. Texel aspect distortion stays the same, except that the space the axes are inverted compared to TSC. Λ_x increases along the X-axis, while Λ_y increases along the Y-axis. Far better property of ASC, considering the area distortion, can be verify also by comparing Fig. 4.6 and Fig. 4.3.

4.3. Outerra Spherical Cube

Outerra Spherical Cube (OSC) is an SCM projection used in the Outerra planetary 3D engine [8]. The engine has the ability to render the whole planet with a full range of detail levels, from space down to individual blades of grass, and thus requires more uniform sampling than previous schemes. Unlike the other described projections, OSC does not have a closed form for the forward transformation. Hence, a Newton's iterative method is used in algorithm 1.

```

x = sin φ cos θ
y = sin θ
z = √(1 - x² - y²)
M = (1/(2√2 - 2) - 1) = 0.207106781
a = Mx²y²
b = -M(x² + y²)
c = -z
d = 1 + M
repeat
  F = az⁴ + bz² + cz + d
  F' = 4az³ + 2bz + c
  dF = F/F'
  z = z - dF
until |dF| < ε
x = zx
y = zy

```

Algorithm 1: OSC forward transformation

Inverse transformation:

$$\begin{aligned}
 M &= (1/(2\sqrt{2} - 2) - 1) = 0.207106781 \\
 z &= 1 + M(1 - x^2)(1 - y^2) \\
 l &= \sqrt{x^2 + y^2 + z^2} \\
 x &= x/l \\
 y &= y/l \\
 \varphi &= \arcsin\left(\frac{x}{\cos(\arcsin(y))}\right) \\
 \theta &= \arcsin(y)
 \end{aligned}
 \tag{4.5}$$

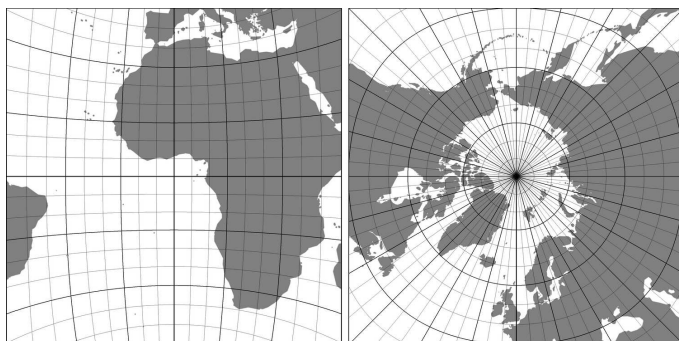


FIG. 4.7: Front and top faces of the OSC projection.

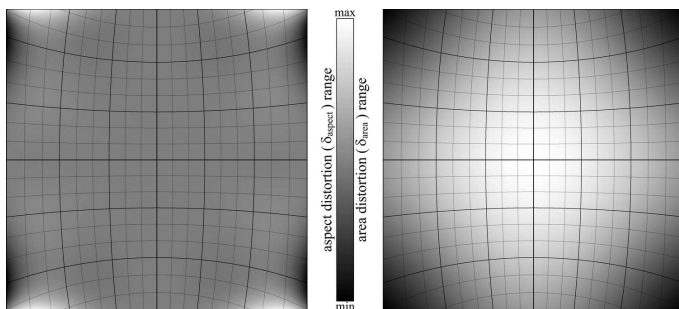


FIG. 4.8: The distribution of the aspect (left side) and area (right side) distortion over the face of the cube when OSC projection is used. The texel aspect distortion ranges from 0.934 to 1.006 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 1.013$), while the texel area distortion ranges from 0.324 to 1.0 ($\delta_{area}^{max}/\delta_{area}^{min} = 3.088$).

OSC has the least aspect distortion of all SCM projections described in this paper. There is another approach with no aspect distortion - a conformal SCM projection proposed in [21]. However, it is based on infinite Taylor series, and as such it is less suitable for our purpose.

The OSC texel area distortion (δ_{area}) is significant and it radially increases with the distance from the center of the cube face (Fig. 4.8). It is less than in the case of TSC, but much greater than for any other SCM projection.

Although the forward transformation requires iterations, the convergence is fast. With at most five iterations, a very high precision is achieved. The maximum error for the Earth-sized sphere, imposed by transformations, is less than 1mm if double extended precision is used.

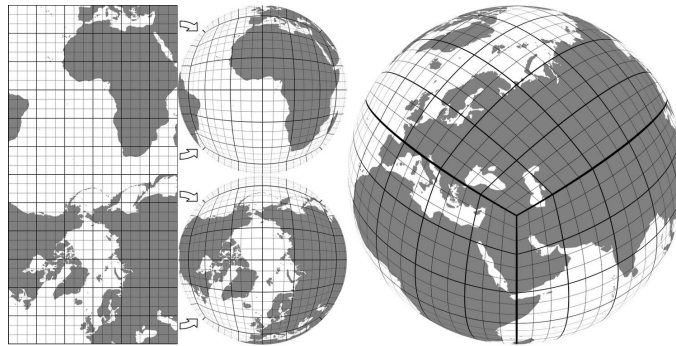


FIG. 4.9: The inverse transformation of the rectangular equidistant grid in OSC planar space and the continent coastlines to a globe surface.

4.4. COBE Quadrilateralized Spherical Cube

COBE quadrilateralized Spherical Cube (CSC) is an SCM projection based on research on the feasibility of a Quadrilateralized Spherical Cube (QLSC) Earth Data Base system, carried out in the early 1970s [3]. The purpose of the proposed projection was to minimize both area and shape distortion, and it was used, at least as it was reported, primarily in the U.S. Navy and later at NASA for the COsmic Background Explorer (COBE) project.

Forward transformation:

$$\begin{aligned}
 \tilde{x} &= \tan \varphi \\
 \tilde{y} &= \frac{\tan \theta}{\cos \varphi} \\
 x &= F(\tilde{x}, \tilde{y}) \\
 y &= F(\tilde{y}, \tilde{x})
 \end{aligned}
 \tag{4.6}$$

$$\begin{aligned}
 (4.7) \quad F(x, y) = & x\gamma + x^3(1 - \gamma) \\
 & + xy^2(1 - x^2) \left[\Gamma + (M - \Gamma)x^2 \right. \\
 & \left. + (1 - y^2) \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (C_{ij}x^{2i}y^{2j}) \right] \\
 & + x^3(1 - x^2) \left[\Omega - (1 - x^2) \sum_{i=0}^{\infty} D_i x^{2i} \right]
 \end{aligned}$$

However, QLSC has wrong forward transformation (or inverse transformation, if the notation from the original paper is used), which disqualifies it from any serious usage. That is the reason it is omitted from this paper, although it was a very important reference for all later spherical cube map studies. CSC was probably an effort to modify QLSC to be used in COBE project. As reported by Calabretta [1], the initial numeric parameters and equations derived in [3] were changed, as defined in the following formulae.

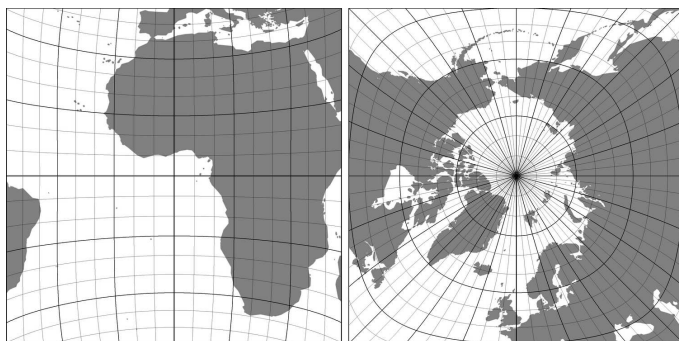


FIG. 4.10: Front and top faces of the CSC projection.

Instead of an infinite series, practical implementations use only a few terms. The following set of parameters is in use for the CSC:

$$\begin{aligned}
 (4.8) \quad & \gamma = 1.37484847732 & C_{00} &= 0.141189631152 \\
 & M = 0.004869491981 & C_{10} &= 0.0809701286525 \\
 & \Gamma = -0.13161671474 & C_{01} &= -0.281528535557 \\
 & \Omega = -0.159596235474 & C_{20} &= -0.178251207466 \\
 & D_0 = 0.0759196200467 & C_{11} &= 0.15384112876 \\
 & D_1 = -0.0217762490699 & C_{02} &= 0.106959469314
 \end{aligned}$$

Inverse transformation:

$$\begin{aligned}
 \tilde{x} &= I(x, y) \\
 \tilde{y} &= I(y, x) \\
 (4.9) \quad I(x, y) &= x + x(1-x^2) \sum_{j=0}^N \sum_{i=0}^{N-j} P_{ij} x^{2i} y^{2j} \\
 \varphi &= \arctan \tilde{x} \\
 \theta &= \arctan(\tilde{y} \cdot \cos \varphi)
 \end{aligned}$$

The COBE implementation confines N to 6 and uses the following set of best-fitting values for coefficients P_{ij} :

$$\begin{aligned}
 (4.10) \quad & P_{00} = -0.27292696 & P_{12} = -0.56800938 & P_{30} = 0.54852384 \\
 & P_{01} = -0.02819452 & P_{13} = 1.50880086 & P_{31} = -1.74114454 \\
 & P_{02} = 0.27058160 & P_{14} = -1.41601920 & P_{32} = 0.98938102 \\
 & P_{03} = -0.60441560 & P_{15} = 0.52032238 & P_{33} = 0.08693841 \\
 & P_{04} = 0.93412077 & P_{20} = -0.22797056 & P_{40} = -0.62930065 \\
 & P_{05} = -0.63915306 & P_{21} = 0.48051509 & P_{41} = 1.71547508 \\
 & P_{06} = 0.14381585 & P_{22} = 0.30803317 & P_{42} = -0.83180469 \\
 & P_{10} = -0.07629969 & P_{23} = -0.93678576 & P_{50} = 0.25795794 \\
 & P_{11} = -0.01471565 & P_{24} = 0.33887446 & P_{51} = -0.53022337 \\
 & P_{60} = 0.02584375 & &
 \end{aligned}$$

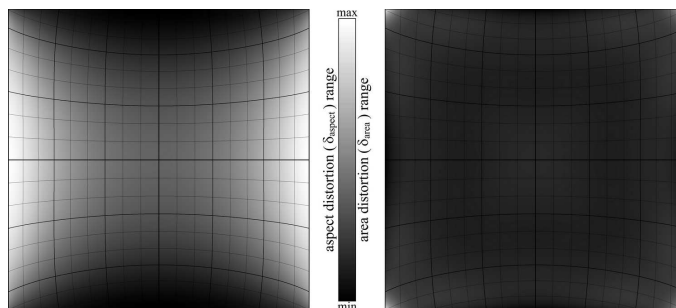


FIG. 4.11: The distribution of the aspect (left side) and area (right side) distortion over the face of the cube when CSC projection is used. The texel aspect distortion ranges from 0.65 to 1.538 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 2.365$), while the texel area distortion ranges from 0.94 to 1.325 ($\delta_{area}^{max}/\delta_{area}^{min} = 1.41$).

CSC is an approximately equal-area projection with $\delta_{area} \in [0.94, 1.32]$. The ratio of maximum and minimum texel area distortions is the same as for ASC

($\delta_{\text{area}}^{\text{max}}/\delta_{\text{area}}^{\text{min}} = 1.41$), but the distribution of values is much better ($\text{RMSD}(\delta_{\text{area}} = 0.019)$). The texel aspect distortion is worse than all previously mentioned projections ($\delta_{\text{aspect}} \in [0.65, 1.54]$).

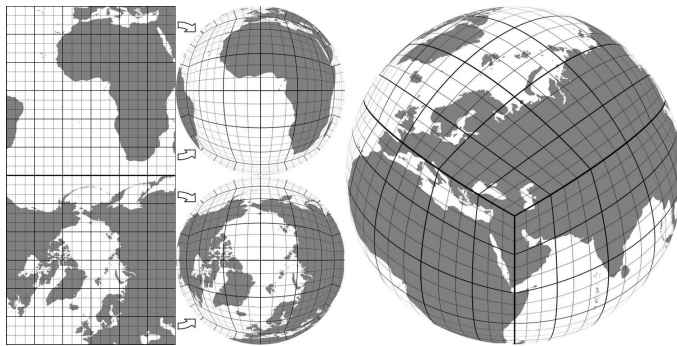


FIG. 4.12: The inverse transformation of the rectangular equidistant grid in CSC planar space and the continent coastlines to a globe surface.

Worse than the texel aspect distortion is the imprecision of the projection. The approximation imposed by using a finite number of terms and best-fitting coefficients in Eq. 4.6 and 4.9 causes a significant error in positioning. After successive inverse and forward transformations, the maximum absolute error of the position on the equivalent sphere is 1.39km. Moreover, CSC stretches an area toward the cube edges. It can be clearly seen in Fig. 4.12. The middle part of Europe almost disappeared at the edge of the cube, while Greenland is much wider than it should be. All these properties disqualify CSC for the application in any geographic information system and, hence, in SCM also.

4.5. Quadrilateralized Spherical Cube

Quadrilateralized Spherical Cube (QSC) is another SCM projection based on the work of Chan and O'Neill [3], described in Sec. 4.4. O'Neill and Laubscher [20] defined an equal-area projection to map the sphere surface to a cube face with the purpose of storing data in hierarchical structures for each cube face. In addition to being equal-area, the QSC projection was designed to limit angular distortions.

Forward transformation:

$$\begin{aligned}
 q &= \cos(\theta) \cos(\varphi) \\
 r &= \cos(\theta) \sin(\varphi) \\
 s &= \sin(\theta) \\
 \tilde{\varphi} &= \arccos(q) \\
 \tilde{\theta} &= \arctan(s, r) \\
 (4.11) \quad \mu &= \arctan\left(\frac{12}{\pi}\right) \cdot \left(\tilde{\theta} + \arccos(\sin(\tilde{\theta}) \cos(\frac{\pi}{4})) - \frac{\pi}{2}\right) \\
 \nu &= \arctan\left(\sqrt{\frac{1 - \cos(\tilde{\varphi})}{\cos^2(\mu) \cdot (1 - \cos(\arctan(1/\cos(\tilde{\theta}))))}}\right) \\
 x &= \tan(\nu) \cos(\mu) \\
 y &= \tan(\nu) \sin(\mu)
 \end{aligned}$$

Inverse transformation:

$$\begin{aligned}
 \nu &= \arctan(\sqrt{x^2 + y^2}) \\
 \mu &= \arctan\left(\frac{y}{x}\right) \\
 t &= \frac{\pi}{12} \tan(\mu) \\
 \tilde{\theta} &= \arctan\left(\frac{\sin(t)}{\cos(t) - 1/\sqrt{2}}\right) \\
 (4.12) \quad \tilde{\varphi} &= \arccos(1 - \cos^2(\mu) \tan^2(\nu) (1 - \cos(\arctan(1/\cos(\tilde{\theta})))))) \\
 q &= \cos(\tilde{\varphi}) \\
 s &= \sqrt{1 - q^2} \sin(\tilde{\theta}) \\
 r &= \sqrt{1 - q^2 - s^2} \\
 \theta &= \arccos(-s) - \frac{\pi}{2} \\
 \varphi &= \arctan\left(\frac{r}{q}\right)
 \end{aligned}$$

The formulae for forward and inverse transformations, given above, apply to one-quarter of the front cube face; the other three-quarters are handled by rotating this definition. This is done by first determining the interval of $\tilde{\theta}$, which defines the quarter, then shifting $\tilde{\theta}$ to the interval of definition $[-\frac{\pi}{4}, \frac{\pi}{4}]$ by adding or subtracting a multiple of $\frac{\pi}{2}$, then computing μ as described, and finally shifting μ back to the original quarter by again adding or subtracting a multiple of $\frac{\pi}{2}$. Furthermore, other cube faces than the front cube face are handled by adapting the computation of $\tilde{\theta}$, e.g. $\tilde{\theta} = \arctan(s, -q)$ for the appropriate cube face.

As can be seen in Fig. 4.13, QSC suffers from significant shape distortion. Furthermore, there are discontinuities at the $x = |y|$ directions (diagonals of the cube

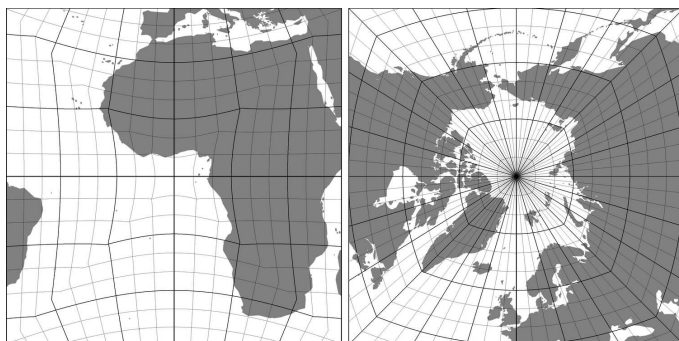


FIG. 4.13: Front and top faces of the QSC projection.

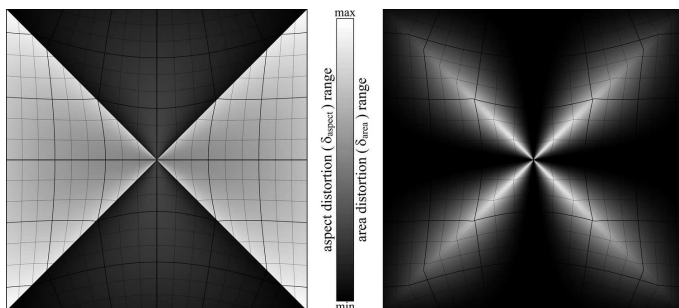


FIG. 4.14: The distribution of the aspect (left side) and area (right side) distortion over the face of the cube when QSC projection is used. The texel aspect distortion ranges from 0.649 to 1.539 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 2.37$), while the texel area distortion ranges from 0.89 to 0.93 ($\delta_{area}^{max}/\delta_{area}^{min} = 1.042$).

faces). These discontinuities change the direction of aspect distortion (which is also at its maximum), cause severe additional texture distortion if intersecting triangles of the underlying mesh and also slightly disturb the equal-area property of the projection. The left side of Fig. 4.16 depicts a distortion caused by intersecting triangles, while the underlying mesh is shown on the right side of the Fig. 4.16. This issue can be solved by splitting the cube faces into four triangular regions, using a very fine tessellation (a pixel-sized triangles) or ray casting rendering (per

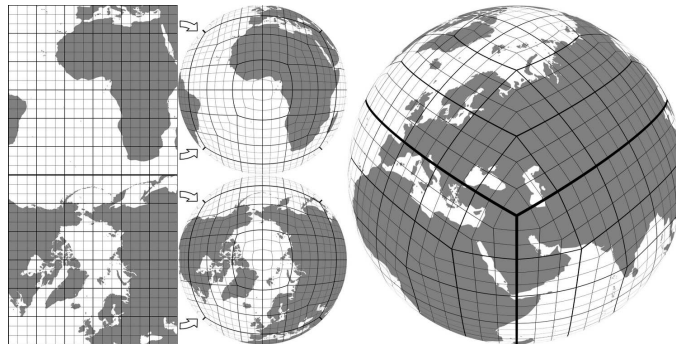


FIG. 4.15: The inverse transformation of the rectangular equidistant grid in QSC planar space and the continent coastlines to a globe surface.

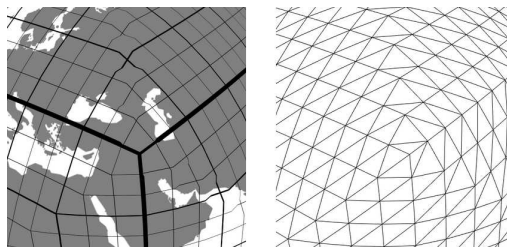


FIG. 4.16: Discontinuities at the cube face diagonals can cause severe texture distortion if they intersect triangles of the textured mesh. Diagonals toward the pole and India ripple the grid, while the diagonal toward the center of Africa does not, since the line of discontinuity is aligned with triangles' edges.

pixel texture sampling).

4.6. Cartesian Spherical Cube

Cartesian Spherical Cube (KSC) emerged from mapping a square to a circle [18] and its generalization to mapping a cube to a sphere [17], proposed by Philip Nowell. The proposed mapping is actually the inverse transformation in a closed form with coordinates defined in Cartesian coordinate system (Eq. 4.15). The initial in-

tent was not to provide a cartographic mapping, but just a sphere parametrization example posted on the Web blog [17]. However, some nice properties of this transformation yielded an implementation of the forward transformation [25] five years later. Prior to the forward transformation (Eq. 4.14), the polar coordinates have to be transformed to the Cartesian coordinates (Eq. 4.13) and the cube face has to be determined (the third row in the Tab. 4.1) according to one-sixth the maximum of all three Cartesian coordinates (the first row in the Tab. 4.1) and its sign (the second row in the Tab. 4.1). Depending on the cube face, input variables for the transformation have to be rearranged (the fourth row in the Tab. 4.1), so that each face can be transformed to the front face of the cube.

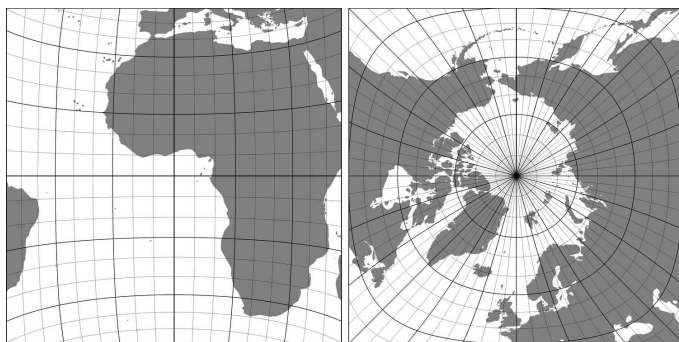


FIG. 4.17: Front and top faces of the KSC projection.

$(x \geq y) \wedge (x \geq z)$		$(y \geq x) \wedge (y \geq z)$		$(z \geq x) \wedge (z \geq y)$	
$x > 0$	$x < 0$	$y > 0$	$y < 0$	$z > 0$	$z < 0$
Face 1 (right)	Face 3 (left)	Face 4 (top)	Face 5 (bottom)	Face 0 (front)	Face 2 (back)
$x' = -z$	$x' = z$	$y' = -z$	$y' = z$		$x' = -x$
$z' = x$	$z' = x$	$z' = y$	$z' = y$		

Table 4.1: Face selection and mapping to a front face before KSC forward transformation.

Forward transformation:

$$\begin{aligned}
 \chi &= \cos \theta \sin \varphi \\
 \psi &= \sin \theta \\
 \zeta &= \cos \theta \cos \varphi
 \end{aligned}
 \tag{4.13}$$

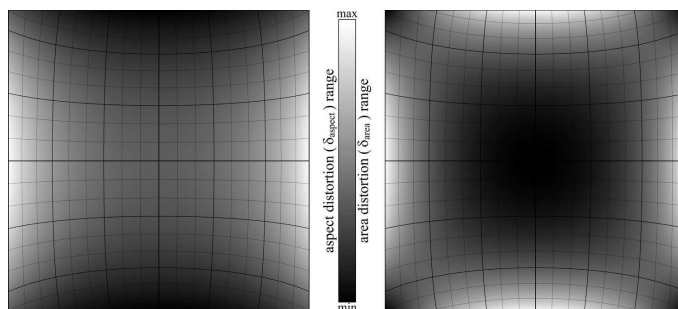


FIG. 4.18: The distribution of the texel aspect (left side) and area (right side) distortion over the face of the cube when KSC projection is used. The texel aspect distortion ranges from 0.577 to 1.731 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 3.0$), while the texel area distortion ranges from 1.0 to 1.155 ($\delta_{area}^{max}/\delta_{area}^{min} = 1.155$).

$$\begin{aligned}
 \xi &= -\sqrt{(2\psi^2 - 2\chi^2 - 3)^2 - 24\chi^2} \\
 x &= \text{sign}(\chi) \cdot \min\left(\sqrt{\frac{\max(\xi + 2\chi^2 - 2\psi^2 + 3, 0)}{2}}, 1\right) \\
 y &= \text{sign}(\psi) \cdot \min\left(\sqrt{\frac{\max(\xi - 2\chi^2 + 2\psi^2 + 3, 0)}{2}}, 1\right) \\
 z &= \text{sign}(\zeta)
 \end{aligned}
 \tag{4.14}$$

Inverse transformation:

$$\begin{aligned}
 x' &= x \cdot \sqrt{\max(1 - \frac{1}{2}y^2 - \frac{1}{2}z^2 + \frac{1}{3}y^2z^2, 0)} \\
 y' &= y \cdot \sqrt{\max(1 - \frac{1}{2}x^2 - \frac{1}{2}z^2 + \frac{1}{3}x^2z^2, 0)} \\
 z' &= z \cdot \sqrt{\max(1 - \frac{1}{2}x^2 - \frac{1}{2}y^2 + \frac{1}{3}x^2y^2, 0)}
 \end{aligned}
 \tag{4.15}$$

$$\begin{aligned}
 \varphi &= \arctan \frac{x'}{z'} \\
 \theta &= \arcsin y'
 \end{aligned}
 \tag{4.16}$$

KSC is an approximately equal-area SCM projection (15% deviation), with a severe texel aspect distortion. It has worse aspect distortion than any other previously mentioned SCM. Another unusual property of the KSC is both shrinking and

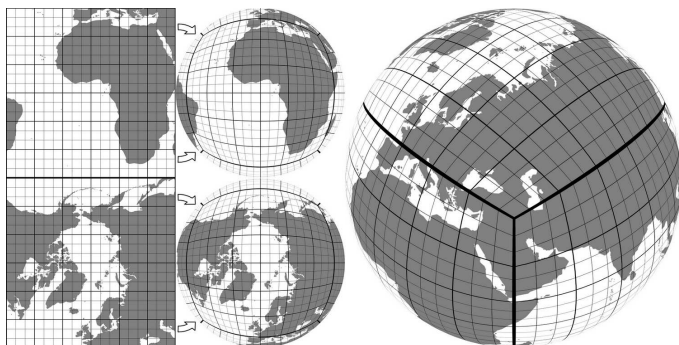


FIG. 4.19: The inverse transformation of the rectangular equidistant grid in KSC planar space and the continent coastlines to a globe surface.

stretching of the texels at the same time along different axes, while moving from the center of the cube face toward the middle of the edges. The texel aspect is constant at the diagonals.

4.7. Hierarchical Equal Area isoLatitude Pixelization

Hierarchical Equal Area isoLatitude Pixelization (HEALPix) [6] is a class of spherical projections with a property of distributing $12N^2$ points as uniformly as possible over the surface of the unit sphere. These hybrid projections combine the Lambert cylindrical equal-area projection, for the equatorial region with the interrupted Collignon projection for the polar regions. This infinite class of projections is parameterized by N_θ and N_ϕ [6] (often referred to as K and H, respectively [2]). N_θ is the number of base-resolution pixel layers between the north and south poles and N_ϕ is the multiplicity of the meridional cuts, or the number of equatorial or circumpolar base-resolution pixels. In this paper, we discuss only the HEALPix projection with $N_\theta = 3$ and $N_\phi = 4$ (Fig. 4.20), since it is the only projection of the whole class that can be rearranged to a cube-based hexahedral projection.

The equatorial and the polar regions meet at latitude $\hat{\theta}$. This particular latitude can be calculated based on the fact that the polar region, as a part of an equal-area projection, needs to be one-sixth of the total area, and that the area of the spherical cap can be calculated as $P = 2\pi(1 - \sin \theta)$ [2]. Hence, $\hat{\theta} = \arcsin(2/3) \approx 41.81^\circ$.

Since the projection differs for the equatorial and polar regions, we will provide forward and inverse transformations separately. The forward transformation of the

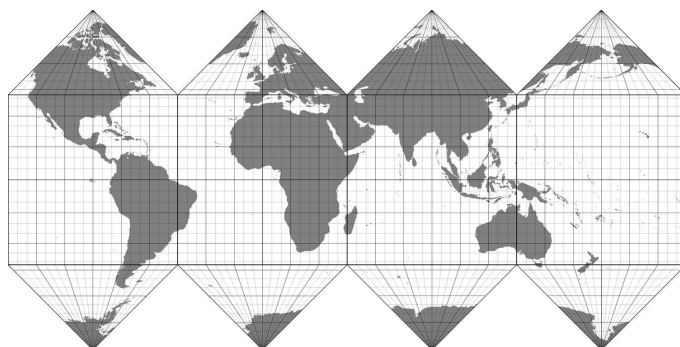


FIG. 4.20: The world map based on HEALPix projection with $N_\theta = 3$ and $N_\phi = 4$.

equatorial region ($|\theta| < \bar{\theta}$) is:

$$(4.17) \quad \begin{aligned} x &= \varphi \cdot \frac{4}{\pi} \\ y &= \frac{3}{2} \cdot \sin \theta \end{aligned}$$

Inverse transformation for equatorial region:

$$(4.18) \quad \begin{aligned} \varphi &= x \cdot \frac{\pi}{4} \\ \theta &= \arcsin \frac{2y}{3} \end{aligned}$$

As it can be seen in the Fig. 4.22, HEALPix is an exact equal-area projection for the equatorial region. Also, the texel aspect distortion (δ_{aspect}) is less than for any other SCM projection, except for OSC.

HEALPix projections are not cube-based, but the four triangles of the interrupted Collignon projection for the polar regions can be rearranged and grouped to form a face of the cube. The process of rearrangement after the forward transformation is summarized in Tab. 4.2. This process should be reversed in the inverse transformation.

Forward transformation for polar regions:

$$(4.19) \quad \begin{aligned} \sigma &= \sqrt{3(1 - |\sin \theta|)} \\ x &= \sigma \varphi \cdot \frac{4}{\pi} \\ y &= 1 - \sigma \end{aligned}$$

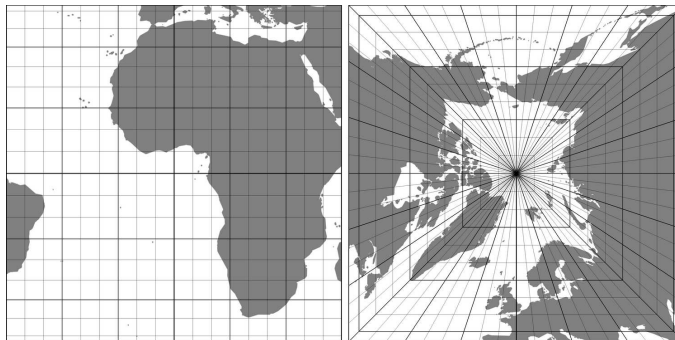


FIG. 4.21: Front and top faces of the HEALPix ($N_\theta = 3$ and $N_\phi = 4$) projection.

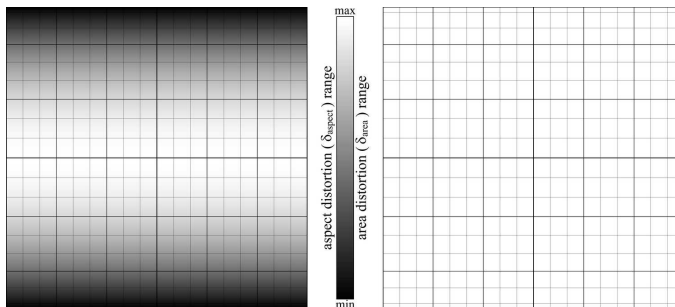


FIG. 4.22: . The distribution of the aspect (left side) and area (right side) distortion over the equatorial face of the cube when HEALPix ($N_\theta = 3$ and $N_\phi = 4$) projection is used. The texel aspect distortion ranges from 0.654 to 1.178 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 1.8$). There is no area distortion.

Inverse transformation for polar regions:

$$(4.20) \quad \begin{aligned} \sigma &= 1 - y \\ \varphi &= \frac{x}{\sigma} \frac{\pi}{4} \\ \theta &= \pm \arcsin \left(1 - \frac{\sigma^2}{3} \right) \end{aligned}$$

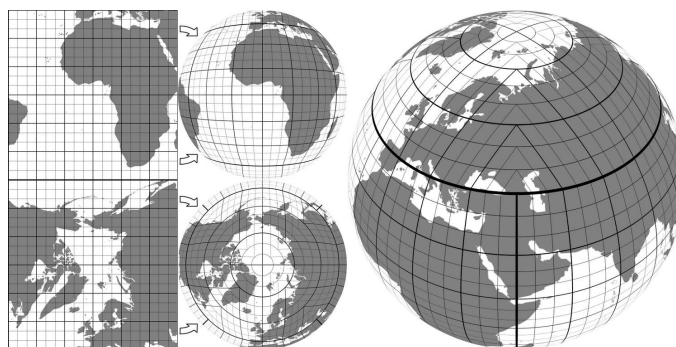


FIG. 4.23: The inverse transformation of the rectangular equidistant grid in HEALPix ($N_\theta = 3$ and $N_\phi = 4$) planar space and the continent coastlines to a globe surface.

φ	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[\frac{\pi}{4}, \frac{3\pi}{4}]$	$[\frac{3\pi}{4}, \frac{5\pi}{4}]$	$[\frac{5\pi}{4}, \frac{7\pi}{4}]$
$\theta > \tilde{\theta}$	$y = y-1$	$z = x$ $x = 1-y$ $y = z$	$x = -x$ $y = 1-y$	$z = x$ $x = y-1$ $y = -z$
$\theta < -\tilde{\theta}$	$y = 1-y$	$z = x$ $x = 1-y$ $y = -z$	$x = -x$ $y = y-1$	$z = x$ $x = y-1$ $y = z$

Table 4.2: Rearrangement of interrupted Collignon projection's triangles into a quad after the forward transformation.

Subface	$y \leq x $	$-x \leq y \leq x$	$y \geq x $	$-y \leq x \leq y$
North polar face	$y = 1+y$	$z = y$ $y = 1-x$ $x = z$	$x = -x$ $y = 1-y$	$z = y$ $y = 1+x$ $x = -z$
South polar face	$x = -x$ $y = 1+y$	$z = y$ $y = 1+x$ $x = z$	$y = 1-y$	$z = y$ $y = 1-x$ $x = -z$

Table 4.3: Splitting the polar quads into interrupted Collignon projection's triangles before the inverse transformation.

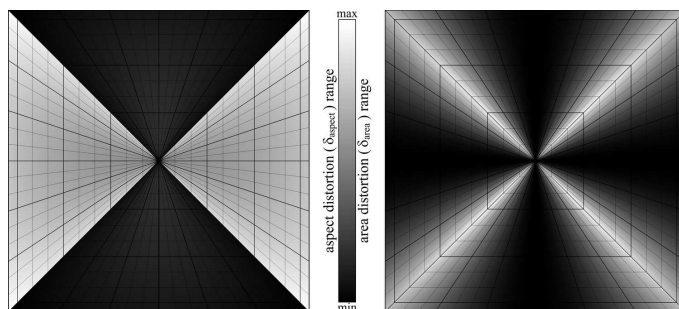


FIG. 4.24: The distribution of the aspect (left side) and area (right side) distortion over the polar face of the cube when HEALPix projection is used. The texel aspect distortion ranges from 0.548 to 1.826 ($\delta_{aspect}^{max}/\delta_{aspect}^{min} = 3.33$), while the texel area distortion ranges from 1.0 to 1.27 ($\delta_{area}^{max}/\delta_{area}^{min} = 1.27$).

Texel aspect distortion is very high in the polar regions (left side of Fig. 4.24). It has the highest value of all described projections, and it can be clearly seen on the graticule and continents shape (right side of the Fig. 4.21). Like in QSC, there are discontinuities at $x = |y|$ directions on the polar faces of the cube, with even higher magnitude. These discontinuities cause the same problems as with QSC, like further increase of the aspect distortion (it is at maximum and change direction at the line of discontinuity), texture rippling over triangles that intersect discontinuity and also disturbing the equal-area property (right side of the Fig. 4.24).

5. Spherical Cube Map Projection Comparison

In this section, we summarize the characteristics of the projections, covered in the previous sections, and present a side-by-side comparison according to the tests described in Sec. 3.

Although precision was not the main criteria used in SCM projection evaluation, poor precision can certainly limit a projections field of application. For example, precision is very important in location services, cadastral surveys and geographic information systems. Almost all SCM projections maintained good precision, with the error introduced by an inverse transformation followed by a forward transformation less than or equal to $1\mu m$ for an Earth-sized planet. The only exception is CSC (and also QLSC, but this projection is not covered in the paper). Because of its imprecision and tendency to stretch the surface toward the edges of the cube, CSC cannot be used for Earth mapping.

Another property used in the evaluation process is the execution time of both forward and inverse transformations. Execution time significantly depends on implementation optimization, compiler, CPU architecture, working frequency, caching scheme, etc. Therefore, Fig. 5.1 depicts normalized values. The normalization is done using the shortest execution time. The tests were executed on Intel Core i7-4700HQ CPU using the Microsoft Visual Studio 2013 C++ compiler on the Microsoft Windows 8.1 operating system. As can be seen in Fig. 5.1, HEALPix is the fastest projection, while OSC and QSC are the slowest ones. The OSC forward transformation, due to its iterative nature, has a relatively long execution time, but even so it has approximately the same speed as QSC. On the other hand, the inverse transformation of OSC is much faster than QSC. KSC has approximately the same execution time for both transformations. Generally, excluding HEALPix, all other SCM projections have the same order of magnitude execution time.

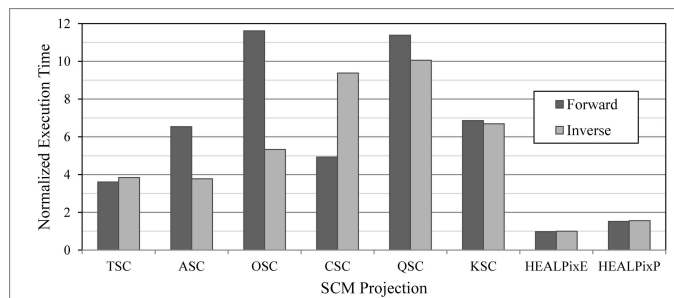


FIG. 5.1: Forward and inverse transformation execution time comparison. Values are normalized using HEALPix equatorial inverse transformation time.

The most important property used in evaluating the quality of SCM projections is distortion. Table 5.1 gives a comparative review of relevant distortion parameters: minimum (*min*) and maximum (*max*) values, maximum-to-minimum ratio (*ratio*) and root-mean-square deviation (*RMSD*) of both texel aspect and area distortion. HEALPix for the polar regions and KSC have the worst aspect distortion. QSC has a significant texel aspect distortion, while TSC and ASC are slightly better. HEALPix and QSC are actually not hexahedral projection. In order to reduce the effects of discontinuities, faces of the cube have to be divided into, at least, four triangular areas with edges aligned with discontinuities. OSC is the only SCM projection that eliminates aspect distortion.

TSC has the worst texel area distortion. OSC is better than TSC, but still has significant area distortion at the corners of the cube faces. ASC has a relatively low area distortion, while KSC can be considered approximately equal-area projection. QSC is classified as an equal-area projection, although the discontinuities slightly

disturb that property. Furthermore, in the tests of this property, we have compared texels all over the cube face with the texel in its center. However, the central texel in the QSC projection crosses both lines of discontinuities, hence its size is bigger than any other texel in the map. That is the reason the maximal value of the QSC area distortion is less than one. The even number of texels is assumed along each axis, which eliminates the extreme value and gives a little better overall equal-area property of QSC. HEALPix for the equatorial region is an exact equal-area projection, while the polar regions suffer from discontinuities the same way as QSC, but even with a higher magnitude.

Projection	δ_{aspect}				δ_{area}			
	min	max	ratio	RMSD	min	max	ratio	RMSD
TSC	0.707	1.414	2.000	0.155	0.222	1.000	4.500	0.506
ASC	0.707	1.414	2.000	0.146	0.707	1.000	1.414	0.153
OSC	0.994	1.006	1.013	0.001	0.324	1.000	3.088	0.280
CSC	0.650	1.538	2.365	0.218	0.940	1.325	1.410	0.019
QSC	0.650	1.539	2.369	0.271	0.894	0.931	1.042	0.099
KSC	0.577	1.732	3.000	0.227	1.000	1.155	1.155	0.063
HEALPix ^E	0.654	1.178	1.800	0.156	1.000	1.000	1.000	0.000
HEALPix ^P	0.548	1.826	3.334	0.437	1.000	1.272	1.272	0.108

Table 5.1: Comparative review of SCM projections relevant distortion parameters: minimum (*min*) and maximum (*max*) values, maximum-to-minimum ratio (*ratio*) and root-mean-square deviation (*RMSD*) of both texel aspect and area distortion.

Even though a tabular review is useful for comparing values, a visual representation is usually more convincing. Fig. 6.1 gives a side-by-side comparison of all evaluated SCM projections. A comparison is done through topographic view with the graticule, texel aspect distortion and texel area distortion distribution over the face of the cube. Since projections may differ in the equatorial and polar regions, both equatorial (front) and polar (top) faces are provided.

Fig. 6.2 compares the inverse transformation effects by reprojecting regular grids from the projection planes back to the spherical surface. The figure reveals how the grid is distorted and also the issues with CSC projection. It can be clearly seen that the shapes of the continents are incorrect in the case of CSC. CSC was used for mapping cosmic background radiation, where the equal-area property was important in representing its density, while the distortion effects were of secondary importance if they were relevant at all or even noticed.

6. Conclusion

A spherical surface cannot be mapped to a plane without distortion. If a projection preserves shapes, it does not preserve area and vice versa. The choice of map projection must therefore always consider the requirements of the application area.

For planet-sized terrain rendering, projections from the sphere onto the six faces of a cube are of particular interest, since the rectangular cube faces allow use of existing file formats for image and data storage, textures with mipmap and anisotropic filtering capabilities as are typical in graphics pipelines, and quadtree hierarchies and clipmaps are commonly used for level-of-detail purposes.

The rendering stage, in particular the texture sampling stage in modern graphics pipelines, dictates the quality of the rendering result of such applications. We, therefore, derive the following quality criteria for the evaluation of map projections: texel aspect distortions, texel area distortions, and efficiency of transformations required for texture sampling.

Both texel aspect and area distortion increase texture size required for the certain level of fidelity, while texel aspect distortion also spends a certain amount of the hardware supported anisotropy range. The efficiency of the transformations directly dictates the time needed for the data preparation and the rendering itself.

Using these criteria, we evaluated a comprehensive list of suitable SCM projections. Each projection has its advantages and disadvantages. A few projections are clearly unsuitable for the task; QLSC because of the wrong forward transformation and CSC because of imprecision and distortion that cannot be corrected by texture filtering. Among other presented projections, both QSC and HEALPix are actually not hexahedral and they introduce discontinuities if they are treated as such. Also, both QSC and HEALPix introduce high texel aspect distortion.

Comparing all presented SCM projections, ASC is probably the best choice, combining easy implementation, relatively fast transformation and moderate distortion. ASC requires the texture storage space twice the size of the nominal value. OSC enables sharper rendering than any other SCM, since the available anisotropy filtering range is not spent on the aspect distortion. However, OSC may require bigger texture storage space (about 54% more than ASC, if the same texture levels blending scheme is used). Also, OSC has slower forward transformation than ASC.

KSC requires approximately the same texture size as OSC, but results in the blurrier rendering of flat, nearly horizontal surfaces, because of spending a significant range of available anisotropic filtering for correcting aspect distortion. TSC is the worst choice considering required texture size (2.25 times more than ASC). Although the texel aspect distortion of the TSC is the same as of ASC, the texel area distortion is far worse than any other SCM projection.

Among the vast number of known projections (and the infinite number of as yet unknown projections), there are certainly more that can be applied to map a sphere to cube faces. Future work will, therefore, include the search for more projections, and their evaluation. Since the particular needs of the planet-sized terrain rendering were not considered when constructing most known map projections, it is possible that a projection specifically created for this task could be superior to the projections evaluated in this paper.

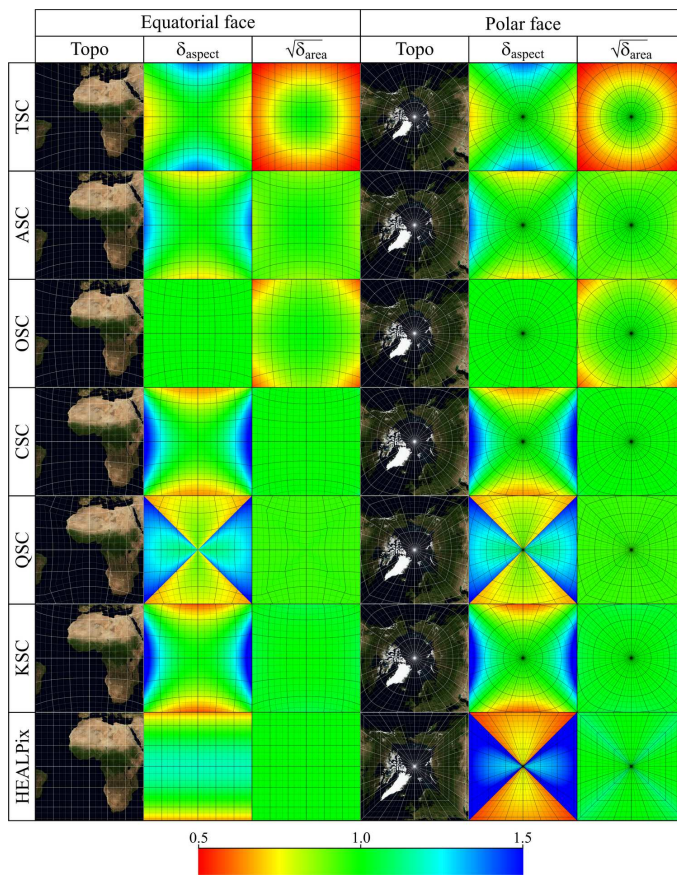


FIG. 6.1: Side-by-side comparison of all evaluated SCM projections.

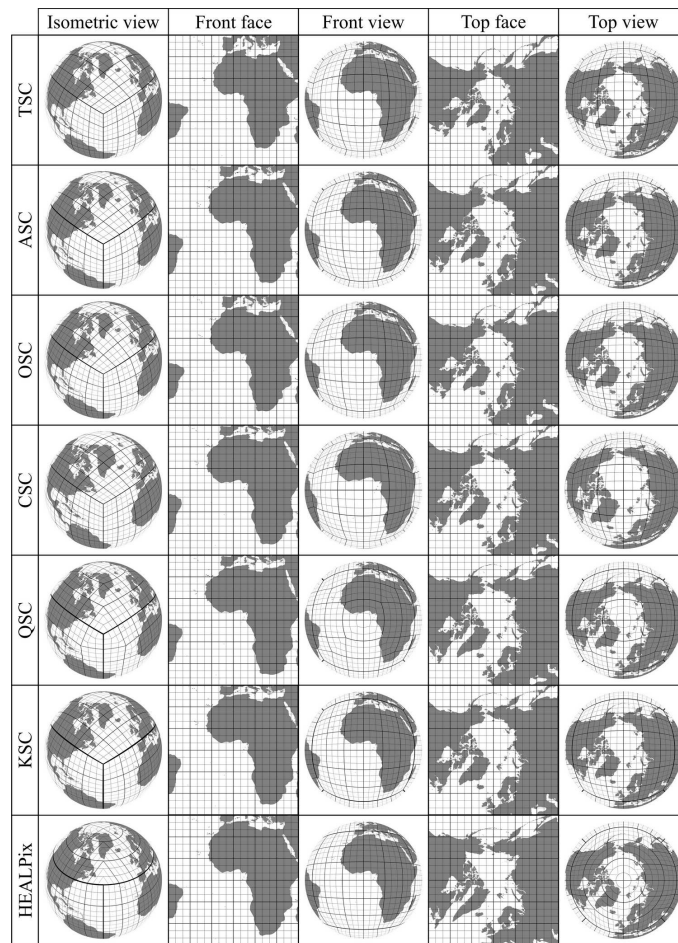


FIG. 6.2: Side-by-side comparison of the SCM inverse transformations effects. The effects are visualized by reprojecting regular grids from the projection planes back to the spherical surface.

REFERENCES

1. M. CALABRETTA AND E. GREISEN, *Representations of celestial coordinates in FITS*, *Astronomy & Astrophysics*, 395 (2002), pp. 1077–1122.
2. M. R. CALABRETTA AND B. F. ROUKEMA, *Mapping on the healpix grid*, *Monthly Notices of the Royal Astronomical Society*, 381 (2007), pp. 865–872.
3. F. CHAN AND E. O'NEILL, *Feasibility study of a quadrilateralized spherical cube earth data base*, Tech. Report EPRF 2-75 (CSC), Environmental Prediction Research Facility, Apr. 1975.
4. A. M. DIMITRIJEVIĆ AND D. D. RANČIĆ, *Ellipsoidal Clipmaps – A Planet-Sized Terrain Rendering Algorithm*, *Computers & Graphics*, (2015), pp. 43–61.
5. GOOGLE, *Google Earth*. <http://www.google.com/earth/index.html>, 2005. Accessed 2014-01-15.
6. K. M. GÓRSKI, E. HIVON, A. J. BANDAY, B. D. WANDELT, F. K. HANSEN, M. REINECKE, AND M. BARTELMANN, *HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere*, *The Astrophysical Journal*, 622 (2005), pp. 759–771.
7. INTEL CORPORATION, *Intel® 64 and ia-32 architectures software developer's manual, vol.1: Basic architecture*. <http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf>, 2014.
8. B. KEMEN AND L. HRABCAK, *Outerra*. <http://www.outerra.com>, 2014.
9. J. KESSENICH, *The OpenGL shading language, language version: 4.50*. <https://www.opengl.org/registry/doc/GLSLangSpec.4.50.pdf>, January 2015.
10. R. KOOIMA, J. LEIGH, A. JOHNSON, D. ROBERTS, M. SUBBARAO, AND T. A. DEFANTI, *Planetary-scale terrain composition*, *IEEE Trans. Visualization and Computer Graphics*, 15 (2009), pp. 719–733.
11. M. LAMBERS AND A. KOLB, *Ellipsoidal cube maps for accurate rendering of planetary-scale terrain data*, in *Proc. Pacific Graphics (Short Papers)*, Sept. 2012, pp. 5–10.
12. R. LERBOUR, *Adaptive streaming and rendering of large terrains*, PhD thesis, Université de Rennes 1, 12 2009.
13. R. LERBOUR, J.-E. MARVIE, AND P. GAUTRON, *Adaptive real-time rendering of planetary terrains*, in *Full Paper Proc. Int. Conf. Computer Graphics, Visualization and Computer Vision (WSCG)*, Feb. 2010, pp. 89–96.
14. MICROSOFT CORPORATION, *Queryperformancecounter function*. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644904\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644904(v=vs.85).aspx), 2014.
15. NASA, *NASA World Wind*. <http://goworldwind.org/>, 2004. Accessed 2014-01-15.
16. NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY (NGA), *Department of Defense (DoD) World Geodetic System (WGS) 1984 – Its Definition and Relationships with Local Geodetic Systems*. http://earth-info.nga.mil/GandG/publications/NGA_STND_0036_1_0_0_WGS84/NGA.STND.0036_1.0.0_WGS84.pdf, 7 2014.
17. P. NOWELL, *Mapping a cube to a sphere*. <http://mathproofs.blogspot.de/2005/07/mapping-cube-to-sphere.html>, July 2005.
18. ———, *Mapping a square to a circle*. <http://mathproofs.blogspot.rs/2005/07/mapping-square-to-circle.html>, July 2005.

19. NVIDIA CORPORATION, *EXT texture filter anisotropic*. https://www.opengl.org/registry/specs/EXT/texture_filter_anisotropic.txt, 11 2014.
20. E. O'NEILL AND R. LAUBSCHER, *Extended studies of a quadrilateralized spherical cube earth data base*, Tech. Report NEPRF 3-76 (CSC), Naval Environmental Prediction Research Facility, May 1976.
21. M. RANČIĆ, R. J. PURSER, AND F. MESINGER, *A global shallow-water model using an expanded spherical cube: nomonic versus conformal coordinates*, Q. J. R. Meteorol. Soc. (1996), pp. 959–982.
22. C. RONCHI, R. IACONO, AND P. PAOLUCCI, *The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry*, Journal of Computational Physics, (1996), pp. 96–114.
23. M. SEGAL AND K. AKELEY, *The OpenGL graphics system: A specification, version 4.5 (core profile)*. <https://www.opengl.org/registry/doc/glspec45.core.pdf>, February 2015.
24. J. SNYDER, *Map projections—a working manual*, vol. 1395 of Professional Paper, US Geological Survey, 1987.
25. STACKOVERFLOW, *Mapping a sphere to a cube*. <http://stackoverflow.com/questions/2656899/mapping-a-sphere-to-a-cube>, April 2010.
26. C. C. TANNER, C. J. MIGDAL, AND M. T. JONES, *The clipmap: A virtual mipmap*, in Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, New York, NY, USA, 1998, ACM, pp. 151–158.
27. T. VINCENTY, *Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations*, Survey Review, 22 (1975), pp. 88–93.
28. L. WILLIAMS, *Pyramidal parametrics*, in Computer Graphics, P. Tanner, ed., vol. 17 of SIGGRAPH '83, ACM, July 1983, pp. 1–11.

Aleksandar M. Dimitrijević
Faculty of Electronic Engineering, University of Niš
Department of Computer Science
Aleksandra Medvedeva 14
18000 Niš, Serbia
aleksandar.dimitrijevic@elfak.ni.ac.rs

Martin Lambers
University of Siegen
Computer Graphics Group
Hoelderlinstrasse 3
57076 Siegen, Deutschland
martin.lambers@uni-siegen.de

Dejan D. Rančić
Faculty of Electronic Engineering, University of Niš
Department of Computer Science

Spherical Cube Map Projections Used In Planet-Sized Terrain Rendering 297

Aleksandra Medvedeva 14
18000 Niš, Serbia
`dejan.rancic@elfak.ni.ac.rs`

Simulation of Time-of-Flight Sensors for Evaluation of Chip Layout Variants

Martin Lambers, Stefan Hoberg, and Andreas Kolb

Abstract—Simulation of time-of-flight (ToF) sensors has mainly been used to evaluate depth data processing algorithms, and existing approaches, therefore, focus on the generation of realistic depth data. Thus, current approaches are of limited usefulness for studying alternatives in sensor chip design, since this application area has different requirements. We propose a new physically based simulation model with a focus on realistic and practical sensor parameterization. The model is suitable for implementation on massively parallel processors such as graphics processing units, to allow fast simulation of many sensor frames across a wide range of parameter sets for meaningful evaluation. We use our implementation to evaluate two alternative approaches in continuous-wave ToF sensor design.

Index Terms—Sensors/sensor phenomena and characterization.

I. INTRODUCTION

CONTINUOUS-WAVE Time-of-Flight (ToF) sensors measure distances based on the time that light travels from an intensity-modulated light source into the scene and back to the sensor. This travel time is derived from the phase shift between measured and reference signal. The phase shift is obtained by electronically correlating both signals in the individual sensor pixels. For this purpose, a typical sensor pixel is of type dual-readout: electrons are gathered during acquisition time in two readout circuits, and an electrical field generated by the reference signal steers electrons into one or the other circuit.

Simulation of ToF sensors is useful for the development and evaluation of ToF imaging and vision algorithms [1]–[3], to produce ground truth and test data. Like the simulation of most imaging sensors, it requires a model of light propagation and illumination, and a model of the individual sensor pixel behaviour [4].

For light propagation and illumination, simulation methods that aim to simulate many sensor frames in a short time typically use rasterization and a local illumination model, i.e. ignoring indirect lighting effects, to leverage the computing power of graphics processing units (GPUs) [5]. Methods that aim to simulate complex illumination effects including multipath effects need to apply global illumination models instead, leading to much higher computational

costs [6], which typically limits these approaches to static scenes and limited parameter variation.

Sensor pixel simulation models can be devised at different abstraction levels. Existing approaches are either based directly on a mathematical formulation of the sensor principle [5], or on a model of the processes that convert photons to voltages [7].

For the purpose of sensor chip layout evaluation, both levels need to be considered. We make the following contributions to the simulation of ToF sensors:

- **Realistic modelling:** We use physical units throughout the simulation, from light source power to sensor pixel readout voltages. This allows to verify simulation results and to interface with existing design and evaluation tools.
- **Sensor parameterization:** We model the individual sensor pixel geometry and layout, in both the dual-readout and single-readout approach, since the placement of components on a sensor pixel is crucial for its performance.
- **Lens parameterization:** We account for vignetting effects by using a thin lens model.
- **Temporal oversampling:** We consider scene motion within the acquisition time and readout time of a single phase image, leading to improved motion artefact simulations.

In this paper, we focus on the evaluation of an alternative sensor chip layout in which each readout circuit contributes to the results of two neighboring sensor pixels (single-readout approach). In effect, this approach increases the amount of optically active areas on the chip while keeping the pixel's fill factor high, and allows to shrink the sensor pixel size [8].

To evaluate the performance of the new single-readout approach in comparison to the dual-readout approach, we simulate both pixel layouts. Since we are interested in practical sensor behaviour, we simulate complete sensors and not just individual pixels, and we consider dynamic scenes to examine effects such as motion artifacts.

Sec. II gives an overview of related work in the field of ToF sensor simulation. Sec. III summarizes the principle of continuous-wave ToF sensors. Sec. IV describes our simulation model. Experimental results and evaluation are presented in Sec. V. Sec. VI concludes the paper.

II. RELATED WORK

Simulating a Time-of-Flight sensor requires the computation of

- the composition of light arriving at a sensor pixel during the acquisition time, and

Manuscript received November 28, 2014; accepted February 21, 2015. Date of publication March 4, 2015; date of current version May 19, 2015. The associate editor coordinating the review of this paper and approving it for publication was Prof. Alexander Fish.

M. Lambers and A. Kolb are with the Computer Graphics Group, University of Siegen, Siegen 57068, Germany (e-mail: martin.lambers@uni-siegen.de; andreas.kolb@uni-siegen.de).

S. Hoberg is with PMD Technologies GmbH, Siegen 57076, Germany (e-mail: s.hoberg@pmdtec.com).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JSEN.2015.2409816

1530-437X © 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

- the sensor pixel response to this incoming light.

Keller and Kolb [5] focus on computing the light propagation in real-time, and use a standard computer graphics lighting model, with a pinhole camera and a point light source at the camera position, and a local illumination model based on Lambertian reflectors. Their sensor model is based directly on the sensor principle: the light that reaches a sensor pixel traveled twice the distance between camera/light source and a surface point. From this information, the phase shift and thus the phase images and the end result are computed as described in Sec. III. Artificial noise is added on top of the simulated data.

Meister et al. [6] focus on realistic light propagation, and therefore go beyond the local illumination computation provided by typical rasterization-based computer graphics approaches. Their global illumination model based on bidirectional path tracing is computationally expensive and thus only suitable for static scenes. However, their technique allows to simulate advanced light propagation, in particular multipath effects. They can either use a basic sensor model similar to the one used by Keller and Kolb, or the physically based model by Schmidt and Jähne.

The sensor simulation approach of Schmidt and Jähne [7] focusses on simulation of the sensor hardware, and thus does not handle light propagation and illumination. It requires precomputed light maps as input. Schmidt and Jähne model optical excitation and target response to simulate the conversion of photons to electrical charges. The use of statistical models allows realistic simulation of various noise effects. Additionally, vendor-specific features such as Suppression of Background Illumination (SBI) are considered.

All of the above approaches focus on simulating the final depth maps acquired from a ToF sensor. Only these depth maps are provided in physical units that are directly comparable to the output of real sensor systems, whereas intermediate results such as phase images and derived values have no physical meaning. Furthermore, parameterization of light source, lens, and sensor layout are limited. In contrast to the global illumination approach by Meister et al., the rasterization-based approach of Keller and Kolb can efficiently handle dynamic scenes. However, motion is only considered between individual phase images; a single simulated phase image is still based on a static snapshot of the scene.

III. CONTINUOUS-WAVE TIME-OF-FLIGHT PRINCIPLE

Time-of-Flight sensors measure distances based on the time t that light travels from the active sensor light source to an object in the scene and back to the sensor. Under the assumption that the light source is a point light source at the sensor position, the light travels the distance d between object and sensor two times: $d = \frac{1}{2} \cdot c \cdot t$, with c being the speed of light.

Continuous-Wave Time-of-Flight sensors emit intensity modulated light in the near infrared range. The pixels on the sensor chip measure the correlation between the reference signal g and the light signal s reflected from the scene.

Following the notation of Kolb et al. [9], we have

$$C(\tau) = s \otimes g = \lim_{T \rightarrow \infty} \int_{-T/2}^{T/2} s(t)g(t + \tau)dt \quad (1)$$



Fig. 1. Four horizontally neighboring sensor pixels in the dual-readout approach with A readout circuit (red), B readout circuit (green), and optically active area (gray).

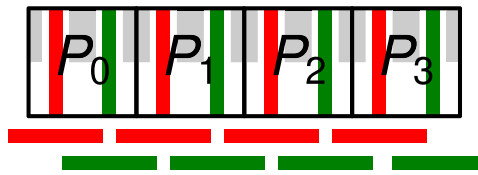


Fig. 2. Four horizontally neighboring sensor pixels in the single-readout approach with A readout circuit (red), B readout circuit (green), and optically active area (gray). The influence areas for each circuit are depicted below the sensor pixels.

Assuming a sinusoidal signal with a modulation frequency f_{mod} , a correlation amplitude a , a correlation bias b , and a distance-dependent phase shift $\phi = 2\pi \cdot 2d \cdot \frac{f_{\text{mod}}}{c}$, the correlation measurement is

$$C(\tau) = \frac{a}{2} \cos(f_{\text{mod}}\tau + \phi) + b \quad (2)$$

The common approach to reconstruct the phase shift ϕ for the distance computation is to use the arctangent on four samples of the correlation function $D_i = C(i \cdot \frac{\tau}{2})$, $i \in \{0, 1, 2, 3\}$. Using the common library function atan2 , we have

$$\phi = \text{atan2}(D_3 - D_1, D_0 - D_2) \quad (3)$$

The correlation function samples D_i are also called phase images. They are obtained by subtracting two signals $N_{A,i}$ and $N_{B,i}$ per pixel: $D_i = N_{A,i} - N_{B,i}$. These signals result from the electrons generated in the optically active area of a sensor pixel, which are directed towards two readout circuits A and B using an electric field that is based on the reference signal g .

In the common dual-readout approach, each sensor pixel has its own A and B readout circuit placed at its left and right border. See Fig. 1. In the single-readout approach, each circuit collects electrons from its two neighboring optically active areas. Sensitive areas that drive electrons to A or B are modulated by the reference signal g . Consequently, the center of generated electrons for A is displaced by half the pixel pitch from the center of generated electrons for B. Thus the difference signal D is also influenced by this shift. See Fig. 2.

The single-readout approach allows a pixel layout with significantly increased fill factor because the number of components for readout circuitry is halved [8]. It is therefore one suitable way to shrink ToF pixels to smaller pixel pitches. In contrast, shrinking the pixel pitch for the common dual-readout approach is not practicable because the resulting fill factor would be very poor. The light-sensitive area of a pixel must have a minimum size to achieve a sufficient signal-to-noise ratio at typical integration times; quantum

efficiency is limited by wavelength and applied sensing material (here silicon). With the single-readout approach, a pixel pitch of $8 \mu\text{m}$ can be achieved, while the dual-readout approach has its limit as $12 \mu\text{m}$. With these pixel pitches, the signal-to-noise ratio of both approaches is comparable, and the total size of the sensor array is kept constant while the single-readout approach provides higher resolution.

On the other hand, the horizontal overlap between pixels in the single-readout approach will likely have a negative influence on the quality of the acquired data: the shifted sensitive areas of the A and B readout circuits can negatively affect the lateral resolution in terms of the modulation transfer function (MTF). The simulator presented in the following is used to evaluate these effects for the single-readout and dual-readout approaches.

IV. SIMULATION

To handle dynamic scenes efficiently, we use the rasterization-based approach of light propagation and illumination, and implement this step and the highly parallelizable problem of simulating many sensor pixels on the GPU.

Like previous approaches, we assume that the modulated light source L and the focus point of the camera C are located at the same position, which without loss of generality is the origin $\vec{0}$. Furthermore, like previous approaches, we assume that surfaces are Lambertian reflectors for the infrared light emitted by L .

In the following, we first describe a model suitable for simulation of the common dual-readout approach. We then solve the problem of simulating the single-readout approach by transforming it to the problem of simulating a special type of dual-readout approach.

A. Light Source

In our model, the light source L is a point light source defined by the following parameters:

- Light power P_L [W].
- Main direction of light propagation \vec{n}_L .
- Radiant intensity $I_L(\theta_L)$, depending on the angle θ_L between the light propagation direction and the main direction \vec{n}_L .

For light sources with homogeneous light propagation over a solid angle ω_L , the light source radiant intensity I_L within ω_L is independent of θ_L :

$$I_L = \frac{P_L}{\omega_L} \quad (4)$$

For an isotropic light source, the solid angle is $\omega_L = 4\pi$. For a light source with aperture angle ϕ_L , the solid angle is given by the area of the spherical cap with height $h = 1 - \cos \frac{\phi_L}{2}$ on the unit sphere: $\omega_L = 2\pi(1 - \cos \frac{\phi_L}{2})$.

If light propagation is not homogeneous, i.e. I_L depends on θ_L , then the light source manufacturer typically provides a table that can be used to look up $I_L(\theta_L)$.

B. Illumination

Consider a point P on a surface that is illuminated by the light source L . See Fig. 3. The surface normal at P is \vec{n}_P . The distance of P to the light origin is $r_P = \|P\|$. The light

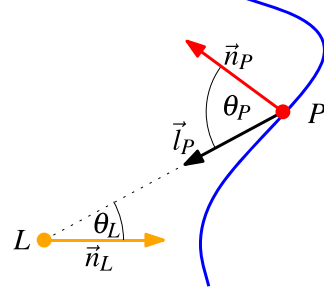


Fig. 3. Surface illumination.

vector (and camera vector) at P is $\vec{l}_P = \frac{-P}{r_P}$. The incident angle is $\theta_P = \arccos(\vec{l}_P \cdot \vec{n}_P)$.

The light propagation angle is $\theta_L = \arccos(-\vec{l}_P \cdot \vec{n}_L)$. With this angle, the light source radiant intensity $I_L(\theta_L)$ is known (see Sec. IV-A).

This radiant intensity can also be written as $I_L(\theta_L) = \frac{dP_L}{d\omega_L}$. Let A denote an infinitesimal area on a light source. Then $d\omega_L = \frac{dA \cos \theta_P}{r_P^2}$, which gives $I_L(\theta_L) = \frac{dP_L}{d\omega_L} = \frac{r_P^2 dP_L}{dA \cos \theta_P}$.

The irradiance E_P [W/m^2] of the surface point P due to the point light source L therefore is:

$$E_P = \frac{dP_L}{dA} = I_L(\theta_L) \frac{\cos \theta_P}{r_P^2} \quad (5)$$

Assuming that the surface is a Lambertian reflector with albedo $\rho \in [0, 1]$, the radiant exitance (or radiosity) B_P [W/m^2] of the surface point P is:

$$B_P = \rho \cdot E_P = \rho \cdot I_L(\theta_L) \frac{\cos \theta_P}{r_P^2} \quad (6)$$

Furthermore, the radiance of a Lambertian reflector is equal in all directions of the hemisphere. This gives us

$$B_P = \int_{2\pi} L \cos \theta d\omega = L \int_{2\pi} \cos \theta d\omega = L\pi \quad (7)$$

Thus, the radiance L_P from P to the sensor is:

$$L_P = \frac{B_P}{\pi} \quad (8)$$

C. Lens

To account for vignetting effects, we use the thin lens model for the sensor camera S , with an f-number N_S (ratio of focal length to aperture diameter). In this model, the irradiance E_S [W/m^2] on the sensor resulting from the radiance L_P is given by the fundamental equation of radiometric image formation [10]:

$$E_S = \left(\frac{\pi}{4} \left(\frac{1}{N_S} \right)^2 \cos^4 \theta_L \right) L_P \quad (9)$$

Further thin lens effects such as aberrations and depth of field are not taken into consideration.

D. Sensor

The sensor consists of an array of $W \times H$ sensor pixels, each with an area A_S . Assuming that one pixel is illuminated only by a small homogeneous area around a surface point P , the optical power P_S [W] irradiated on the pixel is

$$P_S = E_S \cdot A_S \quad (10)$$

P_S is the peak power of the rectangular modulated light signal. It is common that this signal has a duty cycle of 50%. Then the energy accumulated in one pixel over T is

$$W_S = P_S \cdot T \cdot 0.5 \quad (11)$$

This energy is converted into electron-hole pairs in the pixel, depending on the quantum efficiency ν_q that describes how many electrons are generated per incoming photon. The total accumulated charge is mainly depending on the quantum efficiency and the energy (wavelength λ) of photons with is calculated as follows.

$$N_{tot} = \frac{W_S}{\nu_q \cdot \frac{q \cdot \lambda}{h \cdot c}} \quad (12)$$

Here, h is the Planck-constant, c is the speed of light in vacuum, and q is the value of elementary charge.

The total charge N_{tot} is the sum of the electrons accumulated in the two circuits A and B of each pixel: $N_{tot} = N_A + N_B$. The charges N_A and N_B depend on the phase shift and on the four-phase algorithm described in Sec. III. For the four phases $\tau \in 0^\circ, 90^\circ, 180^\circ, 270^\circ$, they are computed as

$$N_A = \frac{N_{tot}}{2} (1 + D \cdot \cos(\tau + \phi)) \quad (13)$$

$$N_B = \frac{N_{tot}}{2} (1 - D \cdot \cos(\tau + \phi)) \quad (14)$$

Here, $D \in [0, 1]$ is the achievable demodulation contrast, and the phase shift ϕ is computed as

$$\phi = 2\pi \cdot 2r_p \cdot \frac{f_{mod}}{c} \quad (15)$$

This model allows to accumulate partial results by summing up charges. This is necessary to allow spatial and temporal oversampling as described in the next sections.

E. Spatial Oversampling

The simulation model described so far considers only a single surface point per sensor pixel, but in reality one sensor pixel covers a larger area. This area may cover inhomogeneous depths, e.g. at object borders. This is the cause of *flying pixels* and related effects.

Like previous approaches [5], we subdivide each sensor pixel into an array of $W_S \times H_S$ subpixels to simulate these effects. The measurements for the full pixel are simply the sum of the measurements of its subpixels.

An effect that has not yet been considered in ToF simulation is that not all areas on a sensor pixel are sensitive to incoming light; some parts are blocked e.g. by control circuits. However, the placement of optically sensitive areas on a sensor pixel affects the measurements.

To simulate this effect, we store an additional mask value $o \in [0, 1]$ per subpixel that represents the optically sensitive

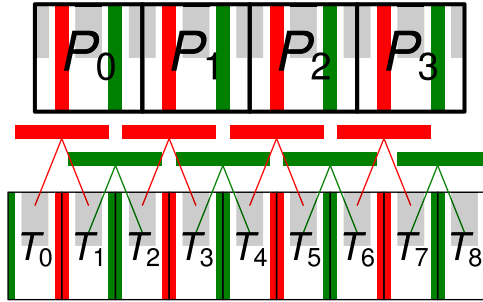


Fig. 4. Reducing the single-readout approach (top row) to the dual-readout approach (bottom row) by introducing virtual dual-readout sensor pixels T_i . The virtual pixels are half as wide as the dual-readout pixels, and are horizontally displaced to them, so that the positions of readout circuits and optically active areas match. The optically active areas of each A readout circuit (red) overlap the optically active areas of its two neighboring B readout circuits (green) and *vice versa*.

portion of the subpixel area. Typically the resulting subpixel mask is identical for all pixels in a sensor.

The simulated results for one subpixel are affected by the value o by taking it into account in the computation of light power in Eq. 10.

F. Temporal Oversampling

The sensor principle assumes that the four phase images D_i refer to the same surface, i.e. the scene is static during the acquisition of the four phase images. In practice, this is not the case. Movements that occur during the acquisition lead to motion artifacts.

Previous simulation approaches [5] accounted for this by simulating the four phase images at different points in time. While the phase image scenes differ in this approach, it is still assumed that no movement occurs during the acquisition of a single phase image.

Since motion artefact compensation algorithms typically work at the level of phase measurements [1], [2], an evaluation of such algorithms using simulated data benefits from accurate motion artifacts even on the level of a single phase image.

Therefore, we subdivide the integration time T into t time steps and simulate separate intermediate phase images according to these shortened integration times as described by Eq. 11. The final phase image is computed as the sum of these intermediate phase images. This assumes that the shortened interval is still significantly longer than $\frac{1}{f_{mod}}$ so that Eq. 1 and Eq. 2 are still valid.

G. Single-Readout Approach

The simulation model described so far applies to the dual-readout approach of continuous-wave ToF sensors.

To simulate the single-readout approach using the same model, we reduce the problem of simulating single-readout results to the problem of simulating dual-readout results by introducing an intermediate simulation step. This intermediate step simulates results for virtual dual-readout sensor pixels T_i that are constructed to match the structure of the single-readout pixels P_i as shown in Fig. 4.

In this construction, the optically active area for each A or B readout circuit in a single-readout sensor pixel now corresponds to the optically active areas of readout circuits in two neighboring dual-readout sensor pixels. The signals $N_A^{P_i}$ and $N_B^{P_i}$ for a single-readout pixel P_i can therefore be computed from the simulated charges for the virtual pixels:

$$N_A^{P_i} = N_A^{T_{2i}} + N_A^{T_{2i+1}} \quad (16)$$

$$N_B^{P_i} = N_B^{T_{2i+1}} + N_B^{T_{2i+2}} \quad (17)$$

Note that the order of A and B circuits in the virtual dual-readout pixels alternates. Therefore, this approach requires that the optically active areas are placed symmetrically between two circuits in horizontal direction.

V. EXPERIMENTAL RESULTS

The simulation model described in Sec. IV was implemented in C++ and OpenGL. All parts of light propagation and illumination and of the sensor simulation are computed on the GPU. Results reported below were acquired on a PC system with an NVIDIA GTX480 graphics card.

A. Evaluation of the Simulation Model

We performed tests to compare measured amplitude data with simulated data.

For this purpose, we set the simulation parameters to match PMD's Camboard Pico which uses Infineons 3D Image Sensor IRS10x0C [11]. In contrast to the original setup, we used an alternative VCSEL light source with a specific intensity profile provided by the manufacturer and integrated it into the simulation model as described in Sec. IV-A.

Fig. 5 shows a simulated amplitude profile for an array of 120×160 sensor pixels. This profile results from the specific intensity profile of the VCSEL and from vignetting effects caused by the lens.

The sensor was mounted on the movable sled of a linear translation stage. A white wall was used as the target scene. The global offset, which describes the averaged phase offset of all pixels, and the Fixed Pattern Phase Noise (FPPN), which is the individual offset of each pixel, were calibrated. Integration time was set to 5 ms. The lambertian albedo parameter of the surface in the simulated scene was tuned to match the real wall measured at a distance of 1 m.

The results shown in Fig. 6 demonstrate a good match between measured and simulated amplitude data for the center pixel. The measured data is affected by the systematic distance error known as wiggling that originates in the triangular correlation function used in the real sensor; the simulation assumes a sinusoidal shaped correlation function as outlined in the sensor principle (see Sec. III), and therefore is not affected by this error. Furthermore, tuning the albedo parameter of the simulated surface at a fixed distance based on measured data affected by wiggling results in simulated amplitude values that are higher than the measured values in the near range ($< 1.5\text{m}$).

Note that our simulation model intentionally does not account for the wiggling error and other errors caused by the electronics used in real sensors. This allows us to compare the effects caused by the sensor principle for the

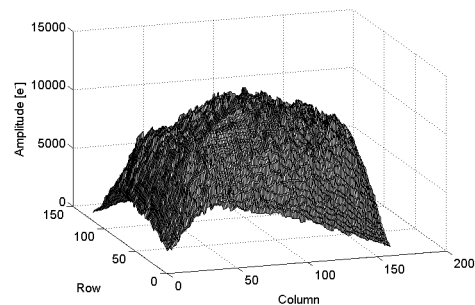


Fig. 5. Simulated amplitude profile.

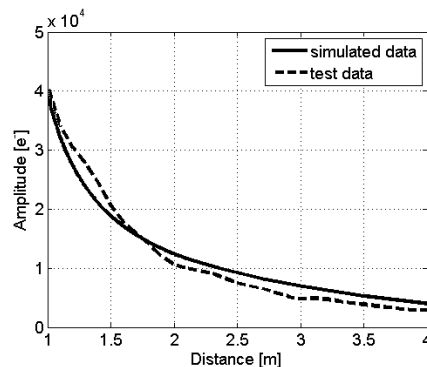


Fig. 6. Comparison of measured and simulated data at the center pixel.

single- and dual-readout approaches without overlay of secondary effects.

B. Evaluation of the Single-Readout Approach

As explained before, the goal of the single-readout approach is to shrink the sensor pixel size. A common pixel pitch scenario is to assume $12 \mu\text{m}$ structure size for the dual-readout approach, and $8 \mu\text{m}$ for the single-readout approach. Pitches in this order of magnitude are described in latest publications related to the dual-readout approach [12]–[14].

For a fair comparison between both approaches, we therefore assume a sensor resolution that is 50% higher in both horizontal and vertical direction for the single-readout approach than for the dual-readout approach.

The signal to noise ratio is comparable since the full well capacity and the sensitive area remain identical, and the pixels typically operate in a regime dominated by shot noise due to ambient light.

1) *Lateral Resolution*: To analyze the lateral resolution precision of both approaches, we use identical simulation parameters except for pixel pitch and number of pixels in horizontal and vertical direction, as explained above. In particular, the opening angle of the simulated sensors are identical.

We then capture a tilted cuboid, with the tilting angle chose to match the slanted edge provided in the ISO 12333

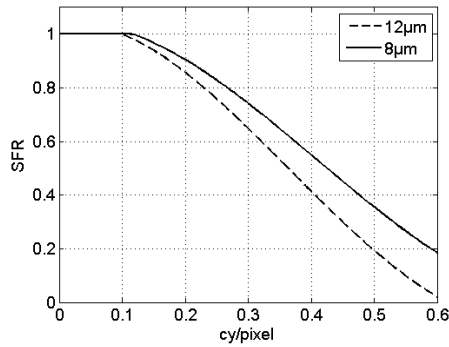


Fig. 7. Spatial Frequency Response (SFR) of $8\ \mu\text{m}$ single-readout pixel and $12\ \mu\text{m}$ dual-readout pixel.

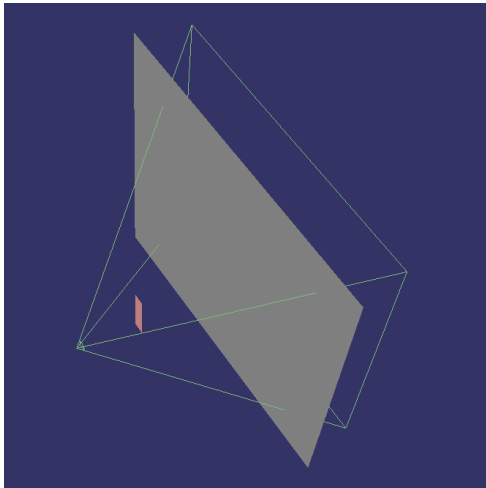


Fig. 8. Test scenario for motion artifacts: static background and moving foreground objects.

chart [15]. A pseudo Modulation Transfer Function (MTF) is then calculated based on the simulated depth maps. The result is presented in Fig. 7. The $8\ \mu\text{m}$ single-readout pixel has a better resolution at Nyquist frequency despite the overlapping sensitive areas described in Sec. IV-G.

2) *Motion Artifacts*: As described in Sec. IV-F, motion artifacts are caused by changes in the scene during acquisition of the four phase images. Since the four samples represented by the phase images do not correspond to a single correlation function as assumed in Eq. 2 and Eq. 3, such scene changes lead to failures in phase shift reconstruction that are hard to predict, and therefore to depth errors of widely varying magnitude. If a sensor pixel is affected by scene motion, then its depth value is useless; there is currently no known way to estimate the error.

For this reason, to measure the susceptibility to motion artifacts of a given sensor, we count the number of pixels affected by motion artifacts for a given dynamic scene, but we disregard the magnitude of the depth errors.

To determine which pixels are affected by motion artifacts, we need to compare the simulated depth value of each pixel with the true depth. However, since the scene is dynamic, there are no true depth values for moving objects: during the acquisition time of one sensor image, the depth observed by a sensor pixel varies.

In order to count pixels affected by motion artifacts, we therefore construct a scene in which only two true depth values can occur: one for the foreground (a moving object) and one for the static background. A simulated depth value for one pixel that is neither the background depth nor the foreground depth must be affected by motion.

An object that exhibits a single depth value would have to be a spherical cap, and movement would have to be restricted to a path with constant distance to the camera. In order to work with planar objects and linear movement paths instead, we compute cartesian coordinates from the simulated radial depth values using the intrinsic camera parameters, and then compare z coordinates instead of depth values.

Based on the requirements listed above, our test scenario is as follows. As static background object, we use a plane with a distance of 1.5 m to the sensor. The plane size is chosen so that it fills the complete view of the sensor. A rectangle of 10 cm with and 50 cm height moving horizontally in a distance of 60 cm from the sensor is our foreground object. See Fig. 8.

If the z component of the cartesian coordinates computed from a simulated depth value differs by more than one millimeter from both the true foreground and true background z coordinate, then the corresponding sensor pixel is considered to be affected by motion artifacts. This threshold works for this simulated scenario because other effects that would decrease precision and increase noise in real sensors are not considered here. However, note that this error measurement also counts flying pixels; a complete separation of these two error categories is not possible in dynamic scenes.

Since the difference between the single-readout and dual-readout approaches only affects pixel rows and not the columns, we simulate horizontal movement. We use constant speeds between 0 and 180 cm per second for each measurement. The simulated time period for each measurement is 0.2 seconds, which corresponds to 26 sensor depth frames in our sensor parameterization.

With an aperture angle of 70° and a horizontal resolution of 256 pixels, a speed of 100 cm per second of an object at 60 cm distance corresponds to ca. 305 horizontally swept sensor pixels per second.

For a fair comparison we assume a higher resolution for the $8\ \mu\text{m}$ single-readout approach than for the $12\ \mu\text{m}$ dual-readout approach. Therefore, we compare the ratios of affected and unaffected pixels.

Fig. 9 (top) shows that this ratio is consistently slightly greater for the single-readout approach than for the dual-readout approach, regardless of object speed. If we assume a structure size of $12\ \mu\text{m}$ and therefore the same resolution for both sensor approaches, this difference is larger

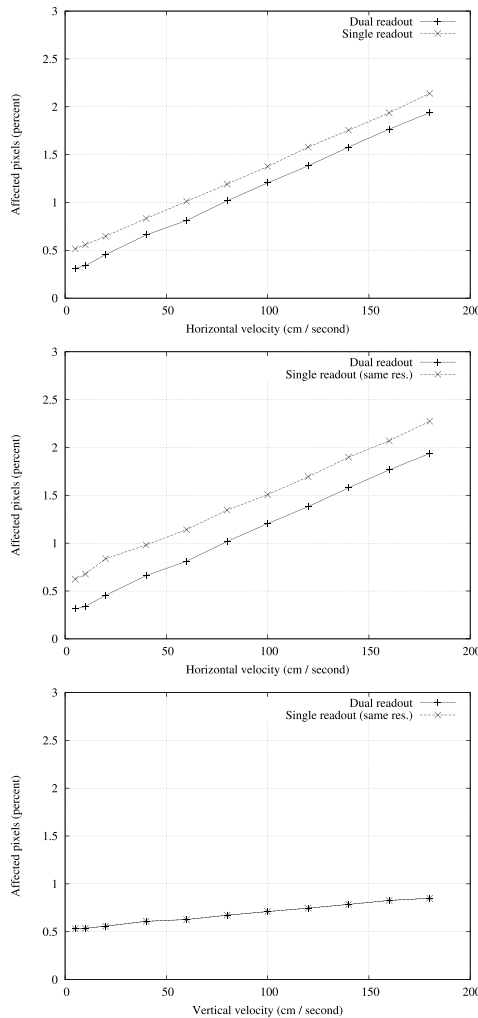


Fig. 9. Ratio of pixels affected by motion artifacts at different object speeds. Top: comparison of the dual-readout and single-readout approaches assuming $8 \mu\text{m}$ structures for the single-readout approach and $12 \mu\text{m}$ structures for the dual-readout approach, resulting in a difference in sensor resolution. Middle: the same comparison assuming $12 \mu\text{m}$ structures and thus identical resolutions for both approaches. Bottom: comparison with identical resolutions using vertical instead of horizontal object movement.

(Fig. 9 middle). When simulating vertical motion and assuming the same resolution for both sensor approaches, the ratios are exactly equal, as expected (Fig. 9 bottom). The nearly constant ratio of flying pixels in the scene can be measured at object speeds near zero.

The absolute differences between true depth and simulated depth values is not meaningful because motion leads to

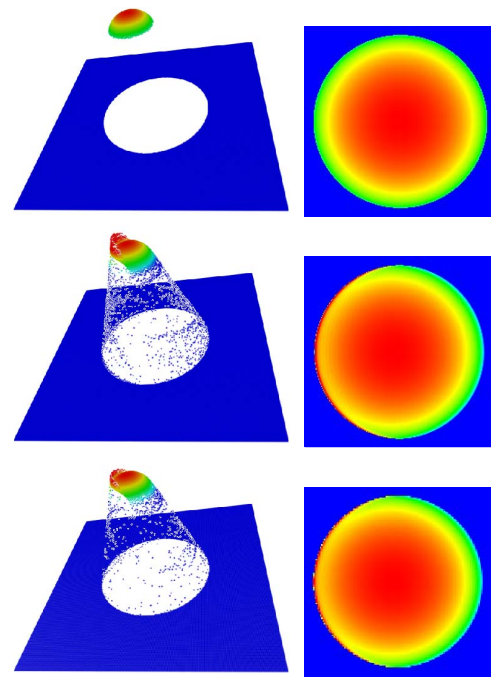


Fig. 10. A simulation result using a sphere as foreground object. Top: a static sphere; no movement occurs. Middle: a sphere that moves from left to right with a velocity of 1 m per second, simulated using the single-readout approach with $8 \mu\text{m}$ structures. Bottom: the same moving sphere simulated using the dual-readout approach with $12 \mu\text{m}$ structures.

unpredictable depth values, as explained above. In this test scenario, absolute differences between 1 mm (the lowest detected error) and 1481 mm occurred.

We also verified that our simulation approach for the single-readout sensor summarized in Eq. 16 and Eq. 17 does not suffer from a directional bias: the results for left-to-right and right-to-left motion of the foreground object show no significant difference, regardless of simulated movement speeds. Furthermore, experiments confirm that vertical movement does not exhibit differences between the two approaches.

In this evaluation with planar objects, pixels affected by motion artifacts can be identified in a simple way. On the other hand, motion artifacts only occur on the left and right borders of the moving foreground object. In contrast, for a non-planar object such as a sphere, each pixel that sees a part of the moving object during the acquisition time will be affected by motion artifacts, since the distance to the observed object surface will change. In this case, true depths do not exist (since depths vary over time), and a true shape does not exist either, so a quantitative analysis is not possible. However, the visual comparison shown in Fig. 10 confirms the results of the previous evaluation: the single-readout approach is stronger affected by motion artifacts than the dual-readout approach.

Furthermore, Fig. 10 shows that the absolute error grows with the variation of the observed depth in both approaches. The observed object shape is not spherical anymore; distortions occur in horizontal direction.

VI. CONCLUSION

Our simulation model improves on existing approaches by using physical units throughout the simulation, by improving parameterization, especially of sensor geometry, and by improved simulation of dynamic scenes through temporal oversampling. We have not presented a noise model for the simulated data, since it was not needed for our evaluation purposes, but per-pixel noise behavior can be integrated into our model in the sensor simulation described in Sec. IV-D. Since we focus on efficient simulation of many frames, to cover a large parameter space and to handle dynamic scenes for motion artifacts, expensive global illumination effects required for the simulation of e.g. multipath effects have not been integrated into our simulation model. Currently our simulation is limited to Lambertian reflectors in the scene. We plan to integrate realistic material models given by measured or modelled Bidirectional Reflectance Distribution Functions (BRDFs) into our model in Eq. 6 – 8. The assumptions that both the light source and the sensor are located in the origin, which is also the center of light rays, has been used before and is considered sufficient for typical sensors intended to record indoor scenes. However, the assumption may not hold for short range applications such as hand gesture recognition. Our sensor model can be enhanced to study the resulting near field effects. In this situation, an area light source can be modelled using multiple point light sources whose contributions are then summed up analogous to the oversampling methods presented in Sec. IV-E and IV-F.

We used a GPU-based implementation of our simulation model to evaluate a new chip layout variant for continuous-wave ToF sensors. The single-readout approach with $8 \mu\text{m}$ structure size provides a better lateral resolution than the common dual-readout approach with $12 \mu\text{m}$ structure size. Even though this new approach is more susceptible to motion artifacts, the improved lateral resolution may be useful for specific application scenarios. Further investigations may find a way to reduce the susceptibility for motion artifacts by reducing the pixel mixing shown in Fig. 2 in a postprocessing step on the sensor.

REFERENCES

- [1] D. Lefloch, T. Hoegg, and A. Kolb, "Real-time motion artifacts compensation of ToF sensors data on GPU," *Proc. SPIE, Three-Dimensional Imag., Vis., Display*, vol. 8738, pp. 87380U-1–87380U-7, May 2013.
- [2] T. Hoegg, D. Lefloch, and A. Kolb, "Real-time motion artifact compensation for PMD-ToF images," in *Time-of-Flight and Depth Imaging* (Lecture Notes in Computer Science), vol. 8200. Berlin, Germany: Springer-Verlag, 2013.
- [3] R. Nair *et al.*, "Ground truth for evaluating time of flight imaging," in *Time-of-Flight and Depth Imaging*. Berlin, Germany: Springer-Verlag, 2013, pp. 52–74.
- [4] EMVA. (2010). *EMVA Standard 1288 Release 3.0*. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.10696>
- [5] M. Keller and A. Kolb, "Real-time simulation of time-of-flight sensors," *Simul. Model. Pract. Theory*, vol. 17, no. 5, pp. 967–978, 2009.
- [6] S. Meister, R. Nair, and D. Kondermann, "Simulation of time-of-flight sensors using global illumination," in *Proc. Vis., Modeling, Vis. (VMV)*, 2013, pp. 33–40.
- [7] M. Schmidt and B. Jähne, "A physical model of time-of-flight 3D imaging systems, including suppression of ambient light," in *Dynamic 3D Imaging* (Lecture Notes in Computer Science), vol. 5742. Berlin, Germany: Springer-Verlag, 2009, pp. 1–15.
- [8] PMDTechnologies GmbH. "Lichtlaufzeitsensor," German Patent DE 10 2013 209 162 A1, May 16, 2013. [Online]. Available: <http://register.dpma.de/DPMAREgister/pat/register?lang=en&AKZ=1020132091621>
- [9] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-flight cameras in computer graphics," *Comput. Graph. Forum*, vol. 29, no. 1, pp. 141–159, 2010.
- [10] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1998.
- [11] Infineon. (2013). *Infineon 3D Image Sensor IRS10x0C*. [Online]. Available: http://www.infineon.com/dgdl/3D+Image+Sensor+IRS10x0C_PB_final_v2.pdf?folderId=db3a3043191a246301192dd3ee2c2ae4&fileId=db3a30433e82b1cf013e847e27e703e9
- [12] A. Payne *et al.*, "7.6 A 512×424 CMOS 3D time-of-flight image sensor with multi-frequency photo-demodulation up to 130 MHz and 2 GS/s ADC," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2014, pp. 134–135.
- [13] S. Kim *et al.*, "Time of flight image sensor with $7 \mu\text{m}$ pixel and 640×480 resolution," in *Proc. Symp. VLSI Technol.*, Jun. 2013, pp. T146–T147.
- [14] L. Pancheri, N. Massari, M. Perenzoni, M. Malfatti, and D. Stoppa, "A QVGA-range image sensor based on buried-channel demodulator pixels in $0.18 \mu\text{m}$ CMOS with extended dynamic range," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2012, pp. 394–396.
- [15] *Photography—Electronic Still Picture Imaging—Resolution and Spatial Frequency Responses*, ISO Standard 12233:2014, 2014.

Martin Lambers received the Diploma degree in mathematics from the University of Münster, Germany, in 2006, and the Dr.Ing. degree in computer science from the University of Siegen, Germany, in 2011.

He is currently a Researcher with the Computer Graphics and Multimedia Systems Group, University of Siegen. His research interests include processing, analysis, and visualization of multimodal sensor data.

Stefan Hoberg received the Diploma degree in electrical engineering from the University of Siegen, Germany, in 2007.

He is currently pursuing the Ph.D. degree at the Research Training Group "Imaging New Modalities". In 2007, he joined PMD Technologies GmbH as an Analog IC Design Engineer. In 2012, he joined the High-Frequency and Quantum Electronics Group at the University of Siegen.

Mr. Hoberg's research interests include modeling of time-of-flight sensor systems, and design of low noise high-sensitive pixel readout circuitry.

Andreas Kolb received the Dr.Ing. degree in computer science from the University of Erlangen, in 1995, under the supervision of Prof. H.-P. Seidel. His academy background is in computer graphics and visualization.

He was the Vice Dean of Research and Junior Scientists with the Faculty of Science and Technology, University of Siegen, from 2011 to 2015. He has been the Head of the Computer Graphics and Multimedia Systems Group, University of Siegen, since 2003. Since 2009, he has been a Spokesman of the Research Training Group 1564 "Imaging New Modalities" funded by the German Research Foundation.

Prof. Kolb has been a member of the Center for Sensor Systems at the University of Siegen since 2005, the ACM/SIGGRAPH since 2002, the Eurographics Association since 2001, and the Gesellschaft für Informatik since 1998.

Ground Truth for Evaluating Time of Flight Imaging

Rahul Nair^{1,2}, Stephan Meister^{1,2}, Martin Lambers³, Michael Balda⁴, Hannes Hoffmann⁴, Andreas Kolb³, Daniel Kondermann^{1,2}, and Bernd Jähne¹

¹ Heidelberg Collaboratory for Image Processing (HCI),
Heidelberg University, Germany
`{firstname.lastname}@iwr.uni-heidelberg.de`
<http://hci.iwr.uni-heidelberg.de/>

² Intel Visual Computing Institute, Saarland University, Germany
<http://www.intel-vci.uni-saarland.de/>

³ Computer Graphics Group, University of Siegen, Germany,
`{firstname.lastname}@uni-siegen.de`
<http://www.cg.informatik.uni-siegen.de/>

⁴ Metrilus GmbH, Erlangen, Germany
`{firstname.lastname}@metrilus.de`
<http://www.metrilus.de/>

Abstract. In this work, we systematically analyze how good ground truth (GT) datasets for evaluating methods based on Time-of-Flight (ToF) imaging data should look like. Starting from a high level characterization of the application domains and requirements they typically have, we characterize how good datasets should look like and discuss how algorithms can be evaluated using them. Furthermore, we discuss the two different ways of obtaining ground truth data: By measurement and by simulation.

1 Introduction

Time-of-Flight imaging is known to suffer from various random and systematic error sources such as multi path, depth wiggling and sensor noise (cf. Chapter 2). Their low resolution additionally restricts their suitability to tasks that do not require a high lateral accuracy. Many methods have been proposed in literature to overcome these problems. Many of them were also published together with ground truth data on which the algorithms were validated. As these works center on the methods themselves, usually less attention is given to the nature of the ground truth (GT) data, with the content chosen to be ‘realistic’ without further specification what ‘realistic’ actually means. Furthermore, the simultaneous optimization of the GT data and the method at hand runs in danger of over fitting the algorithm to the data or vice versa. Therefore, we believe that a more rigorous definition of ground truth for Time-of-Flight imaging is necessary, independent of a specific method at hand or a specific camera manufacturer. The goal of this paper is to better define many of these problems. Starting from

M. Grzegorek et al. (Eds.): Time-of-Flight and Depth Imaging, LNCS 8200, pp. 52–74, 2013.
© Springer-Verlag Berlin Heidelberg 2013

a problem domain analysis we will investigate and discuss requirements for good GT data in Section 2. Next, in Section 3, we will discuss different ways of creating GT data by measuring reference data or by simulation. Then we will explore existing datasets and discuss what characteristics good datasets have in Section 4 before discussing performance measures to evaluate algorithms in Section 5. Finally, we will conclude with a section on best practices and lessons we learned during ground truth acquisition in Section 6.

2 Application Domains and Requirements Engineering

It is unlikely that we can find one generic algorithm which optimally works under all circumstances: This is called the generalization-specialization-dilemma. It states that given an application, our algorithm might either be so specific that it is not able to deal with previously unobserved data (overfitting). On the other hand the algorithm might generalize well over many scenarios but yield mediocre results in each of them. Thus, in order to analyze the appropriateness of an ToF algorithm for a given application, we need to know the application.

On the other hand, there might be an infinite number of yet unknown applications for ToF algorithms. It seems unlikely that we can first enumerate all applications and then analyze the performance of each and every algorithm for each and every application. System engineers found a way around this problem by identifying a number of meaningful and intuitive properties for each system component (c.f. Table 1). These are measured and then listed in a specification sheet. These properties are selected by finding those which are, ideally, important for as many relevant applications as possible. In order to select the most indicative properties, all currently available applications are considered. Then, by experimentation, system properties are selected and tested for their usefulness.

For four example applications we have identified a set of requirements, and analyzed which ones are relevant for each application. Table 1 enumerates the importance of several requirements for the example applications. Based on these findings, ground truth data and appropriate test scenarios can be acquired / generated to evaluate the performance and suitability of depth imaging devices and algorithms with respect to the application requirements. In the following we will discuss the requirements of various application fields, except for multimedia, which is discussed in Chapter 6. As low-level pre-processing algorithms are used in all applications below, we will first summarize their requirements separately.

Low-Level Pre-processing Algorithms

Well-known issues (cf. Chapter 1) such as noise (cf. Chapter 2), multi-path or motion-induced artifacts often need to be reduced before high-level algorithms can try to understand the scene. Superresolution (SR) is regularly used to scale up depth images and thereby increase the detail of depth edges. Usually, this approach is coupled with denoising and sometimes with sensor-fusion (cf. Chapter 6) where other cameras are used to obtain hints on how to increase the detail.

Table 1. Requirements and their importance for selected example applications. A + indicates an important requirement, 0 is less important and - unimportant.

Requirements	Gesture Control	Room Supervision	Driver Assistance/ Robotics	Multimedia
Low Latency	+	-	+	+
Low Noise / High Precision	0	0	+	+
High Accuracy	-	-	+	-
High Frame Rate	+	0	+	+
Motion Robustness	+	0	+	+
Robustness Against Environmental Influences	0	0	+	+
Interference Robustness	-	+	0	0
Low Hardware Requirements	0	0	+	-
Graceful Degradation Self-Inspection	0	+	+	0
Depth Range	-	0	+	-
Lateral Resolution	+	0	-	+

Multi-path is a largely unsolved problem, whereas motion artifacts can already be handled to some degree. In terms of requirements, these algorithms have in common, that they should ideally be able to annotate their outcome with confidences so that higher-level methods are able to judge whether they want to use the data at all.

All of these algorithms address information-theoretic problems: given a subset of the information of the scene, how can we add believable detail from other sources? Information is added by prior knowledge as simple as interpolation kernels, regularization techniques or more complex cues such as e.g. temporal coherence or different modalities (such as RGB color).

Other algorithms on intermediate and higher levels of semantic understanding could be 3D reconstruction and camera/object tracking as well as object detection and scene understanding. All of the discussed requirements can play a more or less important role, such as depth accuracy is important for 3D reconstruction but not necessarily for scene understanding whereas speed and beauty can play a role depending on the application domain.

Gesture Control

Gesture Control is one of the major and most mature depth imaging applications. Most gesture interaction systems do not need an overly high depth accuracy and precision, as they use distinct gestures to trigger actions and the accuracy of interactions is primarily limited by ergonomic considerations. However, low latency and motion robustness are mandatory for a good and pleasant gesture control system.

Room Supervision

With the room supervision application in mind, we are interested in the number of people in a room. Therefore, we want to use multiple cameras that observe the room from different angles to ensure that no person is occluded. The use of multiple cameras requires that they are able to observe the same scene without interfering with each other. The processing power required for image processing is increased with each added camera.

Reliability is very important in this application, so the system should be able to detect when it is in an undefined state. Since exact localization of the persons is not required, other requirements can be relaxed. Latency, motion robustness, accuracy and noise are not a big issue.

Driver Assistance/Mobile Robotics

In driver assistance and autonomous mobile robotics, i.e. realtime systems with limited computational resources and possible public safety issues, speed is not the only important criterion: Energy, memory and bandwidth consumption have to be taken into consideration, as many subsystems are competing for limited system resources. Accuracy is also relevant, as the desired behavior depends on the distance to a detected obstacle - e.g. start braking or initiate an evasive maneuver. Finally, the algorithm should not only have self-inspection capabilities, but also degrade gracefully, as small irritations such as specular reflections on other cars or interference with other ToF-equipped vehicles can occur frequently.

In the following chapters, we will discuss different techniques to create datasets for benchmarking such applications and requirements.

3 Ground Truth Generation

There are two ways of generating ground truth data. By measurement or by simulation. Both these approaches assume that the reference data is dense and has an error which is an order of magnitude smaller than the expected ToF error¹. In all cases, two datasets are created simultaneously: A scene captured using ToF along with reference data for comparison. If reference data is measured, the accuracy of the reference modality and alignment issues need to be considered (cf. Chapter 3.1). GT can also be obtained by simulation. Here, a scene with known geometry is used as a starting point and the ToF data simulated using various ToF models (cf. Chapter 3.2). Even though such models usually need to make simplifications to remain tractable, they offer the opportunity of white box testing. Hence, this shifts the problem to the question what a good simulation is. The reference data may be exact but the derived data can show subtle differences compared to a real sequence.

¹ In Section 5.2 on weak and sparse ground truth we will discuss what can be done if this requirement does not hold anymore.

3.1 By Measurement

We will introduce various methods which can be used for measuring ground truth before discussing alignment issues that are relevant to all these methods. These datasets can be used for general black-box testing. In the last part, we will also discuss various strategies for isolating specific aspects of Time-of-Flight imagers for white-box testing.

3.1.1 Methods

A. High Precision Scanning

High precision scanning techniques include Time-of-Flight and triangulation based laser scanners as well as structured light scanners. **Structured light scanners** and **triangulation based laser scanners** both infer depth by triangulating the position of some active illumination pattern. Figure 1 depicts two example scans using these modalities. These scanners typically have an accuracy of 1 - 100 microns and can safely be considered to be an order of magnitude more accurate than ToF imagers [1]. They usually have a limited working volume of a few liters and have a working range of up to 2 meters[. As all optical measurement techniques they succumb to objects with specular surfaces. Therefore, for best results, often the object to be scanned has to be coated with a diffuse paint.

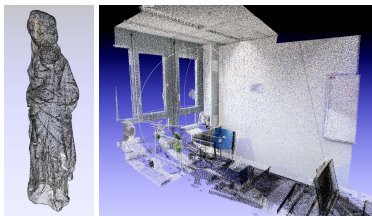


Fig. 1. Left: high density point cloud of statue scanned using structured light. Right: Point cloud of office space acquired using terrestrial laser scanning.

In recent times ToF based **terrestrial laser scanners (TLS)** commonly used for large scale terrain and building scanning have received more attention for creating ground truth datasets for applications such as stereo matching or optical flow [2] and most recently also for Time-of-Flight imaging [3]. Most ToF laser scanners claim to have an accuracy of a few millimeters over a wide range of distances (2 - 100 m), making them an ideal modality to create ground truth for static scenes. As with ToF imaging the accuracy can deteriorate depending on the actual sensor-scene setup[1], though the effects

of such errors are certainly smaller than in ToF cameras. Still, a few aspects should be considered while creating TLS datasets. Depending on model and position of the scanner as well as the skill of the operator the error can reach multiple centimeters.

Mixed Pixels. As the pixels have a certain size, the laser beam also has a certain width and is subject to beam divergence which causes mixed pixel effects at object boundaries. Since the reconstruction formulas used in most ToF scanners are not as highly nonlinear as in the ToF imager 1, these points lie between the foreground and background depth. Depending on the distance of the near object this can still lead to a substantial broadening of the object. Current TLS systems employ lasers with starting beam diameters of 6mm and beam divergences in the range of ≈ 0.1 mrad. Although the most often used gaussian beam profile allows for a more exact localization of the beam center in orthogonal direction, mixed pixel effects will be observed in a region of the size of the same magnitude as the beam diameter. A rough calculation with a 40 degrees FOV and 200 pixels sensor resolution for the ToF imager, yields that a pixel accounts for around 6 mm at 2 m distance (Effects dependent on the point spread function not taken into consideration). That means that especially super resolution, flying pixel compensation and denoising algorithms should evaluate whether mixed pixel effects actually affect their evaluation at depth boundaries. An example scan with mixed pixels can be seen in Figure 2.



Fig. 2. LiDAR based point cloud with high amount of flying pixels

Material Based Offset. Similarly Clark et al[4] and Boehler et al[1] have reported a material/albedo (amplitude) based offset in the order of magnitude of a centimeter. Though they cite different intensity based offsets, this can be due to calibration/environmental effects or equipment degradation (see last point). Again a visual inspection of the laser scan data is

advisable to ensure that such effects are not present or are compensated for [4].

Resolution/Sparsity. The resolution or point density of various scanners can often be adjusted in a certain range. This can alleviate problems caused by depth discontinuities making sure that the scene is sampled denser than by the ToF camera to evaluate. Keep in mind that this will not automatically compensate for mixed pixel effects. Scan density can also be reduced in favor of speed. Velodyne scanners for example have a very small vertical resolution of only 64 scanlines. The advantage is that scan times are significantly lower, reaching even the typical frame times of ToF cameras. This provides for some interesting evaluation scenarios as reference data can be both accurate and dynamic but is also sparse in at least one spatial direction.

Angle of Inclination. A small effect of surface inclination towards the laser beam was observed by [5]. This effect was accounted to be up to 2 mm such that it alone most probably will not affect scan quality.

Scanning Volume/Shadowing. TLSs are made for large scale scanning. Hence, using them in closed cluttered scenes will lead to a lot of shadowing which requires tedious additional scans (especially as these scanners are usually quite heavy).

Calibration/Environmental Effects. Boehler et al[1] reported that scan performance can deteriorate depending on handling and age of the scan equipment. Also as high precision mirrors are used, a drift in depth can be caused due to temperature variations. Many of these effects can be compensated for by appropriate calibration and we refer to the methods proposed in [6].

Scan Time. Scan Time for these devices is typically at least a few seconds, although real-time scanners with severely reduced lateral resolution exist. This limits their use to static ground truth scenes.

B. Kinect Fusion

Although depth cameras using different modalities may only offer comparable but not superior accuracy, their output can still be useful for evaluations. Examples would be the use of intelligent data fusion or temporal integration approaches. We will demonstrate this on the example of the KinectFusion pipeline [7,8].

The Kinect camera itself does not have a better lateral resolution than typical Time-of-Flight cameras and its depth accuracy is also only in the centimeter range [9]. Depth images acquired from the camera are therefore not directly suited for evaluation.

Nonetheless, an interesting approach to ground truth generation using the Kinect was presented by Meister et. al [10]. The Kinect Fusion algorithm presented by Newcombe, Izadi et al. [7,8] uses the input of a Kinect camera to recreate a 3D representation of a scene. This is done by converting the

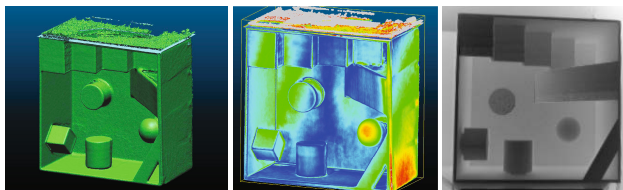


Fig. 3. Left: KinectFusion generated mesh of evaluation target. Middle: Euclidean error of mesh (red:15mm). Right: ToF depth image of the test target.

cameras depth data into a voxel based implicit surface representation [11]. Each new view and camera position is matched to the previous ones using a special variant of an iterative closest point algorithm and the voxel volume is then updated with new surface data.

This approach could be described as volumetric superresolution since the resulting 3D representation is more accurate and shows less noise than the individual depth frames. The polygon meshes created by applying marching cubes to the zero-level set of the surface are known to have a geometrical precision of 10 to 80 mm, depending on scene size. This does at least fulfill the requisites for weak ground truth (c.f Section 5.2). Advantageous is that this method works without complicated setup procedures and does not require expensive equipment. It should be noted though, that as Kinect Fusion relies on depth map registration, the quality will also depend on the amount of “clutter” in the scene.

C. Self Made Targets

Up until now arbitrary scenes were measured with ToF and a reference modality for later comparison.

A different approach to creating real world test objects is to start off with a computer model and produce them using various manufacturing processes. This allows to create targets with specific characteristics such as known curvature or controllable reflection properties using different materials.

Hand Measurement/Construction. Though probably considered somewhat archaic, for simple geometries it is possible to create the test targets manually. Objects with accuracy of a few millimeters or lower can be created using standard materials like wood or metal. The test object in Figure 5 was created from fiberboard with an accuracy of $\approx 1\text{mm}$.

Milling. On the other end of the accuracy spectrum, processes such as CNC milling have a precision of a couple of microns. The size of objects created this way is typically limited. An example object can be seen in Figure 4.

3D Printing. Automated 3D manufacturing even for private home use is becoming increasingly popular in the recent years. So called 3D printers which can produce arbitrary objects by depositing thermoplastics are available for a few hundred Euros. Although typically limited to objects as few dozen centimeters wide they also reach precisions of tenth of millimeters.

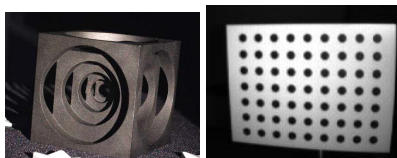


Fig. 4. Left: CNC milled object, Right: Photogrammetric Target

3.1.2 Alignment

Once the same Dataset has been acquired using different modalities, the ToF data needs to be aligned to the 3D data by estimating the relative rotation and translation between the camera and the reference coordinate system. The first decision that has to be made is whether this alignment is done using the point cloud data directly or by first estimating the ToF camera pose in the reference dataset and then back projecting the ToF points into space.

Manual Alignment. As a baseline it is possible to manually align the ToF with reference data. This can be a very tedious job as 6 DOF have to be optimized by hand. We suggest that instead users choose a set of correspondences between the ToF data and the Reference data, and then align them by means of standard pose estimation techniques [12].

Point based matching. This will work if the scene is cluttered enough, otherwise it may result in drifts. Also it should be noted that systematic errors in the scene will lead to a bias in matching. This can be circumvented to a certain extent by applying sparse point matching or robust versions of ICP [13] that also account for the anisotropic noise present in the ToF point clouds.

Using Targets. Photogrammetric targets can be used to semi-automatically align ToF with reference data. In case of RGBD reference data this amounts to finding and matching feature points of known properties. Typical examples are circles or checkerboard corners as they are easy and exact to locate (c.f. Figure 4).

Partial Alignment. In certain situations the geometric properties of specific areas of the image are known (e.g. a planar wall or table). It is therefore also possible to just fit the ToF pixels to these simple models.

3.1.3 Isolation of Effects

As many algorithms claim to target different specific aspects of ToF imaging, datasets should help assess how well they perform. In the following we propose various techniques to separate those specific aspects.

Temporal averaging. If the goal of the algorithm is to remove the statistical fluctuation of the depth map, a GT dataset can be simply created by temporal averaging of the input intensity images. The obtained dataset will still contain all systematic errors, but can still be used to assess the power of different regularizers (See Section 2).

Controlled Movements. Controlled Movements are interesting for algorithms that try to compensate for motion artifacts (Section 1). Basically two parameters can be optimized: *a*) How accurately the movement can be controlled and *b*) the accuracy with which the movement can be tracked. An example for *a*) was given by Schmidt et al. [14] who used a rotating target of known geometry with constant angular velocity to create ground truth for their experiments. Another possibility is to use a rail system to constraint the movement in one dimension. The movement can then either be estimated by tracking a target or by evaluating the optical flow.

Lighting. While the light is usually fixed to the camera, Schmidt et. al [14] separated the light from the sensor. This was done in order to decouple saturation effects from the wiggling error.

Materials. As different materials affect the ToF measurement, especially multi-path, the usage of specific materials can help making measurements with reduced multi-path effect.

- As specular surfaces tend to show more multi-path interference, the usage of highly lambertian surfaces such as spectralon could be used to isolate interreflection effects.
- Often white walls in rooms increase global multi-path effects. Hence, dark absorbing materials such as fleece should be used to reduce these effects.
- Another interesting idea is to use infrared retroreflecting spots or sprays while reducing the integration time such that the direct reflection outweighs the multi-path illumination. Combining this with dark materials in a scene can possibly be utilized to estimate intra-lens reflections.

3.2 By Simulation

Computer generated data is in many cases a suitable alternative to measured ground truth data for algorithm validation [15]. In the context of Time-of-Flight imaging, a sensor simulator can provide such data.

To be useful for Time-of-Flight imaging method evaluation, a sensor simulator must provide two kinds of data: ideal depth data without any noise or artefacts, and realistic sensor data with typical noise characteristics and relevant systematic errors. The latter may include intermediate results that a real sensor may provide to the user, such as phase images (c.f Chapter 1).

The problem of simulating Time-of-Flight sensor data encompasses two areas:

1. Modeling of scene geometry and materials, light source, and light propagation, and
2. Modeling of the sensor hardware and operating principle.

Ground truth data generation for imaging method evaluation requires simulation of sensor data that exhibits a clearly defined and identifiable challenge. Depending on this problem domain, a simulator can be based on models on different abstraction levels in both areas. Typically, some aspects of the complete system have to be simplified in order to maintain tractability. For example, to produce ground truth data for the evaluation of methods to reduce motion artefacts, a simulator might choose relatively straightforward optics and light propagation models in order to allow fast computation of multiple sensor data frames for dynamic data.

3.2.1 Light Propagation

The light propagation model determines the composition of light that reaches each sensor pixel in the sensor hardware model.

For that purpose, the light propagation model must encompass a modulated light source, a scene description consisting of geometry and material properties, a reflection model for each material, and a camera optics model.

Keller and Kolb [16] use a model based on the traditional computer graphics pipeline. The camera is a pinhole camera and the light source is a point light source located at the camera pinhole. All object surfaces in the scene are assumed to be Lambertian reflectors at the wave length of the sensor light source.

In this model, the light that reaches a sensor pixel traveled twice the distance d between camera / light source and a surface point. Thus, the phase shift of this incoming light relative to the light source is known. Furthermore, its amplitude can be computed from d , the direction from the light source to the surface point, and the surface normal.

Knowing phase shift and amplitude allows to compute the four phase images typically generated by Time-of-Flight sensors, and from these phase images the final depth map can be computed. See Sec. 3.2.2.

The advantage of this model is that a simulator can leverage the processing power of Graphics Processing Units (GPUs) to compute light propagation information for many sensor data frames, even for complex and dynamic scenes. Furthermore, existing modeling tools from the computer graphics domain can be used to create and animate test scenes.

Keller and Kolb use a spatial oversampling to simulate typical effects such as flying pixels. For each sensor pixel, the rasterization produces a block of subpixels with light source information for the cone covered by that sensor pixel. This allows to compute the incident light properties as a mixture of the responses of different surface points, which is important e.g. at object boundaries. Additionally, computing the four phase images at distinct points in time allows to simulate motion artefacts in dynamic scenes, with the limitation that the scene is still assumed to be static during the exposure time of each phase image.

The rasterization approach can be extended to support the simulation of more effects by using methods known from the Computer Graphics domain. This includes the approximation of area light sources with multiple point light sources, improved modeling of material and reflection properties, transparent materials, depth of field, and certain types of distortions caused by camera optics.

However, as known from Computer Graphics, rasterization has certain limitations that prohibits its use for more complex light propagation effects. In the context of Time-of-Flight imaging, multi-reflection (or multipath) artefacts are a particularly interesting example. To simulate such effects, global illumination methods such as Photon Mapping or Path Tracing have to be employed. These methods typically have significantly higher computational costs, to the point that they are impracticable for complex dynamic scenes. However, they can generate much more realistic light propagation information.

The main challenge in applying global illumination methods to Time-of-Flight sensor data simulation is that the composition of modulated light reaching a sensor pixel must be known, including phase shifts.

Calculating all possible or even only all physically relevant paths between (multiple) light sources and the camera is practically impossible. So most GI algorithms try to infer the light distribution in a scene using intelligent sampling schemes and by making certain assumptions about the light contributions. Two of these assumptions namely that light propagation is instantaneous and that the lighting situation is in a steady state for a single frame do not hold for ToF imaging.

A possible approach to this problem is the simulation of individual phase raw-frames similar to the mentioned scanline renderers. The path length (and therefore the phase) of light which was reflected multiple times inside a scene can be tracked and then be used to modulate the individual light contributions when they hit a sensor pixel. These modifications could easily be added to most existing global illumination algorithms.

In most cases this image synthesis is a stochastic process and noise in rendered images is of a different nature than the sensor and photon noise present in real cameras. A physically correct simulation would therefore need to be sustained until the render noise has no more significant influence on the generated depth maps. Then correct sensor noise as shown in Section 3.2.2 could then be applied to the raw data. In practice, the minor differences in the noise characteristics of the various simulations seldom justify the massive increase in computation time this approach would necessitate.

A problem arises from the typically high-order parameter space of various global illumination methods. Apart from the possible settings of the render engine such as light sample sizes, recursion depth or sampling parameters, material parameters such as reflectivity, surface roughness or texture can be changed individually. The Time-of-Flight simulation may depend on any of these parameters and finding the correct ones can be a challenging task in itself. Experiences from computer graphics or image synthesis can only be applied partially as they may

be focused on subjective expectations (Does it look good vs. does it look real) or e.g. be limited to the visual spectrum.

Light propagation effects that have not yet been addressed by simulators based on either rasterization or global illumination include advanced optics effects such as lens flare and scattering inside the camera casing. The properties for lenses used in Time-of-Flight imaging are generally quite different from those of regular lenses. For example, they need to be transparent for infrared wavelengths and suitable for intensities with high dynamic range. If a description of the lens is available, the lens effects can be computed using optical engineering software such as OSLO [17] or Zemax [18].

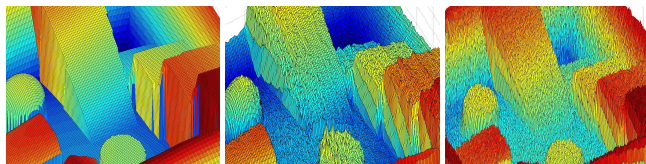


Fig. 5. Left: Ground truth geometry of test object. Middle: Geometry as seen by real ToF camera. Right: Simulation with method by Keller and Kolb.

3.2.2 Sensor Hardware

Given the composition of incoming light as computed by the light propagation model, a sensor hardware model can simulate the sensor pixel response. Again, such hardware models can have different abstraction models, depending on the problem domain.

Keller and Kolb [16] employ a phenomenological sensor model. Based on the light input for each of the four phase images of typical PMD sensors, their simulator computes the ideal theoretical phase image sensor response, as described in Chapter 1. To achieve a realistic simulation result (as opposed to a perfect result), Keller and Kolb apply additive and multiplicative Gaussian noise to these ideal phase images before computing the final depth map sensor response.

This phenomenological model focuses on computational efficiency for the handling of dynamic scenes: similar to their light propagation model, Keller and Kolb leverage the processing power of Graphics Processing Units (GPUs) to compute the sensor response for each pixel in parallel.

Schmidt and Jähne [19,20] employ a physical model of individual sensor pixel components, with the goal of simulating the cause of sensor data imperfections instead of applying noise effects after computation of the sensor response. This leads to better understanding of sensor data, more realistic simulation results, and more fine-grained control over simulator behavior.

In particular, their physical model accounts for real-world effects like non-sinusoidal light modulation, non-rectangular switching functions, non-linear photo response, and the influence of sensor-specific techniques such as the

Suppression of Background Illumination (SBI) method commonly found in PMD sensors. Consequently, their system can simulate various types of sensor noise realistically.

Both the phenomenological approach by Keller and Kolb and the physical modeling approach by Schmidt and Jähne originally used light propagation information generated by basic rasterization methods, but could also be combined with more sophisticated light propagation models.

In addition to these approaches on relatively high abstraction levels, sensor manufacturers can employ chip design evaluators that simulate chip behavior on a transistor level, based on chip descriptions in VLSI. Such simulators naturally require immense computational power even for small sensor resolutions, and are therefore typically impracticable for producing ground truth data targeted at the evaluation of imaging methods.

4 Content Selection and Available Datasets

4.1 Content Selection

An equally important question is which targets or scenes should be used for the ground truth sequences. Three important aspects of ground truth datasets are **interpretability**, **progression** and **realism**. Interpretability refers to the aspect that an algorithm failing or working on a certain dataset should not only tell us that it failed or worked but also give some insight that the algorithm fails due to certain conditions. As a baseline, isolated effects on simple planar geometries can be analyzed. In Section 5.2.1, simple geometries and measurement methods to obtain certain effects are discussed. In more complex scenes masks can be supplied that highlight only certain effects in the scene such as specular reflections, multi-path or transparency. The second aspect - progression - is concerned with avoiding a problem heavily used datasets have. As the optimization criterion is to minimize a certain error measure on this particular dataset researchers tend to overfit their algorithms to this specific dataset[21]. On the other hand, if the dataset is initially too challenging, it may not receive the needed traction in the research community.

Therefore we believe that a large database should include a progression of difficulty to accommodate for this. Progression can be obtained by combining different effects but also by more complex geometries that make simple regularizers etc. fail.

Finally, realism refers to the GT data being relevant to actual use cases. Obviously a dataset can not be exhaustive regarding all possible applications. But once an application domain is identified the goal should be to create ground truth data as close to the actual working conditions as possible.

4.2 Existing Datasets

So far the number of available datasets and scene compositions is limited. Here we present a short but representative selection of the available sets along a short

description of the included data. Either the dataset name or the title of the corresponding publication is given.

Capturing Time-of-Flight Data with Confidence. To calculate confidence values for ToF imaging Reynolds et al. [3] provide a ground truth dataset based on laser scanner data. The set consists of four scenes of which two are augmented with ground truth data. ToF data includes depth maps as well as intensity and amplitude images and intrinsic calibration data. Alignment between the two modalities is enabled by the usage of reflecting calibration markers. The dataset is available at <http://visual.cs.ucl.ac.uk/pubs/tofconfidence/>.

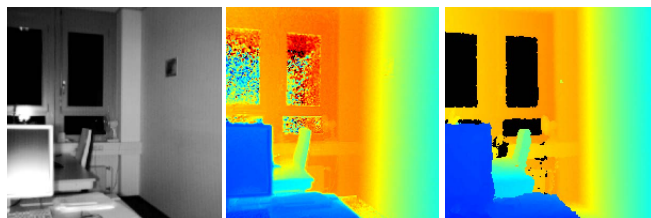


Fig. 6. Left: ToF Intensity Image, Middle: ToF Z-Depth. Observe the 'halo' around object corners. Right: Ground Truth Z-Depth.

HCI LiDAR Dataset. A ToF dataset referenced with terrestrial laser scanner data is available at <http://hci.iwr.uni-heidelberg.de/Benchmarks/>.

It consists of images of an office scene taken with a PMD CamCube 3 ToF camera. Included are intensity images, depth images, sensor raw data as well as camera calibration data. The displayed room was also scanned using a RIEGL VZ-400 terrestrial laser scanner² with an stated accuracy of 5 mm and 3 mm precision. The scan was performed from 6 individual positions with approximately 5 Million points per scan.

Ground truth depth maps were created by manually selecting 2D-to-3D correspondences between the ToF intensity images and a delaunay triangulation of the scanned point clouds. Using these correspondences camera pose estimation was performed and the depth of the triangulated mesh was rendered.

It should be noted that the lidar based polygon mesh contains multiple errors or holes due to occlusions and flying pixels in the point cloud. See Section 3.1 for typical problems regarding laser scanner data. Figure 6 shows a ToF intensity image of the dataset as well as a depth map and corresponding

² <http://www.riegl.com>

ground truth map. The dataset does not adhere to all points mentioned in Section 6 but is still useful for many evaluation tasks.

HCIBOX Depth Evaluation Dataset. An additional dataset is available under the same url (<http://hci.iwr.uni-heidelberg.de/Benchmarks/>). The test object is a wooden box containing several simple geometric objects such as cylinders or spheres, see Figure 5 for an image. This set consists of ToF images of the same type as well as fully calibrated megapixel stereo images. Ground truth depth maps were again created using standard pose estimation techniques. For this set the 3D model was created by measuring the depicted test object by hand, reaching a general accuracy of ≈ 1 mm.

Locally Consistent ToF and Stereo Data Fusion. This dataset created by Mutto et al. [22] contains both rectified stereo camera data as well as data from a MESA SR4000 camera and is intended for use in sensor data fusion approaches (<http://ltm.dei.unipd.it/downloads/tofstereo>). The ToF data does include amplitude, depth and confidence images and is of particular interest as ground truth data for this camera model is rare. Three scenes containing household objects are contained in the set. Ground truth for was created using the Spacetime stereo method by Zhang et al. [23], relying on the stereo systems images itself to create the reference data. Although there are no indications about the accuracy of the used ground truth. Additionally some rendered stereo images and ToF depth maps are also available from the same location. These depth maps are purely synthetic and lack typical ToF artifacts and errors.

5 Application of Ground Truth

5.1 Algorithm Performance Metrics

In this section we will give a short overview over the most often used error metrics and give advise on how or when they should be used.

The performance metric most commonly used for GT evaluation is the mean endpoint error/squared error. It is defined as the mean of the absolute distances between the ground truth and the measured depth. Though useful in many cases, the reduction of performance to a single scalar value may not be too meaningful. To name one example: Often a visually pleasing result that contains a bias is more preferable to a bias free solution with a high variance.

In general, performance metrics can be divided into different classes:

Local metrics are typically defined for every pixel of the depth map and independent of each other. Image processing and analysis has the advantage that these individual observations have a clear and descriptive meaning. Also the spatial structure of these observations has a meaning in itself so it is advantageous to use error metrics which preserve this spatial information. An example for a local metric would be the per pixel endpoint error.

Global metrics which includes the classical mean endpoint error are often derived from local metrics by statistical analysis. This could be standard deviation of the error, higher order momenta or more subjective metrics like the apparent smoothness (with smoothness deliberately left without a strict definition). If a local metric is used, there is practically no reason not to include some simple statistics for this property. Mean, standard deviation, median and quantiles are rather fast to compute and can give additional insights.

Another classification scheme would be the distinction between **direct metrics** which can be computed directly when the ground truth is given (or sometimes even independently of the GT), or **derived metrics** which need more or less extensive postprocessing before they can be applied. Examples would be the fit between polygon meshes which are based on the measured depth data. This is usually more application dependent and may be expressed in terms of the requirements presented in Section 2.

The following enumeration is in no way exhaustive as each application may define its own error metric. These metrics should be considered as a guideline for low-level examinations.

Endpoint error / Bias / Accuracy. The most basic error metric describes the absolute distance between a ToF pixel and the true depth. As ToF cameras are prone to systematic errors as well as high noise it can only give a rough estimate about the quality of a measurement.

Standard deviation / Variance / Precision. The expressiveness of this metric depends on whether it denotes the temporal or spatial deviation. Spatial variance is typically highly dependent on scene geometry and not very descriptive unless a flat wall with uniform depth was imaged. Temporal variance can be of interest when the light situation and material of the underlying pixel is of interest.

(Root) Mean Squared Error ((R)MSE). The RMSE is equal to the sum of the standard deviation and the endpoint error (or bias). It is quite popular due to its statistic properties but is otherwise not very descriptive.

Local curvature/Slope. Due to the various effects of ToF cameras, otherwise planar or piecewise planar objects such as walls or room corners may appear curved or slanted. Differences between the actual and measured surfaces can give insight into the magnitude of those effects. Before this metric can be evaluated a appropriate surface reconstruction must be applied to the depth data.

Edges. Depth and texture edges are considered significant information for many low-level vision and image processing tasks, albeit they may lack a proper definition. ToF depth is known to vary depending on the observed intensity even if the true depth is constant. This can lead to false depth edges in regions where there is actually only a texture edge present. A yet to be defined metric for edge quality could help distinguish between true and false depth edges. This metric would be even more useful for ground truth with labeled depth as well as texture edges.

5.2 Weak Ground Truth

In Computer Vision, the finite accuracy of ground truth is often not taken into consideration. This is fine as long as the GT has an accuracy of over an order of magnitude more everywhere compared to the application at hand. Often there are methods that can create reliable GT data for only certain parts of the scene whereas other parts are erroneous. If those errors can be quantified and the regions clearly be localized such weak GT can still be feasible to use for a quick evaluation, as long as the analysis is mathematically/statistically sound.

5.2.1 Basic Example for Generation of Weak Ground Truth

A plane is a very basic example for ground truth that can be used to evaluate several kinds of distance errors like temperature drift, intensity related errors or distance offsets as well as noise characteristics. Given a carefully chosen setup, other influences such as multi-path or motion artifacts can be eliminated or reduced.

The weak ground truth for a plane can be generated by detecting and evaluating checkerboard corners in the amplitude image. From the detected points and the knowledge of the checkerboard geometry, the position of the checkerboard can be derived. For an ideally calibrated pin-hole camera, and a single corner distance measured in the center, the approximate distance error can be estimated using the intercept theorem:

$$e = \frac{2d_p e_d}{\hat{d}_p^2 - 2e_d \hat{d}_p} f. \quad (1)$$

with d_p being the corner distance, f is the focal length, distance between focal point and checkerboard t , $\hat{d}_p = \frac{d_p}{t} f$ is the according distance of the corners projected onto the sensor and e_d is the detection error with respect to distance on the chip (has to be divided by the pixel pitch to get the detection error in pixels).

For the technical specifications of a MESA SR 4000 camera, a typical corner detection error of a 10th of a pixel (for each corner) a checkerboard at $t = 1$ m distance and with a corner distance of $d_p = 85$ mm, the maximum error in the estimated checkerboard distance is approx. $e = 0.73$ mm. For a checkerboard with n corners the total estimation error decreases to $\frac{e}{\sqrt{n}}$ (assuming normally distributed error statistics). Given a sufficiently large number of corners in the checkerboard, the estimation error can be well below the typical ToF depth error which is in the centimeter range.

5.2.2 Weak Ground Truth of Another Modality

Even if the reference modality has an accuracy of the same order of magnitude as the ToF data, it can still be useful if a error distribution or a confidence score is known. An example would be, if somebody uses a Kinect Fusion scan to assess accuracy of a ToF denoising algorithm. If the reference modality has a error

distribution that is known, The likelihood of the ToF Data given the probability distribution of the reference data can be easily computed. An increasing number of algorithms offer confidence measures [3,24] between 0 and 1 without any further probabilistic interpretation. As low confidence data points are not an error measure, low confidence data may still be interesting. We therefore propose to borrow from sparsification plots used for confidence measures [24]. Normal sparsification plots are concerned with the evaluation of confidence in presence of ground truth. The endpoint error is plotted as a function of removing points with a confidence lower than a certain threshold. In our case we plot the error metric between the ToF data and the reference weak ground truth as a function of the confidence of the reference data.

6 Best Practices

This last part is intended to be a tutorial section explaining how to create good ground truth data which can be comparable with other ground truth datasets. We will discuss various supplemental results that should be made available to facilitate such comparisons. Many points in this section may appear obvious but experience has shown that often data which is missing or was mislabeled during the measurement process is in fact crucial for any further research effort.

Generally speaking, it is advisable to capture as much data and metadata as possible. With data we designate individual frames from the camera, while metadata designates everything else, be it recorded automatically or by additional experiments or setup procedures.

Data and Metadata

The metadata for a ground truth set should at least include:

Temperature The depth output of a ToF camera can be highly temperature dependent. The camera should therefore have reached a steady state and temperature changes due to environmental conditions (sunlight etc.) should either be reduced or at least be recorded.

Light situation This does include the documentation of additional light sources apart from the cameras own as well as significant external changes (e.g. due to cloud movement etc.)

Camera parameters If possible, the camera should be calibrated and camera matrices as well as lens distortion parameters be provided. For lenses with fixed focus and aperture these values can be static over a long time but for otherwise the calibration should be considered invalid each time the lens is touched. If multiple cameras are used (e.g. in a stereo setup) the external calibration should be measured and treated in a similar manner. As ToF systems are often used in image fusion approaches conjunction with other imaging systems this requirement is generally observed in existing datasets. This also includes all the presented example datasets in Section 4.2.

Materials What type of materials are visible in the scene? It should also be taken in to account, that the reflection and illumination behavior of certain materials may be radically different under infrared lighting. Glass for example may not be longer transparent while wood grain can appear much more distinct, etc.

Sensor settings This includes integration times, framerates, gain, etc. If it is adjustable it should be documented.

Software Which capture software and which version of it was used? If available the source code as well as any configuration files should be supplied. Data can be subtly different when seemingly unrelated program parameters or e.g. the capture drivers change.

A written scene description An image may say more than a thousand words but it may not always be as obvious as it seems. If many scenes with only slightly different parameters were captured, the motivation to do so could be included here.

Additional postprocessing or calibration data This may include the measured fixed pattern noise, data about sensor or light inhomogenities or calibration fits for depth calibrated cameras.

Regarding the images, raw data if available should always be saved alongside the derived depth maps. This is important as denoising or postprocessing on raw phase images is a ongoing field of study (See Section 2) and research will benefit from access to this data.

For static scenes it is advisable to capture multiple frames to allow investigation of e.g. temporal noise and to reduce the error by averaging. For dynamic scenes noting the approximate speed of the camera and scene objects allows for sanity checks.

Typical Errors

The following points are easily avoidable but may lead to inferior results or deteriorated data when not detected early:

Under/Overexposure. Depth data on overexposed pixels may be completely incorrect. Depth on pixels with a too low amplitude on the other hand may be more accurate but is prone to severely increased noise and should be considered unreliable.

Recording at different temperatures. The PMD CamCube camera for example needs to run for about 20 minutes before it reaches a temperature steady-state. During this warm-up-time the measured depths may change significantly.

Low-frequency light modulation. Often caused by fluorescent lamps. May not influence the measured depth but the intensities between adjacent frames.

Depth-of-field. Often neglected due to the rather low resolution of the most common ToF cameras and their use of fixed focus optics. Out-of-focus recordings may have increased artifacts based on flying pixels. Edge quality may also be effected.

Ignoring imaging modalities. Assumptions about material behavior (e.g. lambertian or specular reflectance) may not hold under infrared illumination. Black ink on regular paper for example may appear brighter than the surrounding white paper.

Incorrect interpretation of depth data. The depth maps produced by most ToF cameras represent radial depth, which is the distance of the point to the camera center. The depth from triangulation based methods (e.g. stereo) is generally given as z-depth, the orthogonal distance from the sensor plane. With known camera intrinsics both representations can be converted into each other to make them comparable.

Occlusion of depth maps. For ground truth acquired by means of measuring, the fields of view of the different sensors should overlap as much as possible. Small deviations in the scan positions can lead to occlusion in the depth maps, resulting in potentially sparse ground truth.

Multi path from unobserved walls. Often multi path effects from objects just outside the camera frustum can be observed, even though the object itself is not imaged by the camera.

7 Conclusion

While its creation is in no way easy we consider good ground truth to be a necessity for advancements in the field of ToF imaging. Both applicants as well as developers of ToF centric algorithms need ways to interpret their results. Starting with clear requirement definitions the performance of ToF systems and ToF based applications need to be evaluated. We have presented methods to create ground truth data, both *strong* and *weak* as well as metrics to compare and evaluate errors. We hope that with the help of these guidelines additional and more detailed ground truth datasets will soon be made available to the research community.

Acknowledgements. This work is part of a joint research project with the Filmakademie Baden-Württemberg, Institute of Animation. It is co-funded by the Intel Visual Computing Institute and under grant 2-4225.16/380 of the ministry of economy Baden-Württemberg as well as further partners Unexpected, Pixomondo, ScreenPlane, Bewegte Bilder and Tridality . The content is under sole responsibility of the authors.

References

1. Boehler, W., Vicent, M.B., Marbs, A.: Investigating laser scanner accuracy. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34(Part 5), 696–701 (2003)
2. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Computer Vision and Pattern Recognition (CVPR), Providence, USA (June 2012)

3. Reynolds, M., Doboš, J., Peel, L., Weyrich, T., Brostow, G.J.: Capturing time-of-flight data with confidence. In: CVPR (2011)
4. Clark, J., Robson, S.: Accuracy of measurements made with a cyrax 2500 laser scanner against surfaces of known colour. *Survey Review* 37(294), 626–638 (2004)
5. Soudarissanane, S., Lindenbergh, R., Menenti, M., Teunissen, P.: Incidence angle influence on the quality of terrestrial laser scanning points. In: ISPRS Workshop Laserscanning (2009)
6. Lichti, D.D.: Error modelling, calibration and analysis of an am-cw terrestrial laser scanner system. *ISPRS Journal of Photogrammetry and Remote Sensing* 61(5), 307–324 (2007)
7. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, vol. 7, pp. 127–136 (2011)
8. Izadi, S., Newcombe, R.A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A.J., Fitzgibbon, A.: KinectFusion: Real-time dynamic 3D surface reconstruction and interaction. In: ACM SIGGRAPH 2011 Talks, p. 23. ACM (2011)
9. Khoshelham, K.: Accuracy analysis of kinect depth data. In: ISPRS Workshop Laser Scanning, vol. 38, p. 1 (2011)
10. Meister, S., Izadi, S., Kohli, P., Hämmerle, M., Rother, C., Kondermann, D.: When can we use kinectfusion for ground truth acquisition? In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshops & Tutorials (2012)
11. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Robotics-DL Tentative, International Society for Optics and Photonics, pp. 586–606 (1992)
12. Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000)
13. Maier-Hein, L., Schmidt, M., Franz, A.M., dos Santos, T.R., Seitel, A., Jähne, B., Fitzpatrick, J.M., Meinzer, H.P.: Accounting for anisotropic noise in fine registration of time-of-flight range data with high-resolution surface data. In: Jiang, T., Navab, N., Pluim, J.P.W., Viergever, M.A. (eds.) MICCAI 2010, Part I. LNCS, vol. 6361, pp. 251–258. Springer, Heidelberg (2010)
14. Schmidt, M.: Analysis, Modeling and Dynamic Optimization of 3D Time-of-Flight Imaging Systems. PhD thesis, University of Heidelberg (2011)
15. Meister, S., Kondermann, D.: Real versus realistically rendered scenes for optical flow evaluation. In: Proceedings of 14th ITG Conference on Electronic Media Technology (2011)
16. Keller, M., Kolb, A.: Real-time simulation of time-of-flight sensors. *J. Simulation Practice and Theory* 17, 967–978 (2009)
17. Research, L.: Optics software for layout and optimization (oslo), <http://www.lambdaires.com>
18. Zemax, R.: Zemax, optical design software, <http://www.radiantzemax.com>
19. Schmidt, M., Jähne, B.: A physical model of time-of-flight 3D imaging systems, including suppression of ambient light. In: Kolb, A., Koch, R. (eds.) Dyn3D 2009. LNCS, vol. 5742, pp. 1–15. Springer, Heidelberg (2009)
20. Schmidt, M.: Analysis, Modeling and Dynamic Optimization of 3D Time-of-Flight Imaging Systems. PhD thesis, University of Heidelberg (2011)
21. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)

22. Dal Mutto, C., Zanuttigh, P., Mattocchia, S., Cortelazzo, G.: Locally consistent toF and stereo data fusion. In: Fusiello, A., Murino, V., Cucchiara, R. (eds.) ECCV 2012 Ws/Demos, Part I. LNCS, vol. 7583, pp. 598–607. Springer, Heidelberg (2012)
23. Zhang, L., Curless, B., Seitz, S.M.: Spacetime stereo: Shape recovery for dynamic scenes. In: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II–367. IEEE (2003)
24. Haeusler, R., Nair, R., Kondermann, D.: (Ensemble learning for confidence measures in stereo vision)

Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion

Maik Keller
pmdtechnologies

Damien Lefloch
University of Siegen

Martin Lambers
University of Siegen

Shahram Izadi
Microsoft Research

Tim Weyrich
University College London

Andreas Kolb
University of Siegen

Abstract

Real-time or online 3D reconstruction has wide applicability and receives further interest due to availability of consumer depth cameras. Typical approaches use a moving sensor to accumulate depth measurements into a single model which is continuously refined. Designing such systems is an intricate balance between reconstruction quality, speed, spatial scale, and scene assumptions. Existing online methods either trade scale to achieve higher quality reconstructions of small objects/scenes. Or handle larger scenes by trading real-time performance and/or quality, or by limiting the bounds of the active reconstruction. Additionally, many systems assume a static scene, and cannot robustly handle scene motion or reconstructions that evolve to reflect scene changes. We address these limitations with a new system for real-time dense reconstruction with equivalent quality to existing online methods, but with support for additional spatial scale and robustness in dynamic scenes. Our system is designed around a simple and flat point-based representation, which directly works with the input acquired from range/depth sensors, without the overhead of converting between representations. The use of points enables speed and memory efficiency, directly leveraging the standard graphics pipeline for all central operations; i.e., camera pose estimation, data association, outlier removal, fusion of depth maps into a single denoised model, and detection and update of dynamic objects. We conclude with qualitative and quantitative results that highlight robust tracking and high quality reconstructions of a diverse set of scenes at varying scales.

1. Introduction and Background

Online 3D reconstruction receives much attention as inexpensive depth cameras (such as the Microsoft Kinect, Asus Xtion or PMD CamBoard) become widely available. Compared to offline 3D scanning approaches, the ability to obtain reconstructions in *real time* opens up a variety of interactive applications including: augmented reality (AR) where real-world geometry can be fused with 3D graphics and rendered live to the user; autonomous guidance for robots to recon-

struct and respond rapidly to their environment; or even to provide immediate feedback to users during 3D scanning.

The first step of the reconstruction process is to acquire depth measurements either using sequences of regular 2D images (e.g. [19]), or with active sensors, such as laser scanners or depth cameras, based on triangulation or time-of-flight (ToF) techniques. Unlike methods that focus on reconstruction from a complete set of 3D points [5, 7], online methods require *fusion* of many overlapping depth maps into a single 3D representation that is continuously refined. Typically methods first find correspondences between depth maps (data association) and register or *align* depth maps to estimate the sensor's egomotion [1, 24]. The fusion method typically involves removal of outliers e.g. by visibility testing between depth maps [16], observing freespace violations [2], or photo-consistency [12], and merging of measurements into the global model, e.g. using simple weighted averaging [2] or more costly spatial regularization [25, 12].

Recent online systems [6, 11] achieve high-quality results by adopting the volumetric fusion method of Curless and Levoy [2]. This approach supports incremental updates, exploits redundant samples, makes no topological assumptions, approximates sensor uncertainty, and fusion is performed using a simple weighted average. For active sensors, this method produces very compelling results [2, 9, 6, 11]. The drawbacks are the *computational overheads* needed to continuously transition between different data representations: Where point-based input is converted to a continuous implicit function, discretized within a regular grid data structure, and converted back to an (explicit) form using expensive polygonization [10] or raycasting [14] methods. As well as the *memory overheads* imposed by using a regular voxel grid, which represents both empty space and surfaces densely, and thus greatly limits the size of the reconstruction volume.

These memory limitations have led to *moving-volume* systems [17, 23], which still operate on a very restricted volume, but free-up voxels as the sensor moves; or hierarchical volumetric data structures [26], which incur additional computational and data structure complexity for only limited gains in terms of spatial extent.

Beyond volumetric methods, simpler representations have

also been explored. Height-map representations [3] work with compact data structures allowing scalability, especially suited for modeling large buildings with floors and walls, since these appear as clear discontinuities in the height-map. Multi-layered height-maps support reconstruction of more complex 3D scenes such as balconies, doorways, and arches [3]. While these methods support compression of surface data for simple scenes, the 2.5D representation fails to model complex 3D environments efficiently.

Point-based representations are more amenable to the input acquired from depth/range sensors. [18] used a point-based method and custom structured light sensor to demonstrate in-hand online 3D scanning. Online model rendering required an intermediate volumetric data structure. Interestingly, an offline volumetric method [2] was used for higher quality final output, which nicely highlights the computational and quality trade-offs between point-based and volumetric methods. [22] took this one step further, demonstrating higher quality scanning of small objects using a higher resolution custom structured light camera, sensor drift correction, and higher quality surfel-based [15] rendering. These systems however focus on single small object scanning. Further, the sensors produce less noise than consumer depth cameras (due to dynamic rather than fixed structured light patterns), making model denoising less challenging.

Beyond reducing computational complexity, point-based methods lower the memory overhead associated with volumetric (regular grid) approaches, as long as overlapping points are merged. Such methods have therefore been used in larger sized reconstructions [4, 20]. However, a clear trade-off becomes apparent in terms of scale versus speed and quality. For example, [4] allow for reconstructions of entire floors of a building (with support for loop closure and bundle adjustment), but frame rate is limited (~ 3 Hz) and an unoptimized surfel map representation for merging 3D points can take seconds to compute. [20] use a multi-level surfel representation that achieves interactive rates (~ 10 Hz) but require an intermediate octree representation which limits scalability and adds computational complexity.

In this paper we present an online reconstruction system also based around a flat, point-based representation, rather than any spatial data structure. A key contribution is that our system is memory-efficient, supporting spatially extended reconstructions, but without trading reconstruction quality or frame rate. As we will show, the ability to directly render the representation using the standard graphics pipeline, without converting between multiple representations, enables efficient implementation of all central operations, i.e., camera pose estimation, data association, denoising and fusion through data accumulation, and outlier removal.

A core technical contribution is leveraging a fusion method that closely resembles [2] but removes the voxel grid all-together. Despite the lack of a spatial data structure,

our system still captures many benefits of volumetric fusion, with competitive performance and quality to previous online systems, allowing for accumulation of denoised 3D models over time that exploit redundant samples, model measurement uncertainty, and make no topological assumptions.

The simplicity of our approach allows us to tackle another fundamental challenge of online reconstruction systems: the assumption of a static scene. Most previous systems make this assumption or treat dynamic content as outliers [18, 22]; only KinectFusion [6] is at least capable of reconstructing moving objects in a scene, provided a static pre-scan of the background is first acquired. Instead, we leverage the immediacy of our representation to not only robustly segment dynamic objects in the scene, which greatly improves the robustness of the camera pose estimation, but also to continuously update the global reconstruction, regardless of whether objects are added or removed. Our approach is further able to detect when a moving object has become static or a stationary object has become dynamic.

The ability to support reconstructions at quality comparable to state-of-the-art, without trading real-time performance, with the addition of extended spatial scale and support for dynamic scenes provides unique capabilities over prior work. We conclude with results from reconstructing a variety of static and dynamic scenes of different scales, and an experimental comparison to related systems.

2. System Overview

Our high-level approach shares commonalities with the existing incremental reconstruction systems (presented previously): we use samples from a moving depth sensor; first preprocess the depth data; then estimate the current six degree-of-freedom (6DoF) pose of sensor relative to the scene; and finally use this pose to convert depth samples into a unified coordinate space and fuse them into an accumulated global model. Unlike prior systems, we adopt a purely point-based representation throughout our pipeline, carefully designed to support data fusion with quality comparable to online volumetric methods, whilst enabling real-time reconstructions at extended scales and in dynamic scenes.

Our choice of representation makes our pipeline extremely amenable to implementation using commodity graphics hardware. The main system pipeline as shown in Fig. 1 is based on the following steps:

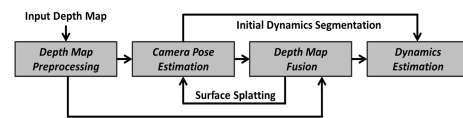


Figure 1. Main system pipeline.

Depth Map Preprocessing Using the intrinsic parameters of the camera, each input depth map from the depth

sensor is transformed into a set of 3D points, stored in a 2D *vertex map*. Corresponding normals are computed from central-differences of the denoised vertex positions, and per-point radii are computed as a function of depth and gradient (stored in respective *normal* and *radius* maps).

Depth Map Fusion Given a valid camera pose, input points are fused into the *global model*. The global model is simply a list of 3D points with associated attributes. Points evolve from *unstable* to *stable* status based on the confidence they gathered (essentially a function of how often they are observed by the sensor). Data fusion first *projectively associates* each point in the input depth map with the set of points in the global model, by rendering the model as an *index map*. If corresponding points are found, the most reliable point is merged with the new point estimate using a weighted average. If no reliable corresponding points are found, the new point estimate is added to the global model as an unstable point. The global model is cleaned up over time to remove outliers due to visibility and temporal constraints. Sec. 4 discusses our point-based data fusion in detail.

Camera Pose Estimation All established (high confidence) model points are passed to the visualization stage, which reconstructs dense surfaces using a surface splatting technique (see Sec. 5). To estimate the 6DoF camera pose, the model points are projected from the previous camera pose, and a pyramid-based dense iterative closest point (ICP) [11] alignment is performed using this rendered *model map* and input depth map. This provides a new relative rigid 6DoF transformation that maps from the previous to new global camera pose. Pose estimation occurs prior to data fusion, to ensure the correct projection during data association.

Dynamics Estimation A key feature of our method is automatic detection of dynamic changes in the scene, to update the global reconstruction and support robust camera tracking. Dynamic objects are initially indicated by outliers in point correspondences during ICP. Starting from these areas, we perform a point-based region growing procedure to identify dynamic regions. These regions are excluded from the camera pose estimate, and their corresponding points in the global model are reset to *unstable* status, leading to a natural propagation of scene changes into our depth map fusion. For more detail, see Sec. 6.

3. Depth Map Preprocessing

We denote a 2D pixel as $\mathbf{u} = (x, y)^T \in \mathbb{R}^2$. $\mathcal{D}_i \in \mathbb{R}$ is the raw depth map at time frame i . Given the intrinsic camera calibration matrix \mathbf{K}_i , we transform \mathcal{D}_i into a corresponding vertex map \mathcal{V}_i , by converting each depth sample $\mathcal{D}_i(\mathbf{u})$ into a vertex position $\mathbf{v}_i(\mathbf{u}) = \mathcal{D}_i(\mathbf{u})\mathbf{K}_i^{-1}(\mathbf{u}^T, 1)^T \in \mathbb{R}^3$ in camera space. A corresponding normal map \mathcal{N}_i is determined from central-differences of the vertex map. A copy of the depth map (and hence associated vertices and normals) are

also denoised using a *bilateral filter* [21] (for camera pose estimation later).

The 6DoF camera pose transformation comprises of rotation ($\mathbf{R}_i \in \mathbb{SO}_3$) matrix and translation ($\mathbf{t}_i \in \mathbb{R}^3$) vector, computed per frame i as $\mathbf{T}_i = [\mathbf{R}_i, \mathbf{t}_i] \in \mathbb{SE}_3$. A vertex is converted to global coordinates as $\mathbf{v}_i^g = \mathbf{T}_i \mathbf{v}_i$. The associated normal is converted to global coordinates as $\mathbf{n}_i^g(\mathbf{u}) = \mathbf{R}_i \mathbf{n}_i(\mathbf{u})$. Multi-scale pyramids \mathcal{V}_i^l and \mathcal{N}_i^l are computed from vertex and normal maps for hierarchical ICP, where $l \in \{0, 1, 2\}$ and $l = 0$ denotes the original input resolution (e.g. 640×480 for Kinect or 200×200 for PMD CamBoard).

Each input vertex also has an associated radius $r_i(\mathbf{u}) \in \mathbb{R}$ (collectively stored in a radius map $\mathcal{R}_i \in \mathbb{R}$), determined as in [22]. To prevent arbitrarily large radii from oblique views, we clamp radii for grazing observations exceeding 75° .

In the remainder, we omit time frame indices i for clarity, unless we refer to two different time frames at once.

4. Depth Map Fusion

Our system maintains a single global model, which is simply an unstructured set of points \tilde{P}_k each with associated position $\tilde{\mathbf{v}}_k \in \mathbb{R}^3$, normal $\tilde{\mathbf{n}}_k \in \mathbb{R}^3$, radius $\tilde{r}_k \in \mathbb{R}$, confidence counter $\tilde{c}_k \in \mathbb{R}$, and time stamp $\tilde{t}_k \in \mathbb{N}$, stored in a flat array indexed by $k \in \mathbb{N}$.

New measurements \mathbf{v} are either added as or merged with unstable points, or they get merged with stable model points. Merging \mathbf{v} with a point \tilde{P}_k in the global model increases the confidence counter \tilde{c}_k . Eventually an unstable point changes its status to stable: points with $\tilde{c}_k \geq c_{\text{stable}}$ are considered stable (in practice $c_{\text{stable}} = 10$). In specific temporal or geometric conditions, points are removed from the global model.

4.1. Data Association

After estimation of the camera pose of the current input frame (see Sec. 5), each vertex \mathbf{v}^g and associated normal and radius are integrated into the global model.

In a first step, for each valid vertex \mathbf{v}^g , we find potential corresponding points on the global model. Given the inverse global camera pose \mathbf{T}^{-1} and intrinsics \mathbf{K} , each point \tilde{P}_k in the global model can be projected onto the image plane of the current physical camera view, where the respective point index k is stored: we render all model points into a sparse *index map* \mathcal{I} . Unlike the splat-based dense surface reconstruction renderer used in other parts of our pipeline (see Sec. 5), this stage renders each point index into a single pixel to reveal the actual surface sample distribution.

As nearby model points may project onto the same pixel, we increase the precision of \mathcal{I} by supersampling, representing \mathcal{I} at 4×4 the resolution of the input depth map. We start identifying model points near $\mathbf{v}^g(\mathbf{u})$ by collecting point indices within the 4×4 -neighborhood around each input pixel location \mathbf{u} (suitably coordinate-transformed from \mathcal{D} to \mathcal{I}).

Amongst those points, we determine a single corresponding model point by applying the following criteria:

1. Discard points larger than $\pm\delta_{\text{depth}}$ distance from the viewing ray $v^s(\mathbf{u})$ (the sensor line of sight), with δ_{depth} adapted according to sensor uncertainty (i.e. as a function of depth for triangulation-based methods [13]).
2. Discard points whose normals have an angle larger than δ_{norm} to the normal $n^s(\mathbf{u})$. (We use $\delta_{\text{norm}} = 20^\circ$.)
3. From the remaining points, select the ones with the highest confidence count.
4. If multiple such points exist, select the one closest to the viewing ray through $v^s(\mathbf{u})$.

4.2. Point Averaging with Sensor Uncertainty

If a corresponding model point \bar{P}_k is found during data association, this is averaged with the input vertex $v^s(\mathbf{u})$ and normal $n^s(\mathbf{u})$ as follows:

$$\bar{v}_k \leftarrow \frac{\bar{c}_k \bar{v}_k + \alpha v^s(\mathbf{u})}{\bar{c}_k + \alpha}, \quad \bar{n}_k \leftarrow \frac{\bar{c}_k \bar{n}_k + \alpha n^s(\mathbf{u})}{\bar{c}_k + \alpha}, \quad (1)$$

$$\bar{c}_k \leftarrow \bar{c}_k + \alpha, \quad \bar{t}_k \leftarrow t, \quad (2)$$

where t is a new time stamp. Our weighted average is distinct from the original KinectFusion system [11], as we introduce an explicit sample confidence α . This applies a Gaussian weight on the current depth measurement as $\alpha = e^{-r^2/2\sigma^2}$, where r is the normalized radial distance of the current depth measurement from the camera center, and $\sigma = 0.6$ is derived empirically. This approach weights measurements based on the assumption that measurements closer to the sensor center will increase in accuracy [2]. As shown in Fig. 2, modeling this sensor uncertainty leads to higher quality denoising.

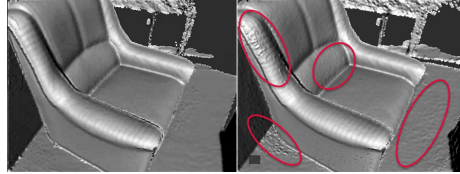


Figure 2. Weighted averaging of points using our method (left) and the method of [11] (right).

Since the noise level of the input measurement increases as a function of depth [13], we apply Eqs. (1) only if the radius of the new point is not significantly larger than the radius of the model point, i.e., if $r(\mathbf{u}) \leq (1 + \delta_r)\bar{r}$; we empirically chose $\delta_r = 1/2$. This ensures that we always refine details, but never coarsen the global model. We apply the time stamp and the confidence counter updates according to Eqs. (2) irrespectively.

If no corresponding model point has been identified, a new unstable point is added to the global model with $\bar{c}_k = \alpha$, containing the input vertex, normal and radius.

4.3. Removing Points

So far we have merged or added new measurements to the global model. Another key step is to remove points from our global model due to various conditions:

1. Points that remain in the unstable state for a long time are likely outliers or artifacts from moving objects and will be removed after t_{max} time steps.
2. For stable model points that are merged with new data, we remove all model points that lie in front of these newly merged points, as these are free-space violations. To find these points to remove, we use the index map again and search the neighborhood around the pixel location that the merged point projects onto¹. This is similar in spirit to the free-space carving method of [2], but avoids expensive voxel space traversal.
3. If after averaging, a stable point has neighboring points (identified again via the index map) with very similar position and normal and their radii overlap, then we merge these redundant neighboring points to further simplify the model.

Points are first marked to be removed from \bar{P}_k , and in a second pass, the list is sorted (using a fast radix sort implementation), moving all marked points to the end, and finally items deleted.

5. Camera Pose Estimation

Following the approach of KinectFusion [11], our camera pose estimation uses dense hierarchical ICP to align the bilateral filtered input depth map \mathcal{D}_i (of the current frame i) with the reconstructed model by rendering the model into a virtual depth map, or *model map*, $\hat{\mathcal{D}}_{i-1}$, as seen from the previous frame's camera pose T_{i-1} . We use 3 hierarchy levels, with the finest level at the camera's resolution; unstable model points are ignored. The registration transformation provides the relative change from T_{i-1} to T_i .

While KinectFusion employs raycasting of the (implicit) voxel-based reconstruction, we render our explicit, point-based representation using a simple surface-splatting technique: we render overlapping, disk-shaped *surface splats* that are spanned by the model point's position \bar{v} , radius \bar{r} and orientation \bar{n} . Unlike more refined surface-splatting techniques, such as EWA Surface Splatting [27], we do not perform blending and analytical prefiltering of splats but trade local surface reconstruction quality for performance by simply rendering opaque splats.

We use the same point-based renderer for user feedback, but add Phong shading of surface splats, and also overlay the dynamic regions of the input depth map.

¹Backfacing points that are close to the merged points remain protected—such points may occur in regions of high curvature or around thin geometry in the presence of noise and slight registration errors. Furthermore, we protect points that would be consistent with direct neighbor pixels in \mathcal{D} , to avoid spurious removal of points around depth discontinuities.

6. Dynamics Estimation

The system as described above already has limited support for dynamic objects, in that unstable points must gain confidence to be promoted to stable model points, and so fast moving objects will be added and then deleted from the global model. In this section we describe additional steps that lead to an explicit classification of observed points as being part of a dynamic object. In addition, we aim at segmenting entire objects whose surface is partially moving and remove them from the global point model.

We build upon an observation by Izadi et al. [6]: when performing ICP, failure of data association to find model correspondences for input points is a strong indication that these points are depth samples belonging to dynamic objects. Accordingly, we retrieve this information by constructing an ICP status map \mathcal{S} (with elements $s_i(\mathbf{u})$) that encodes for each depth sample the return state of ICP's search for a corresponding model point in the data association step:

- `no_input`: $v_k(\mathbf{u})$ is invalid or missing.
- `no_cand`: No stable model points in proximity of $v_k(\mathbf{u})$.
- `no_corr`: Stable model points in proximity of, but no valid ICP correspondence for $v_k(\mathbf{u})$.
- `corr`: Otherwise ICP found a correspondence.

Input points marked as `no_corr` are a strong initial estimate of parts of the scene that move independent of camera motion, i.e. dynamic objects in the scene.

We use these points to seed our segmentation method based on hierarchical region growing (see below). It creates a *dynamics map* \mathcal{X} , storing flags $x_i(\mathbf{u})$, that segments the current input frame into static and dynamic points. The region growing aims at marking complete objects as dynamic even if only parts of them actually move. (Note that this high-level view on dynamics is an improvement over the limited handling of dynamics in previous approaches, e.g., [6].)

In the depth map fusion stage, model points that are merged with input points marked as dynamic are potentially demoted to unstable points using the following rule:

$$\text{if } x_i(\mathbf{u}) \wedge \bar{c}_k \geq c_{\text{stable}} + 1 \text{ then } \bar{c}_k \leftarrow 1 \quad (3)$$

Thus, the state change from static to dynamic is reflected immediately in the model. A critical aspect is the offset of +1 in Eq. (3): it ensures that any dynamic point that sufficiently grew in confidence (potentially because it is now static) is allowed to be added to the global model for at least one iteration; otherwise, a surface that has once been classified as dynamic would never be able to re-added to the global model, as it would always be inconsistent with the model, leading to `no_corr` classification.

For the bulk of the time, however, dynamic points remain *unstable* and as such are not considered for camera pose estimation (see Sec. 5), which greatly improves accuracy and robustness of T .

Hierarchical Region Growing The remainder of this section explains the region growing-based segmentation approach that computes the map \mathcal{X} .

The goal is essentially to find connected components in \mathcal{D} . In the absence of explicit neighborhood relations in the point data, we perform region growing based on point attribute similarity. Starting from the seed points marked in \mathcal{X} , we agglomerate points whose position and normal are within given thresholds of vertex $v(\mathbf{u})$ and normal $\mathbf{n}(\mathbf{u})$ of a neighbor with $x(\mathbf{u}) = \text{true}$.

To accelerate the process, we start at a downsampled \mathcal{X}^2 , and repeatedly upsample until we reach $\mathcal{X}^0 = \mathcal{X}$, each time resuming region growing. (We reuse the input pyramids built for camera pose estimation.)

We improve robustness to camera noise and occlusions by removing stray `no_corr` points through morphological erosion at the coarsest pyramid level \mathcal{X}^2 after initializing it from \mathcal{S} . This also ensures that \mathcal{X}^2 covers only the inner region of dynamic objects.

7. Results

We have tested our system on a variety of scenes (see Table 1). Fig. 3 shows a synthetic scene *Sim*. We generated rendered depth maps for a virtual camera rotating around

	#frames input/processed (fps in./proc.)	#model- points	Avg. timings [ms]		
			ICP	Dyn- Seg.	Fusion
Sim	950/950 (15/15)	467,200	18.90	2.03	11.50
Flower- pot	600/480 (30/24)	496,260	15.87	1.90	6.89
Teapot	1000/923 (30/27)	191,459	15.20	1.60	5.56
Large Office	11892/6704 (30/17)	4,610,800	21.75	2.39	13.90
Moving Person	912/623 (30/20)	210,500	15.92	3.23	16.61
Ball- game	1886/1273 (30/21)	350,940	16.74	3.15	17.66
PMD	4101/4101 (27/27)	280,050	10.70	0.73	3.06

Table 1. Results from test scenes obtained on a PC equipped with an Intel i7 8-core CPU and an NVidia GTX 680 GPU. Input frames have a size of 640×480 pixels, except for the *PMD* scene which uses a frame size of 200×200 .

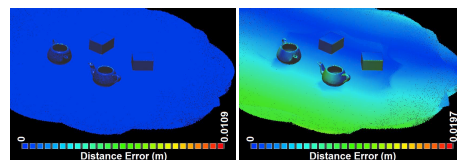


Figure 3. The synthetic scene *Sim*. Left: error in final global model based on ground truth camera transformations. Right: final error based on ICP pose estimation².

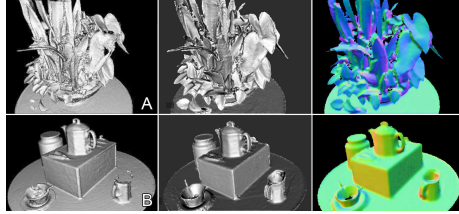


Figure 4. The scenes *Flowerpot* (top row) and *Teapot* (bottom row). A and B show reconstruction results of the original KinectFusion system. The other images show our method (middle: phong-shaded surfels, right: model points colored with surface normals).

this scene and used these as input to our system. This gave us ground truth camera transformations T_i^{GT} and ground truth scene geometry. Using T_i^{GT} , the points in the resulting global model have a mean position error of 0.019 mm. This demonstrates only minimal error for our point-based data fusion approach. The camera transformations T_i obtained from ICP have a mean position error of 0.87 cm and a mean viewing direction error of 0.1 degrees. This results in a mean position error of 0.20 cm for global model points.

The *Flowerpot* and *Teapot* scenes shown in Fig. 4 were recorded by Nguyen et al. [13]. Objects are placed on a turntable which is rotated around a stationary Kinect camera. Vicon is used for ground truth pose estimation of the Kinect, which are compared to ICP for our method and the original KinectFusion system Fig. 5

Fig. 6 shows that the number of global model points for these scenes remains roughly constant after one full turn of the turntable. This demonstrates that new points are not continuously added; and the global model is refined but kept compact. Note that one Kinect camera input frame provides up to 307,200 input points, but the total number of points in the final global teapot model is less than 300,000.

The *Large Office* scene shown in Fig. 7 consists of two rooms with a total spatial extent of approximately $10m \times 6m \times 2.5m$. A predefined volumetric grid with 32-bit voxels and 512 MB of GPU memory would result in a voxel size of more than 1 cm^3 . In contrast, our system does not define the scene extents in advance: the global model grows as required. Furthermore, it does not limit the size of representable details; Fig. 7 shows close-ups of details on the millimeter scale (e.g. the telephone keys). The 4.6 million global model points reported in Tab. 1 can be stored in 110 MB of GPU memory using 3 floating point values for the point position, 2 for the normalized point normal, 1 for the radius, and one extra byte for a confidence counter. Additionally, RGB colors can be stored for each global point, to texture the final model (see Fig. 7 far right). Rather than

²Rendered using CloudCompare, <http://www.danielgm.net/cc/>.

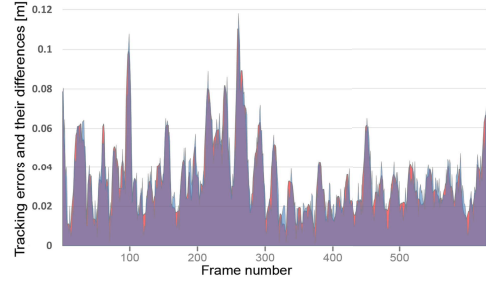


Figure 5. Tracking errors for the original KinectFusion system compared to our point-based approach. Tracking results were computed on the *Flowerpot* sequence, by subtracting Vicon ground truth data from the resulting per frame 3D camera position. For each system, error is computed as the absolute distance between the estimated camera position and the ground truth position (after aligning both coordinate spaces manually). Where the error of the original KinectFusion exceeds that of the new, the gap is colored blue. Where the error of our method exceeds the original, the gap is colored red. Note our method is similar in performance with the largest delta being $\sim 1\text{cm}$.

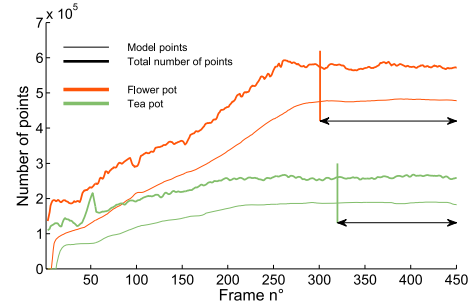


Figure 6. The number of global model points stored on the GPU plotted over time for the *Flowerpot* and *Teapot* scenes. Note after the completion of one full turn of the turntable, the number of points converges instead of continuously growing.

merge RGB samples, we simply store the last one currently.

In the *Moving Person* scene shown in Fig. 8, the person first sits in front of the sensor and is reconstructed before moving out of view. Since the moving person occupies much of the field of view, leaving only few reliable points for ICP, camera tracking fails with previous approaches (see e.g. Izadi et al. Fig. 8 [6]). Our system segments the moving person and ignores dynamic scene parts in the ICP stage, thereby ensuring robustness to dynamic motion.

The *Ballgame* scene shown in Fig. 9 shows two people playing with a ball across a table. Our region growing approach segments dynamics on the object level instead of just

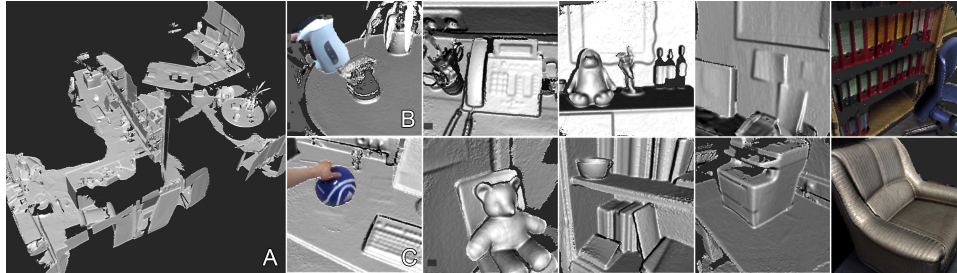


Figure 7. The *Large Office* scene, consisting of two large rooms and connecting corridors. A: overview; B and C: dynamically moving objects during acquisition; Note the millimeter scale of the phone’s keypad. Other close-ups are also shown (right column: RGB textured).

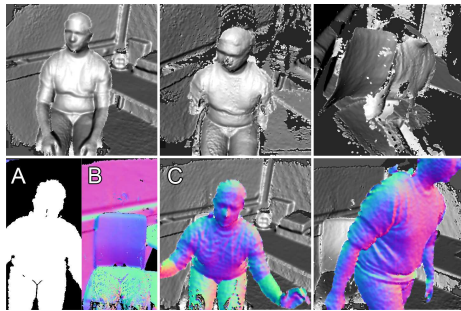


Figure 8. The *Moving Person* scene. Person sits on a chair, is reconstructed, and then moves. Dynamic parts occupy much of the field-of-view and cause ICP errors with previous approaches (top row). Segmenting the dynamics (A) and ignoring them during pose estimation (B) allows increased robustness (bottom row).

the point level: each person is recognized as dynamic even if only parts of their bodies are actually moving. Static objects that start moving are marked as dynamic and their model points are demoted to unstable status, while dynamic objects that stop moving eventually reach stable status in the global model when the observed points gain enough confidence.

Most scenes shown throughout this paper were recorded with a Microsoft Kinect camera in near mode, but our method is agnostic to the type of sensor used. Fig. 10 shows an example scene recorded with a PMD CamBoard ToF camera, which exhibits significantly different noise and error characteristics [8]. In this example, we used the per-pixel amplitude information provided by PMD sensors in the computation of the sample confidence α (see Sec. 4.2).

8. Conclusion

We have presented a new system for online 3D reconstruction which demonstrates new capabilities beyond the state-of-art. Our system has been explicitly designed to allow

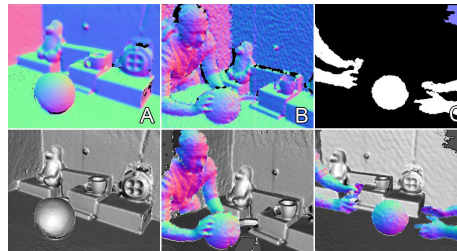


Figure 9. The *Ballgame* scene consists of two people moving a ball across a table. A: global model colored with surface normals; B: raw input data of the previously static ball being picked up; C: segmentation of dynamic parts; Bottom row: reconstructed result (model points + dynamic parts).



Figure 10. A: the *PMD* scene acquired with a PMD ToF camera. B and C: close-ups using per-pixel intensity values for coloring.

for a single point-based representation to be used throughout our pipeline, which closely fits the sensor input, and is amenable to rendering (for visualization and data association) through the standard graphics pipeline.

Despite the lack of a spatial data structure, our system still captures many benefits of volumetric fusion, allowing for accumulation of denoised 3D models over time that exploit redundant samples, model measurement uncertainty, and make no topological assumptions. This is achieved using a new *point-based fusion* method based on [2]. Reconstructions at this scale, quality, speed and with the ability to deal with scene motion and dynamic updates have yet to be demonstrated by other point-based methods, and are core contributions of our work.

There are many areas for future work. For example, whilst our system scales to large scenes, there is the additional possibility of adding mechanisms for streaming subset of points (from GPU to CPU) especially once they are significantly far away from the current pose. This would help increase performance and clearly the point-based data would be low overhead in terms of CPU-GPU bandwidth. Another issue is sensor drift, which we do not currently tackle, instead focusing on the data representation. Drift in larger environments can become an issue and remains an interesting direction for future work. Here again the point-based representation might be more amenable to correction after loop closure detection, rather than resampling a dense voxel grid.

Acknowledgements

This research has partly been funded by the German Research Foundation (DFG), grant GRK-1564 Imaging New Modalities, and by the FP7 EU collaborative project BEAMING (248620). We thank Jens Orthmann for his work on the GPU framework osgCompute.

References

- [1] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 14(2):239–256, 1992. [1](#)
- [2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. Comp. Graph. & Interact. Techn.*, pages 303–312, 1996. [1](#), [2](#), [4](#), [7](#)
- [3] D. Gallup, M. Pollefeys, and J.-M. Frahm. 3d reconstruction using an n-layer heightmap. In *Pattern Recognition*, pages 1–10. Springer, 2010. [2](#)
- [4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Robotics Research*, 31:647–663, Apr. 2012. [2](#)
- [5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proc. SIGGRAPH)*, 26(2), 1992. [1](#)
- [6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. ACM Symp. User Interface Softw. & Tech.*, pages 559–568, 2011. [1](#), [2](#), [5](#), [6](#)
- [7] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. EG Symp. Geom. Proc.*, 2006. [1](#)
- [8] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight cameras in computer graphics. *Computer Graphics Forum*, 29(1):141–159, 2010. [7](#)
- [9] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, et al. The digital michelangelo project: 3D scanning of large statues. In *Proc. Comp. Graph & Interact. Techn.*, pages 131–144, 2000. [1](#)
- [10] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987. [1](#)
- [11] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. IEEE Int. Symp. Mixed and Augm. Reality*, pages 127–136, 2011. [1](#), [3](#), [4](#)
- [12] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. IEEE Int. Conf. Comp. Vision*, pages 2320–2327, 2011. [1](#)
- [13] C. Nguyen, S. Izadi, and D. Lovell. Modeling Kinect sensor noise for improved 3D reconstruction and tracking. In *Proc. Int. Conf. 3D Imaging, Modeling, Processing, Vis. & Transmission*, pages 524–530, 2012. [4](#), [6](#)
- [14] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Proc. IEEE Vis.*, pages 233–238. IEEE, 1998. [1](#)
- [15] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proc. Conf. Comp. Graphics & Interact. Techn.*, pages 335–342, 2000. [2](#)
- [16] M. Pollefeys, D. Nistér, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, et al. Detailed real-time urban 3D reconstruction from video. *Int. J. Comp. Vision*, 78(2):143–167, 2008. [1](#)
- [17] H. Roth and M. Vona. Moving volume KinectFusion. In *British Machine Vision Conf.*, 2012. [1](#)
- [18] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):438–446, 2002. [2](#)
- [19] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Conf. Comp. Vision & Pat. Rec.*, volume 1, pages 519–528. IEEE, 2006. [1](#)
- [20] J. Stückler and S. Behnke. Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras. In *Proc. IEEE Int. Conf. Multisensor Fusion & Information Integration*, pages 162–167, 2012. [2](#)
- [21] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. Int. Conf. Computer Vision*, pages 839–846, 1998. [3](#)
- [22] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *Proc. IEEE Int. Conf. Computer Vision Workshops*, pages 1630–1637, 2009. [2](#), [3](#)
- [23] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. Technical report, CSAIL, MIT, 2012. [1](#)
- [24] C. Yang and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992. [1](#)
- [25] C. Zach. Fast and high quality fusion of depth maps. In *Proc. Int. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, volume 1, 2008. [1](#)
- [26] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Graph. Models*, 75(3):126–136, 2013. [1](#)
- [27] M. Zwicker, H. Pfister., J. V. Baar, and M. Gross. Surface splatting. In *Computer Graphics (Proc. SIGGRAPH)*, pages 371–378, 2001. [4](#)

**User Interface for Volume Rendering
in Virtual Reality Environments**

**Jonathan Klein, Dennis Reuling, Jan Grimm, Andreas Pfau,
Damien Lefloch, Martin Lambers, Andreas Kolb**

Version: February 8, 2013

Faculty of Science and Technology
Institute for Vision and Graphics (IVG)
Computer Graphics and
Multimedia Systems Group
Prof. Dr. Andreas Kolb

Abstract

Volume Rendering applications require sophisticated user interaction for the definition and refinement of transfer functions. Traditional 2D desktop user interface elements have been developed to solve this task, but such concepts do not map well to the interaction devices available in Virtual Reality environments.

In this paper, we propose an intuitive user interface for Volume Rendering specifically designed for Virtual Reality environments. The proposed interface allows transfer function design and refinement based on intuitive two-handed operation of Wand-like controllers. Additional interaction modes such as navigation and clip plane manipulation are supported as well.

The system is implemented using the Sony PlayStation Move controller system. This choice is based on controller device capabilities as well as application and environment constraints.

Initial results document the potential of our approach.

1. Introduction

Volume Rendering visualizes 3D grids of voxels. Each voxel typically stores a scalar value representing density, as retrieved via a 3D scanning technique such as CT or MRI. Direct Volume Rendering techniques such as volume ray casting work directly on the voxel data instead of extracted geometry such as isosurfaces. Such techniques use a *transfer function* to map voxel values to opacity and color. The volume ray caster then generates a ray through the 3D grid for every pixel in the image plane, samples the voxel data along the ray, and composites the opacity and color information given by the transfer function to compute the final pixel color.

A basic transfer function is a one-dimensional function that directly maps a scalar voxel value to opacity and color. Volume Rendering applications require user interface concepts that allow efficient and precise design and refinement of such transfer functions, to enable the user to visualize the interesting parts of the volume data set. In the traditional 2D graphical user interface domain of desktop systems, this problem is solved using 2D widgets that typically allow mouse-based manipulation of the functions [3]. This paper focuses on one-dimensional transfer functions, but note that advanced two-dimensional transfer functions models exist that take the gradient or the curvature at the voxel location into account and require even more complex user interfaces.

Since Virtual Environments are especially well suited to explore spatial properties of complex 3D data, bringing Volume Rendering applications into such environments is a natural step. However, defining new user interfaces suitable both for the Virtual Environment and for the Volume Rendering application is difficult. Previous approaches mainly focused on porting traditional 2D point-and-click concepts to the Virtual Environment [8, 5, 9]. This tends to be unintuitive, to complicate the interaction, and to make only limited use of available interaction devices.

2. Related Work

In this paper, we propose an intuitive 3D user interface for Volume Rendering based on interaction devices that are suitable for Virtual Reality environments. We focus on a simplified approach to design and refine transfer functions that allows intuitive use of interaction devices, specifically the Sony PlayStation Move controller system. Our demonstration system also supports other Volume Rendering interaction modes such as navigation and clip plane manipulation.

The remainder of this paper is organized as follows. Sec. 2 discusses related work. In Sec. 3, we describe our user interface concepts in detail, and present its implementation based on Sony PlayStation Move controllers in a Virtual Reality Lab. Initial results are shown in Sec. 4. Sec. 5 concludes this paper.

2. Related Work

One of the first applications of Volume Rendering in a Virtual Reality environment was presented by Brady et al. in 1995 [1]. This early work concentrated on navigation using a Wand-like device. In 2000, Wohlfahrter et al. presented a two-handed interaction system with support for navigating, dragging, scaling, and cutting volume data in a Virtual Reality environment [12]. Neither of these early approaches supported transfer function manipulation.

One of the first works on transfer function editing for Volume Rendering in Virtual Reality environments was presented by Schulze-Döbold et al. in 2001 [8]. Their transfer function editor requires a 6 DOF controller with three buttons. The controller is primarily used to point at an interaction element to select it for manipulation. To control scalar values, the editor uses virtual knobs that are manipulated by twisting the hand. The three buttons are used to manipulate position and size of the 2D transfer function editor inside the 3D environment. This interface is directly based on the 2D desktop point-and-click interface. Consequently, the authors refer to the controller as a 3D mouse. Schulze-Döbold later refined the user interface based on feedback collected in a user study [7], but the principal design remained unchanged.

Wössner et al. reuse Schulze-Döbold's work for the purpose of collaborative volume exploration in distributed setups [13]. Kniss et al. split the task of defining multidimensional transfer functions into a classification step and an exploration step [5]. The classification step, which defines the transfer function, is performed prior to visualization on a classical 2D desktop system using the mouse. The Virtual Reality interface is based on Schulze-Döbold's work.

Later works also mainly use variations of this approach of bringing 2D point-and-click interfaces to 3D environments [4, 9]. An exception is the work of Tawara and Ono from 2010, in which they combined a Wiimote and a motion tracking cube to get a tracked manipulation device for a volume data application [11]. However, their approach focuses on volume segmentation in augmented reality; in particular, it does not support transfer function manipulation.

3. 3D User Interface for Volume Rendering

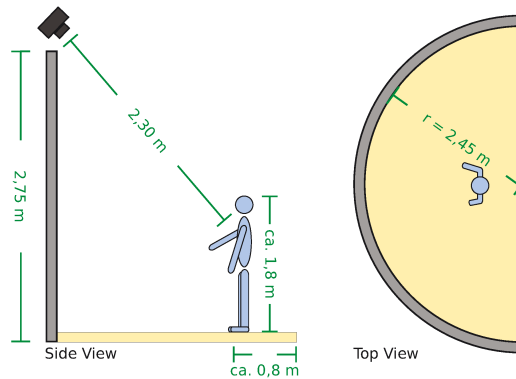


Figure 1: Overview of the Virtual Reality environment

3. 3D User Interface for Volume Rendering

A user interface for Volume Rendering applications must support two key interaction modes:

- *Navigation.* This allows to inspect the volume from various perspectives by applying translations and rotations. A Virtual Reality environment with user tracking additionally allows the user to move around the volume.
- *Transfer function manipulation.* A transfer function allows to visualize the interesting parts of the volume (by assigning color and opacity to the interesting voxel value ranges) and at the same time remove other parts of the volume from view (by mapping the corresponding voxel values to zero opacity).

In addition to these modes, Volume Rendering applications usually provide supplemental tools such as clip planes.

In this paper, we focus on the transfer function manipulation mode of the user interface.

3.1. Choice of Input Devices

Our choice of input devices, interface concepts, and implementation details was based on the interaction requirements given by the Volume Rendering application, with special emphasis on transfer function manipulation, and on the specific constraints given by the Virtual Reality environment.

The Virtual Reality environment which we used in this project has an open cylindrical screen and a floor screen, providing a wide field of view. See Fig. 1 and 2. The cylindrical screen has four rear-projection stereo channels and the floor screen has two front-projection stereo channels. Additionally, the environment provides optical user tracking.

3. 3D User Interface for Volume Rendering



Figure 2: *Interactive Volume Rendering in the Virtual Reality environment*

We considered interaction devices that are based on readily available and affordable hardware components and provide enough input facilities for both navigation and transfer function editing. Devices that fulfill these criteria include the Microsoft Kinect, the Nintendo Wiimote, and the Sony PlayStation Move. Traditional game controllers as well as the Flystick provided by our optical tracking system were excluded since they do not provide enough input facilities.

Microsoft Kinect uses an optical tracking system to follow the movement of the user, allowing full-body gesture interaction. Unfortunately the Kinect system requires the camera to be placed directly in front of the user, which was not possible in our environment. We experimented with a Kinect camera placed at the top of the screen and looking at the user with an angle of approximately 45 degrees, but this setup leads to unusable tracking data.

The concepts of the Wiimote and the Move are similar. In both systems the user holds a wand-like controller in his hand that can measure its movement and orientation. The Move system additionally uses an optical tracking system to determine the absolute position of the controller, and can therefore provide position and orientation data with higher precision and reliability. In contrast to the Kinect system, the Move system works fine with the camera mounted on top of the screen, as shown in Fig. 1. Furthermore, the Move controller has a glowing bulb on top whose color can be changed. This gives interesting opportunities for user feedback (see Sec. 3.2.1).

Recently, Takala et al. identified a lack of user interface concepts based on the PlayStation Move system, partly due to the locked-up nature of the SDK [10]. That situation has changed: the SDK is now freely available for educational and research purposes. Additionally, prices for the hardware part of system dropped significantly.

For these reasons, we chose to base our experiments on the PlayStation Move system.

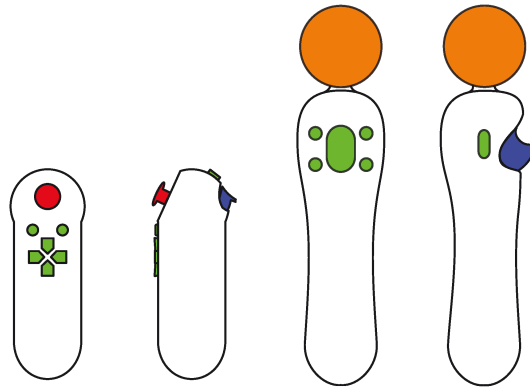


Figure 3: The Sony PlayStation Move Nav-Pad (left) and controller (right). Both devices provide digital buttons (green) and an analogue trigger (blue). The Nav-Pad provides an additional analogue stick (red), while the controller has a bulb (orange) on its top that can glow in different colors.

3.2. Interface Concept

To allow both navigation and transfer function manipulation and to have enough input facilities for the various functionalities required by these two modes, we decided to use both the Move controller (for the right hand) and an additional Move Nav-Pad (for the left hand). See Fig. 3. The Move controller, whose bulb is tracked by the camera of the system, is used for manipulation (translation and rotation in navigation mode, and modification in transfer function editing mode), while the Move Nav-Pad is used for selection and switching purposes. This configuration is classified by Schultheis et al. [6] as an asymmetric two-handed interface using two Wand-like input devices.

3.2.1. Transfer Function Editing

Small changes to a transfer function can result in significant changes in the visualization result, and usually more than one range of voxel values represents interesting parts of the volume data. For this reason, typical Volume Rendering applications allow to specify the transfer function using piecewise linear building blocks. However, this requires a very fine-grained control, typically using a Mouse-based interface.

In order to allow more intuitive manipulation of transfer functions using hand-held devices that typically favor coarser control movements, we use a different transfer function model. In our model, a transfer function is defined as the sum of window functions, called *peaks*. Each peak highlights a small voxel value range in a specific color.

3. 3D User Interface for Volume Rendering

A peak p is defined by its center c , width w , and height h :

$$p(x) = \begin{cases} h \cdot \sin\left(\frac{\pi}{2w}(x - c + w)\right) & c - w \leq x \leq c + w \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

(Alternatively, different window functions could be used).

The result of a transfer function t for a voxel value x is then defined as the result of alpha-blending the peak colors using the peak value $p(x)$ as the opacity value.

In practical use, only a small number $n < 8$ of peaks is required, since more peaks tend to clutter up the visualization result. An example is given in Fig. 4.

This transfer function model significantly reduces the number of parameters that a user has to modify, while still allowing flexible and powerful transfer functions.

The user can add a peak to a transfer function and select its color from a list of predefined colors using digital buttons on the Move Nav-Pad (see Fig. 3). Similarly, peaks can be deleted, temporarily disabled, or selected for parameter adjustment using additional Nav-Pad buttons.

To change the center c , width w , and height h of a peak, the Move controller is used. We tried different combinations of mapping these three peak parameters to the x -, y -, and z -axes of the Move controller. Using the z -axis proved to be unintuitive and therefore difficult to control. Our current solution is that the user has to choose (using buttons on the Move controller) to adjust either c and h or w . This has the advantage that both adjustments take place in the x/y -plane. The reason for separating w from the other parameters was that the visualization result is especially sensitive to changes of peak widths, so that users tend to first define position and height of a peak and then fine-tune the result by adjusting its width.

To provide the user with visual feedback about the current transfer function properties and the selection state, we display an overview widget at a fixed position in the Virtual Environment, as shown in Fig. 4. Note that this widget is for informational purposes only and does not require traditional point-and-click functionality.

As an additional aid, we set the color of the glowing Move controller bulb to the color of the transfer function that is currently selected. Experienced users can use this feedback to determine the current state of transfer function manipulation without looking at the overview widget.

3.2.2. Navigation

Navigation is implemented by performing translation and rotation using the tracked Move controller. Translation is active while the largest digital button of the Move controller is pressed, while rotation is active while the analogue trigger of the Move controller is pressed. See Fig. 3. Translation works by directly applying Move controller position changes to the volume. Rotation works by mapping horizontal controller movements to volume rotations around the y -axis, vertical movements to volume rotations around the x -axis, and controller rotations around the z -axis to volume rotations around the z -axis.

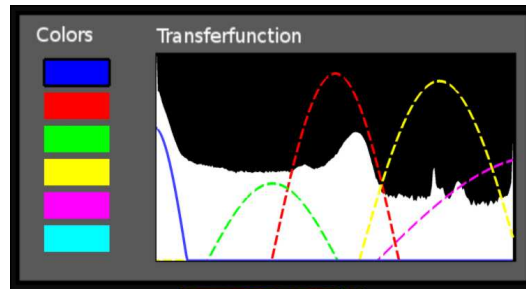


Figure 4: Transfer function defined by $n = 5$ peaks. Each peak assigns color and opacity information to a voxel value range. The histogram of voxel values is displayed in white in the background.

An obvious alternative would be to directly apply both Move controller position and orientation changes to the volume while in navigation mode, but separating translation and orientation in the way described above allows a more fine-grained control of movement, which is useful to examine smaller volume areas in detail. Furthermore, mapping controller movements to rotations instead of directly using the controller orientation avoids uncomfortable wrist positions. Requiring a button to be pressed for navigation mode allows the user to move freely in the Virtual Reality environment without unintentionally moving the volume.

3.3. Implementation

The physical setup is described by Fig. 1.

Our software implementation is based on the Equalizer framework for parallel and distributed rendering [2]. This allows the application to run across the six nodes of our render cluster, each of which renders one of the stereo channels using two graphics cards.

We used the official Move.Me SDK from Sony for connecting the PlayStation Move system to Equalizer. The controller sends its sensor data via Bluetooth to the PlayStation console, on which the Move.Me SDK runs as a server application. Our application acts as a client to this server and receives position, orientation, and button state data via network UDP packets. This data is transformed to custom input events and handed over to the Equalizer event handling mechanisms to allow consistent input event handling.

The GPU-based volume ray caster is based on a ray casting shader implementation provided by the Visualization Library project¹. The ray caster is straightforward but proved sufficient for our purposes while being fast enough for interactive use in a Virtual Reality environment.

¹<http://www.visualizationlibrary.org>

4. Initial Results

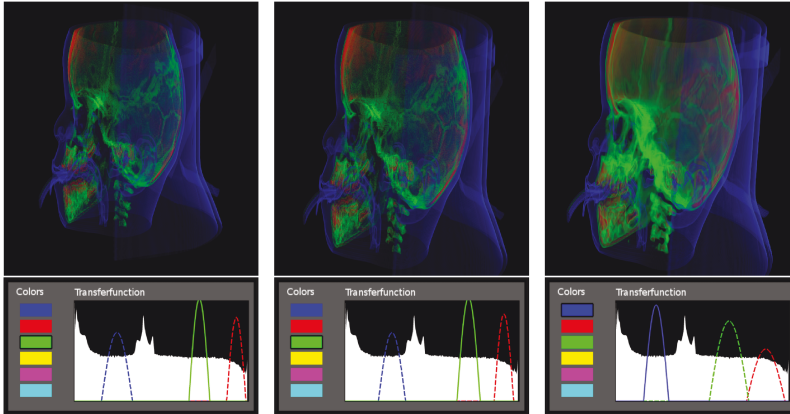


Figure 5: Predefined result

Figure 6: Result of the experienced user

Figure 7: Result of the inexperienced user

4. Initial Results

In the figures throughout this paper, we used the “Baby Head” data set available from the volume library of Stefan Roettger².

As a first test of our concept, we performed an experiment involving one test user with experience in both Virtual Reality and Volume Rendering applications and another test user with no experiences in these domains.

The task for these two test users was to reproduce a visualization result for the “Baby Head” data set, starting from a default transfer function. The predefined visualization result was produced with a transfer function consisting of three peaks that separate the skeleton (green), the teeth (red), and the skin (blue). See Fig. 5.

The experienced test user was able to solve the task in less than half a minute with good results, shown in Fig. 6. After an introduction to controller usage and button configuration, the inexperienced user was able to achieve a satisfying result, shown in Fig. 7, in approximately one minute.

In this and similar experiments with several test data sets (only one of which is shown here), the simplified transfer function model was powerful enough and the transfer function manipulation was precise enough to highlight interesting aspects the data. More demanding data sets might require more fine-grained control, which may require changes to the transfer function model and/or adjustments to the interface sensitivity.

The two-handed interface requires some coordination between both hands which inexperienced users are unaccustomed to. However, after some practice, this does not seem

²<http://schorsch.efi.fh-nuernberg.de/data/volume/>

5. Conclusion

pose a problem.

Another challenge especially for inexperienced users is to remember the mapping between controller movements and buttons to the various interaction functionalities of the application. We plan to integrate an online help function that displays images of the controllers on screen, along with schematic descriptions of their usage, in order to avoid the need for leaving the Virtual Reality environment to look up user interface documentation.

Despite these shortcomings, we believe the approach has the potential to be usable in real-world Volume Rendering applications.

5. Conclusion

We propose a 3D user interface concept for Volume Rendering in Virtual Environments. Unlike previous approaches, this concept does not try to map traditional 2D point-and-click concepts to the Virtual Environment; instead, it is based on a set of intuitive user actions using the Sony PlayStation Move controller system.

For this purpose, a simplified transfer function model was designed that allows a reduction of interaction complexity. This comes at the cost of reduced flexibility and precision when compared to a traditional 2D desktop interface, but we believe that our system is still flexible and precise enough for exploration of most volume data sets, while allowing faster and more intuitive manipulation of transfer functions.

In the future, we would like to refine the interface based on user feedback. Furthermore, it would be interesting to explore the possibility to extend our approach to two-dimensional transfer functions.

References

1. R. Brady, J. Pixton, G. Baxter, P. Moran, C. Potter, B. Carragher, and A. Belmont. Crumbs: a virtual environment tracking tool for biological imaging. In *Proc. Biomedical Vis.*, pages 18–25, 82, Oct. 1995.
2. S. Eilemann. Equalizer: A scalable parallel rendering framework. *IEEE Trans. Visualization and Computer Graphics*, 15(3):436–452, May 2009.
3. K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. AK Peters, 2006.
4. C. He, A. Lewis, and J. Jo. A novel human computer interaction paradigm for volume visualization in projection-based virtual environments. In *Proc. Int. Symp. Smart Graphics*, pages 49–60, 2007.
5. J. Kniss, J. Schulze, U. Wössner, P. Winkler, U. Lang, and C. Hansen. Medical applications of multi-field volume rendering and VR techniques. In *Proc. IEEE VGTC Symp. Vis.*, pages 249–254, 2004.

5. Conclusion

6. U. Schultheis, J. Jerald, F. Toledo, A. Yoganandan, and P. Mlyniec. Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks. In *IEEE Symp. 3D User Interfaces (3DUI)*, pages 117–124, Mar. 2012.
7. J. Schulze-Döbold. *Interactive Volume Rendering in Virtual Environments*. PhD thesis, University of Stuttgart, Aug. 2003.
8. J. Schulze-Döbold, U. Wössner, S. Walz, and U. Lang. Volume rendering in a virtual environment. In *Proc. Immersive Projection Technology Workshop and Eurographics Workshop on Virtual Environments*, pages 187–198, 2001.
9. R. Shen, P. Boulanger, and M. Noga. MedVis: A real-time immersive visualization environment for the exploration of medical volumetric data. In *Proc. Biomedical Vis.*, pages 63–68, July 2008.
10. T. Takala, P. Rauhamaa, and T. Takala. Survey of 3DUI applications and development challenges. In *IEEE Symp. 3D User Interfaces (3DUI)*, pages 89–96, Mar. 2012.
11. T. Tawara and K. Ono. A framework for volume segmentation and visualization using augmented reality. In *IEEE Symp. 3D User Interfaces (3DUI)*, pages 121–122, Mar. 2010.
12. W. Wohlfahrter, L. Encarnação, and D. Schmalstieg. Interactive volume exploration on the studydesk. In *Proc. Int. Projection Technology Workshop*, June 2000.
13. U. Wössner, J. P. Schulze, S. P. Walz, and U. Lang. Evaluation of a collaborative volume rendering application in a distributed virtual environment. In *Proc. Eurographics Workshop on Virtual Environments (EGVE)*, page 113ff, May 2002.

Ellipsoidal Cube Maps for Accurate Rendering of Planetary-Scale Terrain Data

M. Lambers and A. Kolb

Computer Graphics Group
University of Siegen

Abstract

Advances in sensor technology lead to a rapidly growing number of terrain data sets with very high spatial resolution. To allow reliable visual analysis of this data, terrain data for planetary objects needs to be rendered with accurate reproduction of every detail.

This combination of very large scale and very fine detail is challenging for multiple reasons: the numerical accuracy of typical data types is not sufficient, simple spherical planet models fail to accurately represent the data, and distortions in map projections used for data storage lead to sampling problems.

In this paper, we propose the Ellipsoidal Cube Map model to address these problems. We demonstrate the possibilities of the model using a simple renderer implementation that achieves interactive frame rates for a variety of data sets for Earth, Moon, and Mars.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.8 [Computer Graphics]: Applications—

1. Introduction

Today, spaceborne and airborne Remote Sensing systems provide data sets for the Earth's surface with details on the decimeter or even centimeter scale. This is true both for image-like data, e.g. from multispectral sensors or Synthetic Aperture Radar (SAR) systems, and for elevation data, e.g. from Interferometric SAR (InSAR) or Light Detection And Ranging (LIDAR) systems. Additionally, high-resolution data sets increasingly become available for other objects in the solar system, such as the Moon and Mars.

This raises the problem of rendering high-resolution terrain data sets for planetary-scale objects with adequate precision to allow reliable visual analysis. Compared to terrain rendering systems that are only concerned with limited local areas, additional challenges in the areas of numerical accuracy, planet modelling, and terrain data map projection must be addressed.

In this paper, we propose the *Ellipsoidal Cube Map* (ECM) model. The main contributions are as follows:

- A data model that uses an ellipsoid as the common ref-

erence, to allow accurate interpretation of high-resolution data.

- A terrain data map projection that provides nearly uniform data sampling quality across the complete planet surface, without singularities at poles or decreasing quality in border regions.
- Methods to split geometry computations for rendering into two parts: static double-precision computations and dynamic single-precision computations. This allows accurate rendering using efficient single-precision rendering pipelines.

In contrast to previous approaches, the proposed techniques allow accurate rendering of planet-wide data sets at decimeter and centimeter resolutions across the complete planet surface. Furthermore, the model allows to generate or augment terrain data at rendering time for applications such as erosion simulations, procedural terrain detail generation, or interactive fusion of multiple data sets.

We demonstrate the techniques using a simple level-of-detail and rendering approach. Alternatively, the ECM model can be combined with existing performance-optimized terrain rendering approaches.

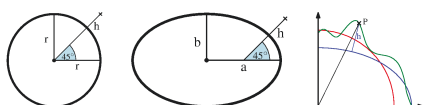


Figure 1: Measurement of elevation h and latitude $\phi = 45^\circ$ in a spherical planet model (left) and in an ellipsoidal planet model (middle). Right: transforming a point P on a planet surface (green) from an ellipsoidal planet model (blue) to a spherical planet model (red) requires exact knowledge of its elevation h ; uncertainty in h results in uncertainty in the coordinates on the sphere surface.

2. Related Work

Numerical Accuracy: Planetary-scale scenes require coordinates with large absolute values, and high-resolution terrain data requires small differences between these values to be preserved. The associated limitations of the single-precision floating point data type commonly used in rendering pipelines are well documented [Tho05, PG07, KLJ*09]. A recent discussion is given by Cozzi and Ring [CR11].

While using double-precision coordinate data for scene management is required, rendering pipelines typically use a single-precision data type for efficiency reasons. One approach to solve this problem is to divide the scene into multiple segments with local origins and render these segments sequentially with appropriate model-view matrices computed on the CPU in double-precision [LKR*97, RLIB99]. An alternative approach with simpler scene management is to shift the origin of the scene to the camera position for rendering [Tho05]. This preserves detail in vicinity to the camera, and limits precision loss to regions far away from the camera where it does not affect the rendering result.

Cignoni et al. use camera-centric base information for each patch of geometry and use linear interpolation in single-precision on the GPU across the patch, which introduces an error that needs to be accounted for [CGG*03]. Kooima et al. use specialized methods to refine spherical geometry in local coordinate systems on the GPU, but their texture coordinate precision is limited to meter scale on an Earth-sized planet [KLJ*09]. Lambers and Kolb use costly double-precision computations for selected intermediate steps on the GPU [LK09], leading to reduced rendering performance.

Planet Model: The sphere is the most widely used planet model [LKR*97, RLIB99, CGG*03, KLJ*09]. However, the true shape of most planetary objects is much better represented by a reference ellipsoid, which is an ellipsoid of rotation given by a semi-major axis a and a semi-minor axis b . The prevalent reference ellipsoid for the Earth is defined by the World Geodetic System 1984 (WGS84) standard [US04]. Remote Sensing data sets are generally given relative to a reference ellipsoid for accuracy reasons.

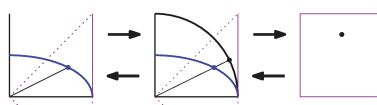


Figure 2: Forward and inverse map projection. For forward projection, a point on the reference ellipsoid (left) is first shifted to the sphere (middle), and then projected onto the cube side (right).

A precise transformation of data sets to a spherical model requires exact knowledge of elevation information (see Fig. 1), but this transformation is complex, and elevation data with sufficient precision is often not available. None of the existing sphere-based approaches are documented to apply such a transformation. Instead, elevation data is usually interpreted along sphere normals. This introduces errors (see Fig. 7). To avoid the impractical transformation and all associated errors entirely, a rendering system must use the reference ellipsoid model. Only few approaches do this [WRH99, LK09].

Map Projection: A popular map projection is the equidistant cylindrical (or plate carrée) projection. Like all projections that map the complete planet surface, the plate carrée projection suffers from singularities, in this case at the poles. The associated sampling problems in the polar regions cause significant storage and data access overhead in the renderer as well as radial blur in the rendered image [KLJ*09].

The only way to avoid these problems is to use multiple maps. Kooima et al. combine plate carrée projection for the equatorial region and two additional projections for the polar regions using weighted averages for smooth transitions [KLJ*09]. Cignoni et al. inscribe a sphere into a cube and then use Gnomonic projection to map the sphere surface to the six cube sides [CGG*03]. See Fig. 3. Gnomonic projection suffers from shape and area distortions that rapidly increase away from the projection center [Sny87]. Lebour et al. proposed an adjustment to Gnomonic projection [LMG10] that can reduce these distortions to some degree (see Sec. 3).

3. Ellipsoidal Cube Map

We propose the Ellipsoidal Cube Map (ECM), consisting of a reference ellipsoid, a circumscribing cube, and a map projection from the ellipsoid surface to the six cube sides.

The reasons for choosing a reference ellipsoid as the base model are given in Sec. 2. Since using a single map projection will always result in singularities [KLJ*09, Sny87], the base model must be subdivided. The circumscribing cube is a uniform subdivision that avoids special handling and blending of different regions, and allows simple and efficient quadtree-based data management and level of detail.

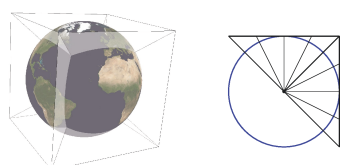


Figure 3: Left: a spherical planet model inscribed into a cube. Right: Gnomonic projection maps the surface onto tangential cube sides.

To be able to use the same map projection for each cube side, the ellipsoidal base model has to be shifted to a spherical model before applying the map projection, as shown in Fig. 2. Note that this shift applies to points directly on the reference surface, i.e. with zero elevation, and can therefore be performed without loss of precision.

With this base model shift, the remaining problem is to find a suitable projection from the sphere surface to the cube sides. There are basically two kinds of distortions to consider minimizing in a projection from the sphere surface to a plane: angle distortions and area distortions [FH05]. Angle distortions lead to shape distortions so that e.g. circular areas on a sphere surface are mapped to ellipsoidal areas on a plane. *Conformal* projections preserve angles. Area distortions cause equally sized areas on a sphere to be mapped to regions of different size on a plane. *Equal-area* projections preserve area ratios.

In a rendering system, both angle and area distortions cause sampling overhead and rendering artifacts [KLJ*09]. Since the sphere surface is not developable, a projection onto a plane cannot be both conformal and equal-area at the same time [Sny87, FH05]. For these reasons, one must choose a projection that limits both angle and area distortions to an acceptable degree.

A map projection that is applicable to the circumscribing cube must map each of the six sphere surface areas to square maps. We further require that each cube side is handled equally, which excludes the HEALPix projection that uses different projections for the polar sides and the equatorial sides of a cube [CR07].

We considered the following projections for use with the ECM model:

- Gnomonic projection (see Fig. 3). This straightforward method is used e.g. by Cignoni et al. [CGG*03]. It is neither conformal nor equal-area, and in fact distortions are known to increase rapidly with growing distance from the projection center [Sny87].
- Adjusted Gnomonic projection [LMG10]. The Gnomonic cube side coordinates $u, v \in [-1, 1]$ are transformed by $f(x) = \frac{4}{\pi} \cdot \text{atan}(x)$ to reduce distortions.

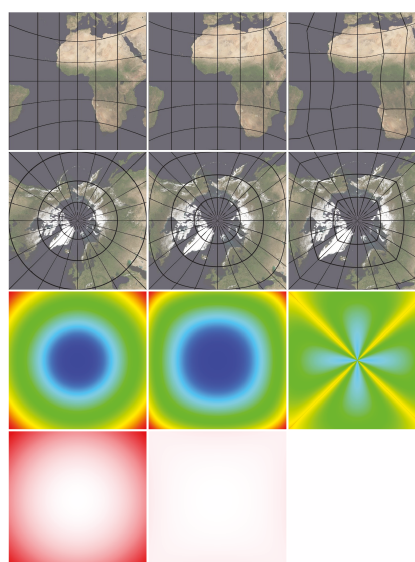


Figure 4: Different projections from the sphere surface to a cube side. From left to right: Gnomonic projection, Adjusted Gnomonic projection, and Quadrilateralized Spherical Cube projection. The latitude and longitude lines in the first and second row have distances of 15° . The third row shows angular distortion color-coded from blue (no distortion) to red (maximum distortion). The fourth row shows area distortion color-coded from white (no distortion) to full red (maximum distortion).

- Quadrilateralized Spherical Cube (QSC) projection. Chan and O'Neill [CO75] and O'Neill and Laubscher [OL76] developed a QSC model in which they inscribed a cube to a sphere and defined hierarchical structures on each cube side for data storage. For that purpose, they designed the QSC projection to be equal-area and at the same time limit angle/shape distortions.

For the visual comparison of the candidates shown in Fig. 4, we computed angle and area distortions across a cube side for each projection candidate based on the standard method of Tissot's Indicatrix as described by Snyder [Sny87]: angle distortion corresponds to Tissot's maximum angular deformation ω , and area distortion corresponds to differences in Tissot's areal scale factor s . The maximum angular distortion is 31.1° for both Gnomonic and Adjusted Gnomonic projection, and 24.9° for QSC projection. Area distortions are clearly strongest in Gnomonic projec-

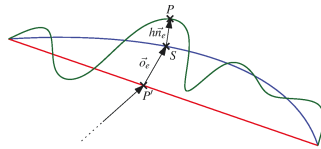


Figure 5: Computation of camera-centric coordinates. P' is interpolated from the quad corners along the red line. The displacement vector $\vec{\sigma}_e$ is the offset between P' and the corresponding position S on the ellipsoid surface (blue). The final position P on the true surface geometry (green) is computed by applying the elevation h along the normal \vec{n}_e .

tion, with a maximum ratio of 5.19. Adjusted Gnomonic projection significantly reduces area distortions to a maximum ratio of 1.41. Per construction, QSC projection does not suffer from area distortions at all.

Based on these results, we chose the QSC projection for use with the ECM model.

Each cube side is the root of a quadtree hierarchy in which terrain data maps are stored. For a sphere with a given radius r , a quadtree level l provides a fixed, uniform resolution across the complete sphere surface, since the QSC projection is equal-area. For an ellipsoid with axes $a = r$ and $b < r$, small variations are introduced by the preceding shift to a sphere, depending on the ellipsoid flattening.

4. Geometry Computations for Rendering

As described in Sec. 2, scene management must use double precision, but for efficiency reasons it is desirable to use only single-precision in the rendering pipeline while still maintaining full accuracy, and that can be achieved using camera-centric coordinates for rendering.

The ECM model allows to split coordinate computations required for rendering one quad into static double-precision computations, performed once on the CPU, and dynamic single-precision computations, performed for each frame in the rendering pipeline.

Static double-precision computations:

- For each quad: cartesian planetocentric coordinates Q_0, \dots, Q_3 of the quad corners on the ellipsoid surface.
- For each quad sample: the displacement vector $\vec{\sigma}_e$ between a position P' bilinearly interpolated from Q_0, \dots, Q_3 and the true ellipsoid surface position S , and the normal vector \vec{n}_e at S . See Fig. 5.

Q_0, \dots, Q_3 are stored in double-precision. To render with camera-centric coordinates, the double-precision camera coordinates C are subtracted, and the resulting camera-centric quad coordinates Q'_0, \dots, Q'_3 are given to the rendering pipeline in single-precision.

The displacement and normal maps are computed in double-precision because they depend on planetary-scale coordinate values. The results, however, are local to the quad reference, and invariant to translation. Therefore, these maps are static and can be stored and used in single-precision. The symmetry of the ellipsoid can be exploited to reduce the number of computations.

Dynamic single-precision computations: To render a given quad, quad-relative vertex coordinates $(s, t) \in [0, 1]^2$ need to be transformed to camera-centric cartesian coordinates $P \in \mathbb{R}^3$. Using the camera-centric quad corner coordinates Q'_0, \dots, Q'_3 , the precomputed displacement and normal maps, and the current elevation map, this can be done in single-precision in the rendering pipeline as follows:

- Compute a position P' by bilinearly interpolating Q'_0, \dots, Q'_3 using (s, t) .
- Read $\vec{\sigma}_e$ from the displacement map, \vec{n}_e from the normal map, and h from the elevation map at position (s, t) .
- Compute the final position $P = P' + \vec{\sigma}_e + h \cdot \vec{n}_e$.

With increasing quadtree levels, an ellipsoid surface patch represented by a quad converges to a plane, and thus displacement vectors converge to zero and normals converge to the plane normal. For this reason, if an application determines that the maximum length of a displacement vector for a given quadtree level is negligible, it can ignore displacement and normal maps starting from that level.

If a single static elevation data set is all that ever needs to be rendered, the static displacement map, normal map, and elevation map can be combined into a single static displacement map during preprocessing. Alternatively, keeping these maps separate allows to switch elevation data sets during rendering, and even to generate or augment elevation data on the fly, e.g. for terrain editing applications [ŠBBK08] or systems that interactively compose multimodal remote sensing data [LK09].

5. Implementation

Like other quadtree-based terrain rendering systems, our demonstration implementation consists of two parts: offline preprocessing and interactive rendering.

During preprocessing, quadtree hierarchies are built from the input elevation and texture data sets. An appropriate number of quadtree levels is chosen based on the nearly uniform resolution of each level. Preprocessing involves remapping the input data. In contrast to other approaches, precomputing geometry information is not necessary.

The interactive rendering step uses a common quadtree-based level of detail approach. In the first stage, the cube side quads that are necessary to render the current view are selected. In the second stage, the necessary data from the preprocessed terrain maps is asynchronously transferred to GPU memory (with caching) and rendered.

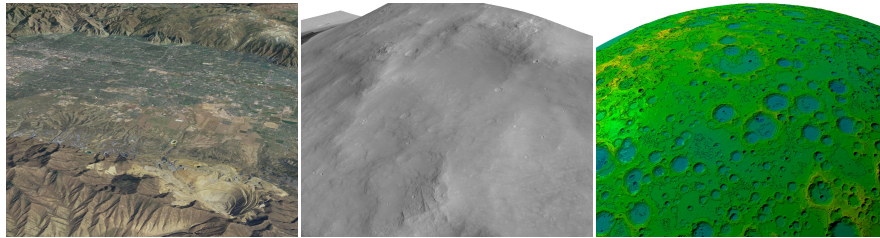


Figure 6: Left: an area in Utah, with different elevation data sets fused at rendering time. Middle: a HiRISE image of Mars, combined with elevation data from the Mars Orbiter Laser Altimeter. Right: elevation data of the Moon, derived from Lunar Reconnaissance Orbiter Camera (LROC) images. The texture data is computed from the elevation data at rendering time.

The first stage works as follows. The set of quads selected for rendering is initialized with the six root quads of the cube sides. The bounding box of each selected quad is projected to screen space, and unless it is outside the visible area or the number of pixels covered is less than $t = 1.5$ times the number of data samples it provides, it is replaced by its four subquads. This process is repeated until no more changes to the set of selected quads are necessary.

In the second stage, our demonstration implementation simply renders each quad with full geometric detail by rendering $2^k \times 2^k$ subquads for quads that provide $2^{k+1} \times 2^{k+1}$ data samples. This brute-force approach still results in interactive frame rates with current graphics hardware; alternatively, implementing a view-dependent level of detail mechanism is also possible. Skirts around each quad avoid discontinuities between quads of different quadtree levels [DSW09, LMG10].

6. Experimental Results

The experimental results described below were obtained on a Linux PC with an Intel Core i7-930 processor and an NVIDIA 480GTX graphics card. Preprocessing used 8 threads where possible. The quad size was 512×512 samples, and the viewport size was 1280×1024 pixels.

The Utah HRO 2006 data set provides photos with ca. 25 cm ground resolution. It consists of 3239 files, each storing 16000×16000 RGB samples using lossy compression. This corresponds to 2316.7 GiB of uncompressed input data. Preprocessing the data set, including lossless compression of the resulting quads, took 114 hours and 54 minutes.

The SRTM 90m Digital Elevation Database version 4 [JRNG08] provides data for Earth's land masses between $+60^\circ$ and -60° latitude. It consists of 872 files, each storing 6000×6000 16-bit samples (58.5 GiB uncompressed data). Preprocessing this data set took 9 hours and 30 minutes, including lossless compression.

© The Eurographics Association 2012.

Fig. 6 shows several example scenes. The left image shows a view of an area in Utah. The elevation data is combined from different data sets at 2 m and 5 m resolution at rendering time, allowing interactive examination and comparison of data sets. 21 quads from levels 6–11 are rendered at approximately 35 frames per second. The middle image shows a HiRISE image of Mars overlaid over elevation data from the Mars Orbiter Laser Altimeter. 21 quads from quadtree levels 10–13 are rendered at approximately 50 frames per second. The right image shows elevation data for the Moon derived from Lunar Reconnaissance Orbiter Camera images. Per-pixel lighting is applied. The color gradient texture, including isolines, is computed dynamically from the elevation data during rendering, allowing interactive examination of the elevation data. 25 quads from quadtree levels 2–3 are rendered at approximately 30 fps.

7. Discussion

The main characteristics of the ECM model (ellipsoidal model, QSC projection, accurate rendering in single-precision) affect both quality and efficiency.

The ellipsoidal model avoids errors caused by interpreting elevation data relative to a sphere. See Fig. 7. Approaches that use a sphere model should first transform data sets based on the best elevation model available, thereby reducing this error significantly for planetary objects with small flattening and high-quality elevation models such as Earth. However, this transformation is impractical and is not documented to be applied by any sphere-based approach. Our ellipsoid-based model avoids this transformation and associated errors entirely.

QSC projection avoids data access overhead and sampling problems caused by map projection distortions as described by Kooima et al. [KLJ*09]. It is more expensive than Gnomonic projection, but this additional cost only affects the preprocessing step. During rendering, only the computation of quad corner points Q_0, \dots, Q_3 involves QSC projec-

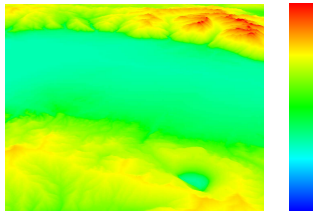


Figure 7: The Utah scene (see Fig. 6 left) with color-coding of the error caused by interpreting elevations along a sphere normal instead of an ellipsoid normal. The used color range (right) encodes the error from 0 m (blue) to 11 m (red).

tion. With typically 20–30 quads required for a view, these runtime costs are negligible.

The rendering approach used with the ECM model maintains full accuracy in single-precision and replaces costly and/or inaccurate geometry refinements, as used by previous approaches, with lookups of static precomputed data.

8. Conclusion

We propose the Ellipsoidal Cube Map model to overcome accuracy limitations of existing techniques to render planetary-scale terrain data sets. The ECM model is based on a reference ellipsoid circumscribed by a cube, and uses a map projection that preserves areas and angles better than alternatives. Accurate rendering can be achieved using only single-precision computations in the rendering pipeline. Generation or augmentation of elevation data at rendering time is possible.

The simple renderer described in Sec. 6 can demonstrate interactive frame rates at full detail level; alternatively, the ECM model can be combined with more sophisticated, performance-optimized terrain rendering methods.

Acknowledgements

We thank the providers of data used in this work: NASA Earth Observatory, the International Centre for Tropical Agriculture (CIAT), the Utah GIS portal, the Mars Orbiter Laser Altimeter (MOLA) and HiRISE missions, and the Lunar Reconnaissance Orbiter mission.

References

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Planet-sized batched dynamic adaptive meshes (P-BDAM). In *Proc. IEEE Visualization* (2003), pp. 147–154. 2, 3

[CO75] CHAN F., O’NEILL E.: *Feasibility Study of a Quadrilateralized Spherical Cube Earth Data Base*. Tech. Rep. EPRF 2-75 (CSC), Environmental Prediction Research Facility, Apr. 1975. 3

[CR07] CALABRETTA M. R., ROUKEMA B. F.: Mapping on the HEALPix grid. *Monthly Notices of the Royal Astronomical Society* 381, 2 (2007), 865–872. 3

[CR11] COZZI P., RING K.: *3D Engine Design for Virtual Globes*. CRC Press, June 2011. 2

[DSW09] DICK C., SCHNEIDER J., WESTERMANN R.: Efficient geometry compression for GPU-based decoding in realtime terrain rendering. *Computer Graphics Forum* 28, 1 (2009), 67–83. 5

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Dodgson N. A., Floater M. S., Sabin M. A., (Eds.), Mathematics and Visualization. Springer, 2005, pp. 157–186. 3

[JRN08] JARVIS A., REUTER H., NELSON A., GUEVARA E.: Hole-filled seamless SRTM data v4, 2008. International Centre for Tropical Agriculture (CIAT). 5

[KLJ*09] KOOIMA R., LEIGH J., JOHNSON A., ROBERTS D., SUBBARAO M., DEFANTI T. A.: Planetary-scale terrain composition. *IEEE Trans. Visualization and Computer Graphics* 15, 5 (2009), 719–733. 2, 3, 5

[LK09] LAMBERS M., KOLB A.: GPU-based framework for distributed interactive 3D visualization of multimodal remote sensing data. In *Proc. IEEE Int. Geoscience and Remote Sensing Symposium* (2009), vol. 4, pp. IV–57–IV–60. 2, 4

[LKR*97] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L., DEN BOSCH A., FAUST N.: *An Integrated Global GIS and Visual Simulation System*. Tech. Rep. GIT-GVU-97-07, Georgia Institute of Technology, Mar. 1997. 2

[LMG10] LERBOUR R., MARVIE J.-E., GAUTRON P.: Adaptive real-time rendering of planetary terrains. In *Full Paper Proc. Int. Conf. Computer Graphics, Visualization and Computer Vision (WSCG)* (Feb. 2010). 2, 3, 5

[OL76] O’NEILL E., LAUBSCHER R.: *Extended Studies of a Quadrilateralized Spherical Cube Earth Data Base*. Tech. Rep. NEPRF 3-76 (CSC), Naval Environmental Prediction Research Facility, May 1976. 3

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *Vis. Comput.* 23, 8 (2007), 583–605. 2

[RLIB99] REDDY M., LECLERC Y., IVERSON L., BLETTER N.: TerraVision II: Visualizing massive terrain databases in VRML. *IEEE Trans. Computer Graphics and Applications* 19, 2 (Mar. 1999), 30–38. 2

[ŠBBK08] ŠT’AVA O., BENEŠ B., BRISBIN M., KRIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. In *Proc. Eurographics Symposium on Computer Animation* (2008), pp. 201–210. 4

[Sny87] SNYDER J.: *Map projections—a working manual*, vol. 1395 of *Professional Paper*. US Geological Survey, 1987. 2, 3

[Tho05] THORNE C.: Using a floating origin to improve fidelity and performance of large, distributed virtual worlds. In *Proc. Int. Conf. on Cyberworlds* (2005), pp. 263–270. 2

[US 04] US NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY: World Geodetic System 1984 (WGS84). <http://earth-info.nga.mil/GandG/wgs84/>, 2004. Accessed 2012-04-30. 2

[WRH99] WARTELL Z., RIBARSKY W., HODGES L.: *Efficient Ray Intersection for Visualization and Navigation of Global Terrain using Spheroidal Height-Augmented Quadrees*. Tech. Rep. GIT-GVU-99-20, Georgia Institute of Technology, 1999. 2



Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Structural performance evaluation of curvilinear structure detection algorithms with application to retinal vessel segmentation

Xiaoyi Jiang^{a,*}, Martin Lambers^b, Horst Bunke^c

^a Department of Mathematics and Computer Science, University of Münster, Germany

^b Computer Graphics Group, University of Siegen, Germany

^c Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland

ARTICLE INFO

Article history:
Available online 23 May 2012

Keywords:
Performance evaluation
Curvilinear structure
Vessel network
Airway tree
Graph matching

ABSTRACT

Curvilinear structures are useful features in a variety of applications, particularly in medical image analysis. Compared to other commonly used features such as edges and regions, there is relatively few work on performance evaluation methodologies for curvilinear structure detection algorithms. For instance, a pixel-wise comparison with ground truth has been used in all recent publications on vessel detection in retinal images. In this paper we propose a novel structure-based methodology for evaluating the performance of 2D and 3D curvilinear structure detection algorithms. We consider the two aspects of performance, namely detection rate and detection accuracy, separately, in contrast to their mixed handling in earlier approaches that typically produces biased impression of detection quality. By doing so, the proposed performance measures give us a more informative and precise performance characterization. Experiments on both synthetic and real examples will be given to demonstrate the advantages of our approach.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Performance evaluation is an important issue in pattern recognition (Bharkad and Kokare, 2011; Cardoso and Sousa, 2011). Extensive early work exists for evaluating algorithms for edge detection and region-based segmentation. Already 1997, for instance, Heath et al. (1997) listed 12 edge detection evaluation methods. A discussion of related literature can be found in (Jiang, 2005; Jiang et al., 2006). In contrast there is only very little work on evaluating algorithms for curvilinear structure detection.

The term *curvilinear structure* denotes a line or a curve with some *width*. In contrast to edges, they have the same shape but non-negligible and varying width. Curvilinear structures are useful features for a variety of applications (finding roads or rivers in aerial images, detecting lanes for traffic tasks, etc.). Particularly in medical imaging they belong to the most widely observed and important features; examples are blood vessels, bones, airway trees, and other thin structures.

It is the purpose of this work to discuss the weaknesses of an approach widely used in medical image analysis literature and to propose an improved evaluation methodology. Throughout this paper our discussion will be exemplified by the task of detecting blood vessels in retinal images. However, it is important to point out that our approach is in no way bounded to retinal images only,

but instead applicable in the general context of the evaluation of 2D and 3D curvilinear structure detection algorithms.

Reliable segmentation of the vasculature in retinal images is a nontrivial task for image analysis and has immense clinical relevance. Blood vessel appearance is an important indicator for diagnoses including diabetes, hypertension, and arteriosclerosis. For this purpose one needs a quantification of features of veins and arteries such as color, diameter, tortuosity, and opacity. Vessel detection provides the fundament for this kind of diagnosis making and indirectly also for other problems. Automatic detection algorithms for pathologies like microaneurysms may be improved if regions containing vasculature is excluded from the analysis (Frame et al., 1998). In addition, knowledge about the location of vessels can aid in registration of retinal images (Zana and Klein, 1999) and detection of other features like optic disc and fovea (Hoover and Goldbaum, 2003). The retinal vessels can be further separated into arteries and veins (Rothaus et al., 2009), which are fundamental to computing the important AV-ratio (ratio of artery to vein calibres). For all these reasons reliable vessel segmentation in retinal images and evaluation of vessel segmentation algorithms is of interest both from a theoretical and a clinical point of view. Similar conclusion applies to other curvilinear structure detection tasks in medical image analysis as well.

We motivate our work with a detailed discussion of the drawbacks of early approaches in Section 2. Then, we describe an improved evaluation methodology in Section 3. Experimental work demonstrating the usefulness of our approach follows in Section 4.

* Corresponding author. Tel.: +49 251 8333759; fax: +49 251 8333755.
E-mail address: xjiang@math.uni-muenster.de (X. Jiang).

Finally, some discussions conclude the paper. This paper is an extended version of the conference paper (Jiang et al., 2011) and contains more detailed literature review, additional technical details, and substantially extended experimental work.

2. Drawbacks of non-structural approaches

The efforts of performance evaluation in computer vision can generally be classified into four distinct categories: theory-based, human evaluation, ground truth (GT) based, and task-based. Our methodology falls into GT-based evaluation. The term ground truth is used to denote some reference result that represents the expected ideal segmentation. The basic idea of GT-based evaluation is then to compute some measure of differences between machine segmentation result and the ground truth.

Many algorithms have been proposed for vessel segmentation in retinal images (Chaudhuri et al., 1989; Fang et al., 2005; Hoover et al., 2000; Jiang and Mojon, 2003; Lam and Yan, 2008; Lam et al., 2010; Martinez-Perez et al., 1999; Staal et al., 2004; Zana and Klein, 2001). While visual inspection has been applied in early approaches for performance evaluation, recent works report on experimental results based on large datasets with manually specified ground truth. Hoover et al. (2000) have collected a dataset STARE (Structured Analysis of the RETina) of 20 retinal images which were manually segmented by two observers. The DRIVE (Digital Retinal Images for Vessel Extraction) dataset (Niemeijer et al., 2004; Staal et al., 2004) consists of 40 images with manual segmentation from three observers. Also the authors of Fang et al. (2005) reported on a dataset of 35 retinal images with ground truth. Two of the datasets (STARE and DRIVE) are publicly available.

In all those works using manually specified ground truth, a straightforward method is used for performance evaluation. Given a machine-segmented result image (MS) and its corresponding hand-labeled ground truth image (GT), any pixel which is marked as vessel in both MS and GT is counted as a true positive. Any pixel which is marked as vessel in MS but not in GT is counted as a false positive. The true positive rate (TPR) is established by dividing the number of true positives by the total number of vessel pixels in GT. The false positive rate (FPR) is computed by dividing the number of false positives by the total number of non-vessel pixels in GT. As an alternative, the FPR can also be based on the total number of non-vessels pixels within the circular field of view¹ (FOV) only (Niemeijer et al., 2004). This latter version seems to be more reasonable and thus will be consistently used in this work. If different pairs of sensitivity and specificity can be achieved, for instance by thresholding a soft classification or various parameter sets, the performance of a system can be investigated by receiver operating curves (ROC). The closer a ROC approaches the top left corner (TPR = 100%, FPR = 0%), the better the performance of the system. In (Lam and Yan, 2008) a variant of TPR is introduced to emphasize on pathological regions that are especially important but difficult to deal with.

Similar pixel-wise comparison has also been used for evaluating binarization methods (Lee et al., 1990) and building extraction from aerial imagery (Shufelt, 1999). While this approach is suitable there for comparing large regions, its application to curvilinear structures as elongated and thin regions is more questionable. This is illustrated in Fig. 1 with several modified versions of the GT. MS_{thin} results from thinning the GT at some places while in MS_{del} some vessel sections are deleted and others remain unchanged. For both MS_{thin} and MS_{del} we obtain TPR = 85.1% and FPR = 0.0%, indicating an equal rate of 85.1% correct detection and no spurious

vessels. But in reality there are substantial differences between the two MS images. In MS_{thin} the entire vessel network is correctly detected, but some vessels have a smaller width than GT. In contrast MS_{del} perfectly equals GT except the deleted parts. A more objective performance measure would be $TPR(MS_{\text{thin}}) = 1.00$ and $TPR(MS_{\text{del}}) < 1.00$, indicating the percentage of the correctly detected part of the vessel network. The correctly detected parts of the vessel network can be further evaluated with respect to the detection accuracy, i.e., the width error. Then, we would expect a non-zero width error for MS_{thin} and zero width error for MS_{del} , respectively.

A second situation in Fig. 1(e) and (f) illustrates a related problem. Again, the two MS images MS_{exp} and MS_{ins} have equal performance measures TPR = 100% and FPR = 1.7%, implying a full detection of the vessel network and 1.7% spurious vessels in both cases. Here MS_{exp} emerges from GT by locally expanding GT while MS_{ins} equals GT plus eight spurious (diagonal) vessel parts. Different from MS_{ins} , the spurious vessel pixels in MS_{exp} cause vessel width errors, but do not change the vessel network structure in any way. Intuitively, a measure $FPR(MS_{\text{exp}}) = 0$ and $FPR(MS_{\text{ins}}) > 0$ thus makes more sense. Accordingly, MS_{exp} and MS_{ins} should be associated with non-zero and zero width error, respectively.

The examples above clearly show the drawbacks of the early approach to evaluating the performance of vessel segmentation algorithms. Due to the nature of curvilinear structures being thin and elongated regions, a pixel-wise comparison is not the most meaningful way of performance assessment. As a matter of fact, the overall performance measures TPR and FPR are both a mixture of two different aspects of performance, namely detection rate (how much of the vessel network structure is detected) and detection accuracy (what is the accuracy of the detected network structure). As shown in the examples above, such a mixture may result in a biased impression of the detection quality.

The non-structural nature of TPR and FPR has been implicitly acknowledged by other authors. Fang et al. (2005), for instance, state "Visual inspection is a way to examine extraction results for blood vessels in retinal images. Our method is able to recover an almost perfect morphological structure for high contrast images". Later, they use TPR and FPR to measure the detection quality without any further discussion about the accuracy of structure detection. This simply means that despite of the use of TPR and FPR, they were only able to obtain some qualitative impression of structure detection accuracy by visual inspection.

Another problem is discussed in (Niemeijer et al., 2004): "A disadvantage is that in this way the wider vessels have a larger influence on the end result than the smaller vessels. [...] Many of the smallest vessels of the gold standard and the second manual segmentation are not or partly visible in the automatic segmentations. In an application where small vessel detection is critical a method with a lower overall accuracy could still be marked the better method if it would segment more of the smallest vessels". With the current definition of TPR and FPR we do not distinguish between wider and smaller vessels in any way and thus a more sophisticated performance assessment as formulated by the authors of (Niemeijer et al., 2004) is not possible.

Based on the discussion above we believe that the key of a more meaningful way of performance assessment is to separate the two factors detection rate and detection accuracy. In our previous work (Jiang and Mojon, 2002) we represent the structure of a vessel network by its thinned version of midline points of one pixel width. The structures of the GT and the MS vessel network are compared by a point matching process. Then, the detection rate is measured by TPR and FPR defined by means of the matched midline points, reflecting the structural differences of the two networks. Afterwards, the computation of detection accuracy is based on the width information of matched midline points. While this approach is exactly what we need to alleviate the problems addressed above,

¹ The retinal images typically have a limited field of view, mainly due to the curvedness of human retina. If needed, multiple images can be fused using image registration techniques to form a montage with a larger field of view.

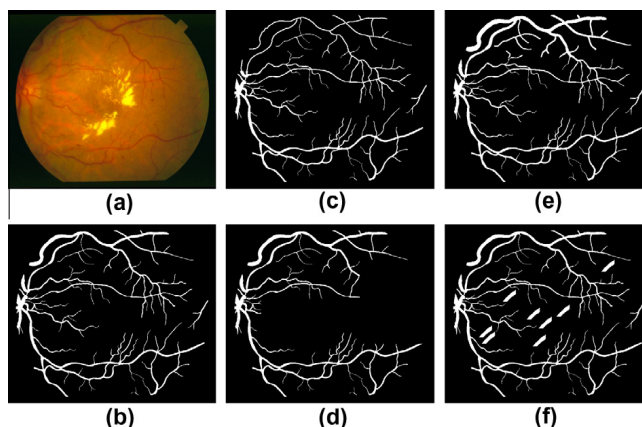


Fig. 1. (a) Retinal image; (b) GT; (c) MS_{min} : partial thinning of GT; (d) MS_{del} : deletions in GT; (e) MS_{exp} : partial expanding of GT; (f) MS_{ins} : insertions in GT.

the point matching process is an ad hoc one and results in many non-optimal matchings. In this work we further develop the approach of (Jiang and Mojon, 2002) by introducing an optimal point matching process. In addition the experimental validation has been substantially extended to demonstrate the usefulness of our approach.

Similar philosophy has been applied in (Niemeijer et al., 2010) to evaluate the detection accuracy of microaneurysms in retinal images. Instead of counting pixels the performance is measured by comparing microaneurysms as a whole (by using a simple correspondence procedure).

3. Structural evaluation methodology

The basic assumption is that for each test dataset (image or volume), we have a corresponding GT dataset with the curvilinear structures manually specified. Our approach is thus a supervised one.

Given a binary dataset V , for example a blood vessel image, we define its structure as the set of midline points along with the width information. Such a representation fully characterizes the curvilinear network by two information sources, allowing us to investigate the detection rate and the detection accuracy separately. We extract this structure in the following way:

- Find the midline points by computing the skeleton of V . The skeleton is denoted by V_s . There are many thinning and skeletonization methods, each with different properties. We use the method from Cardoner and Thomas (1997) because it guarantees that (a) the skeleton is thin (single-pixel wide), (b) the skeleton is connected, and (c) the skeleton can be used to reconstruct the original image with a tolerance of one pixel. Furthermore, it can be generalized to higher dimensions (Romero et al., 2000). Note that for elongated shapes like curvilinear structures under consideration in this work the relatively simple thinning algorithm from Cardoner and Thomas (1997) suffices; for thinning general binary shapes more sophisticated approaches like (Tang et al., 2010) will be needed.
- Compute a distance map of V : Each structure point (i.e., point from the skeleton V_s) is assigned its Euclidean distance d to the background. Then, each structure point p in V_s receives a

corresponding width value $w_p = 2d_p$. There are multiple methods to compute an exact squared Euclidean distance transform for this purpose in linear time. We chose the one presented in (Maurer et al., 2003).

Given a MS and GT, we propose to measure the detection rate by comparing MS_t and GT_t only, i.e., how much of the GT curvilinear network structure is detected in MS. In a second step the width of matched MS_t and GT_t structure points is compared to give a measure of detection accuracy.

The most crucial part of our approach is how to match GT_t and MS_t . We need to match as many structure points of GT_t as possible to structure points in MS_t and vice versa, in a way that ensures that two matched structure points are as similar as possible with respect to both their position and width. We formulate this problem as one of optimal graph matching.

3.1. Graph matching

A graph G is bipartite if its vertices form two disjoint subsets so that no edge exists between vertices in the same subset. The disjoint structure point sets GT_t and MS_t therefore form a bipartite graph G_{gm} if every edge connects a structure point $p \in GT_t$ with a structure point $q \in MS_t$. Such an edge represents a candidate for a match between the structure points p and q . Every edge is associated with cost that depends on the distance of p and q and on the difference of their width information.

A match between the two disjoint vertex sets of a bipartite graph is a set of edges so that each vertex is endpoint of at most one edge. Such a match will be called a *structural matching* in the following. In a structural matching, every structure point in GT_t is matched to at most one structure point in MS_t and vice versa.

The task can now be expressed as the problem of finding an optimal structural matching with minimum cost among all structural matchings with the maximum number of edges in G_{gm} . Such a matching is called a *maximum-cardinality minimum-cost matching*.

To build the graph G_{gm} we need to determine the set of match candidates and to specify their cost. Given the graph G_{gm} , we have to develop a procedure for finding its optimal structural matching \mathcal{M} . Based on this optimal structural matching we finally define a

new set of performance measures. The details of these steps are given in the following subsections.

3.2. Selecting match candidates

Not every pair (p, q) should be a match candidate: The Euclidean distance $d(p, q)$ should not be too high, and p, q should not represent structures of very different width.

To determine the match candidates and therefore the edges in G_{gm} , two thresholds d_{max} and w_{max} are necessary. A pair (p, q) is a match candidate if and only if

$$d(p, q) \leq d_{max} \wedge |w_p - w_q| \leq w_{max}$$

These thresholds are not independent of each other. In the case of thick structures, the allowed difference in position may be higher than in the case of thin structures, where it is more important to match the position exactly. To reflect this, w_{max} is determined from GT_i :

$$w_{max} = c_w \cdot \max\{w_p | p \in GT_i\}.$$

Then, d_{max} is determined from w_{max} :

$$d_{max} = c_d \cdot w_{max}.$$

The factors c_w and c_d are parameters and have to be chosen in advance. Details of choosing parameter values will be discussed in Section 3.6.

3.3. Cost of match candidates

For each match candidate $(p, q) \in G_{gm}$, $p \in GT_i$, $q \in MS_i$, its cost $c(p, q)$ should be proportional both to the Euclidean distance $d(p, q)$ and to the difference $|w_p - w_q|$ of the structure widths. Additionally, the cost should be normalized to $[0, 1]$ to ease the task of defining quality measures later. Because $d(p, q)$ is bounded by d_{max} and $|w_p - w_q|$ is bounded by w_{max} , the following definition fulfills these requirements:

$$c(p, q) = 1 - \left(1 - \frac{d(p, q)}{d_{max}}\right) \cdot \left(1 - \frac{|w_p - w_q|}{w_{max}}\right) \in [0, 1]$$

A good match candidate (p, q) means small value of $d(p, q)$ and $|w_p - w_q|$, which results in small $c(p, q)$. The minimum of $c(p, q)$, 0, is reached only in case of perfect matches. Note that this cost depends on the ground truth segmentation GT since w_{max} and d_{max} are computed from GT_i .

Based on this cost for match candidates, the cost $C(M)$ of a structural matching M between GT_i and MS_i is defined as follows:

$$C(M) = \sum_{(p,q) \in M} c(p, q) \in [0, |M|]$$

summing up the costs of all individual matches in M .

3.4. Computing optimal structural matching

The problem of determining a maximum-cardinality minimum-cost matching on G_{gm} can be expressed as a special case of the assignment problem for which efficient algorithms are known.

For this purpose, the problem can first be reduced to the computation of a minimum-cost perfect match in an auxiliary graph G'_{gm} ; see (Gabow and Tarjan, 1989), Section 202. (A perfect match in a bipartite graph $G = A \cup B$ is a match so that each vertex of A is matched to exactly one vertex of B and vice versa.) We form the auxiliary bipartite graph G'_{gm} as follows. It is created by putting G_{gm} and a copy of G_{gm} together. Then, we connect each vertex in G_{gm} with its copy and each such new edge is assigned the cost $N \cdot c_{max}$, where N is the number of vertices in G_{gm} and c_{max} is the

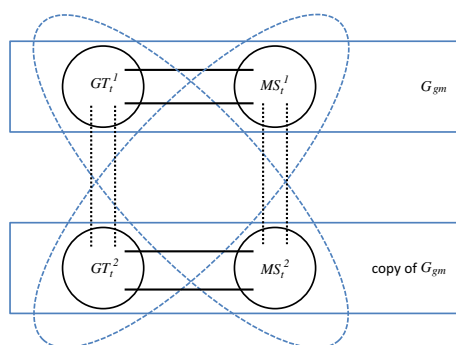


Fig. 2. Construction of auxiliary graph G'_{gm} : The dotted lines represent the edges connecting each vertex in G_{gm} with its copy.

maximum cost assigned to an edge in G_{gm} (in our case $c_{max} = 1$). G'_{gm} is again bipartite with the two disjoint vertex sets $GT_i^1 \cup MS_i^2$ and $MS_i^1 \cup GT_i^2$ (each represented by a dashed ellipse in Fig. 2), where GT_i^1 and MS_i^1 are the vertices of GT and MS part of G_{gm} , respectively, and GT_i^2 and MS_i^2 are the corresponding vertices from the copy of G_{gm} . It can be shown that G'_{gm} contains a minimum-cost perfect match, which corresponds to a maximum-cardinality minimum-cost match in G_{gm} when all edges that end in a vertex of the copy of G_{gm} are eliminated. More details can be found in (Gabow and Tarjan, 1989).

The maximum-cardinality minimum-cost match in G_{gm} is the optimal structural matching \mathcal{M} we are looking for. The remaining problem of finding a minimum-cost perfect match in G'_{gm} is a special case of the assignment problem and can be solved efficiently using the Cost Scale Assignment (CSA) algorithm from Goldberg and Kennedy (1995).

3.5. Quality measures

Based on the optimal structural matching \mathcal{M} we define the following quality measures.

True positives: The successfully detected structure points of GT_i are those that have a corresponding structure point in MS_i according to the optimal structural matching \mathcal{M} . The portion of these successfully detected structure points in \mathcal{M} to the total number of structure points in GT_i is the *true positives rate* (TPR):

$$TPR = \frac{|\mathcal{M}|}{\# \text{ structure points in } GT_i}$$

The definition of TPR tells us how much of the GT curvilinear network structure is successfully detected in the machine segmentation. A measure of the matching quality of the true positives is the *detection error* (DE):

$$DE = \frac{C(\mathcal{M})}{|\mathcal{M}|} \in [0, 1]$$

Recall that $C(\mathcal{M})$ is the cost function bounded by $[0, |M|]$ with zero indicating the best case. The detection error can also be split into two values to separately measure the *position error* (PE) and *width error* (WE) of the successfully detected curvilinear structure:

$$PE = \frac{1}{|\mathcal{M}|} \sum_{(p,q) \in \mathcal{M}} d(p, q); \quad WE = \frac{1}{|\mathcal{M}|} \sum_{(p,q) \in \mathcal{M}} |w_p - w_q|$$

Lower values of DE, PE, and WE correspond to higher accuracy. Note that all three error measures are scaled by $1/|\mathcal{M}|$ because they are

intended to represent the error per pair of correctly detected structure point and its GT correspondence.

False positives: Those structure points of MS_i that have no match in GT_i according to \mathcal{M} are false positives. The *false positives rate* (FPR) is defined as:

$$FPR = \frac{\# \text{ structure points in MS}_i - |\mathcal{M}|}{\# \text{ non-structure points in FOV of GT}_i}$$

Note that the number of non-structure points in the denominator may alternatively be counted in GT_i. In this work we follow the more reasonable convention from Niemeijer et al. (2004) to restrict the consideration to FOV only.

In addition to compute the false positive rate (equivalently the number of false positives in MS_i) it is also interesting to ask about the characteristic, for instance the width, of these spurious structures. It is probably more problematic to erroneously detect thick structures than thin ones. To obtain this information we can establish a width histogram of the false positives.

False negatives: Those structure points of GT_i that have no match in MS_i according to \mathcal{M} are false negatives. The *false negatives rate* (FNR) is defined as:

$$FNR = \frac{\# \text{ structure points in GT}_i - |\mathcal{M}|}{\# \text{ structure points in GT}_i}$$

This measure indicates how much of the GT structure is missing in the machine segmentation. Likewise we can investigate the width characteristics of these false negatives by a width histogram.

The structure-based evaluation procedure is summarized as follows.

```

/* detection rate */
construct GTi from GT and MSi from MSi;
find the optimal structural matching  $\mathcal{M}$  between GTi and MSi;
NoOfTruePositives =  $|\mathcal{M}|$ ;
NoOfFalsePositives = (# structure points in MSi) -  $|\mathcal{M}|$ ;
NoOfFalseNegatives = (# structure points in GTi) -  $|\mathcal{M}|$ ;
TPR = NoOfTruePositives / (# structure points in GTi);
FPR = NoOfFalsePositives / (# non-structure points in FOV of GTi);
FNR = NoOfFalseNegatives / (# structure points in GTi);
/* detection error */
Compute position error PE of true positive structures;
Compute width error WE of true positive structures;
    
```

3.6. Choosing parameter values

Two parameters c_d and c_w are used during the selection of match candidates. These parameters affect the number of match candidates as well as their cost, and therefore also the optimal match \mathcal{M} and the induced quality measures. Fortunately, it turns out that the quality measures are fairly robust to parameter changes.

Since there is no stringent reason to treat distance in position differently from difference in width, $c_d = 1$ and therefore $d_{\max} = w_{\max}$ is a suitable choice. This leaves only c_w to be determined.

The influence on the number of match candidates is more critical than the influence on their cost. Since the optimal matching \mathcal{M} is a maximum-cardinality match, too many match candidates inevitably lead to nonsense matches in \mathcal{M} . Therefore, c_w must not be chosen too large. On the other hand, c_w must not be chosen too small either, to avoid the exclusion of reasonable match candidates. Suppose the segmentation algorithm tends to mark structures wider than they really are. Then a small value of c_w quickly leads to the exclusion of reasonable match candidates.

It turns out that all reasonable match candidates are already included for $c_w \approx 0.5$: The true positives rate conforms to the

expectations. The higher TPR observed for increasing values of c_w comes at the cost of match quality: The position error value PE and the width error value WE increase rapidly already for $c_w \approx 1$. A study of this parameter reported in Section 4.3 indicates quite stable results for $c_w \in [0.4, 0.7]$. For this reason, a good parameter choice is $c_w = 0.5$, $c_d = 1$ and the experimental results reported in Section 4 are based on this parameter setting.

In principle, it would also be possible to choose w_{\max} and d_{\max} locally. For the structure point $p \in GT_i$ currently under examination, one could choose $w_{\max} = c_w \cdot w_p$ and $d_{\max} = c_d \cdot w_{\max}$. The expectation would be that the selection of match candidates adapts to the nature of the different structure regions and therefore further improves the quality measures. It turns out, however, that this method is very sensitive to the choice of parameters. Furthermore, the quality measures show no significant improvements even when suitable values are found. For this reason we will consistently apply the global version with $c_w = 0.5$, $c_d = 1$ in all experiments reported in the next section.

4. Experimental results

The motivation of our work is to develop a structure-based evaluation methodology so that we can overcome the bias problems discussed in Section 2. A series of experiments using both synthetic and real data have been conducted to demonstrate the effectiveness of our approach.

4.1. Synthetic data

First we show how our method evaluates the four images in Fig. 1(c)–(f), see Table 1. As wanted, MS_{thin} has TPR near 100%, implying a full detection of the vessel network structure. The fact that the detected vessels are thinner than GT is expressed by the width error 0.452 (pixel). The width error indirectly results in a position error 0.191. Note that TPR is not perfectly 100% because the thinned version MS_{thin} generally results in a skeleton which slightly differs from that of GT. In contrast MS_{del} leads to TPR = 77.6% only and accordingly 22.4% of the vessel network structure undetected. Since no error has been added to the correctly detected 77.6% of the vessel network in synthesizing MS_{del}, the error measures are all negligible in this case. The missing vessels in MS_{del} are expressed by the high number of false negatives 1729, meaning that 1729 of the structure points of GT_i cannot be matched to the segmentation result. In comparison MS_{thin} only has 32 missing structure points. Based on the histogram of false negatives we see further that the missing vessels are relatively thin; 62.2% of the missing parts have a width up to 3. The

Table 1
Evaluation results for MS_{thin}, MS_{del}, MS_{exp}, and MS_{ins}.

	MS _{thin}	MS _{del}
TPR	99.6%	77.6%
Detection error DE	0.060	0.000
Position error PE	0.191	0.002
Width error WE	0.452	0.002
False positives	0	0
False negatives	32	1729
FN histogram	–	1–2:50.3%, 2–3:11.9%, 3–4:25.0%, 4–5:9.4%
	MS _{exp}	MS _{ins}
TPR	100.0%	100.0%
Detection error DE	0.068	0.000
Position error PE	0.226	0.000
Width error WE	0.462	0.000
False positives	6	400
FP histogram	–	1–2:8.0%, 2–3:4.0%, 3–4:4.0%, ≥4:84.0%
False negatives	0	0

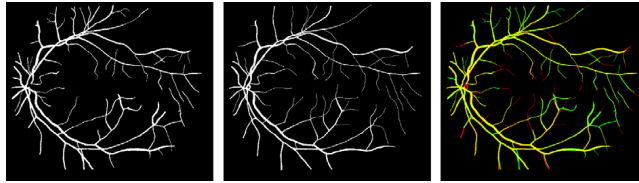


Fig. 3. Two hand-labelings of a retinal image from STARE database and their color overlay for making the tiny differences clearer. The first hand-labeling (left) is coded in green and the second (middle) in red. Common labeling points thus appear yellow. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

interpretation of these evaluation measures is exactly what we postulated for more informative and precise performance evaluation in contrast to the pixel-wise evaluation method.

Similar improvement can also be observed for MS_{exp} and MS_{ins} . Although both MS_{exp} and MS_{ins} have 100% TPR, MS_{exp} is not a perfect detection. The partial expanding in MS_{exp} results in a width error 0.462, indicating some detection inaccuracy. Basically, no false positive is found in MS_{exp} , compared to 400 in MS_{ins} . Furthermore, the spurious vessels in MS_{ins} are mainly thick ones with 84.0% being at least 4 pixels wide (corresponding to the thick added diagonal vessels). The measures on this second image pair again confirm our impression of the segmentation results and demonstrate the more informative and precise nature of the proposed evaluation methodology.

4.2. STARE database

The STARE database (Hoover et al., 2000) contains 20 images of digitized slides (available at <http://www.parl.clemson.edu/stare/probing/>, 700×605 pixels, 8 bits per color channel). There are two hand-labelings made by two different persons (computer scientists with knowledge in ophthalmology), see Fig. 3 for an example. The first hand-labeling, which is usually used as ground truth in performance evaluation (Hoover et al., 2000; Jiang and Mojon, 2003; Staal et al., 2004), took a more conservative view of the vessel boundaries and in the identification of small vessels than the second hand-labeling.

4.2.1. Single image case (STARE)

We start with the single retinal image shown in Fig. 4, together with the corresponding ground truth, and vessel detection MS_1 and MS_2 from two different algorithms (Jiang and Mojon, 2003 and Martinez-Perez et al., 1999). Using the pixel-wise evaluation we obtain: MS_1 : TPR = 91.9%, MS_2 : TPR = 80.3%. There is a large difference (11.6%) in TPR. The evaluation measures based on our approach are: MS_1 : TPR = 89.3%, MS_2 : TPR = 87.2%. Actually, MS_1 only detects 2.1% more of the vessel network structure than MS_2 . The much larger difference of 11.6% above is explained by the fact that MS_1 tends to be thicker than GT. Thus, it produces a better pixel-wise matching result. Measured by our method, this is expressed by a larger width error for MS_1 (1.129) than MS_2 (0.693). Here our performance measures clearly provide a more precise description of the differences between algorithmic results and ground truth.

4.2.2. Whole database (STARE)

We compare our evaluation approach with the early pixel-wise method in three different situations. The first hand-labeling is used as ground truth in all three of them. The results are summarized in Fig. 5.

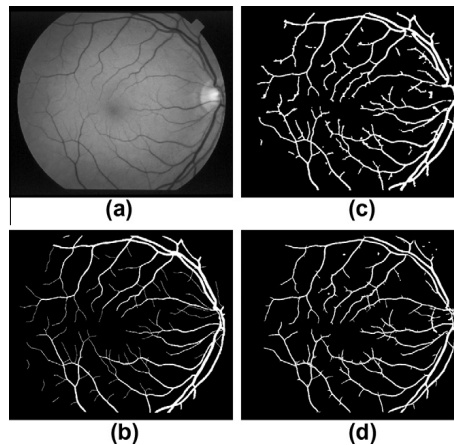


Fig. 4. (a) Retinal image; (b) GT; (c) first detection results MS_1 ; (d) second detection result MS_2 .

Verification-based adaptive local thresholding (Jiang and Mojon, 2003): This vessel detection method has been evaluated on the STARE database. Based on eight parameter sets the ROC is plotted in Fig. 5 (“multi-threshold probing”). Note that the 20 retinal images are divided into a subset of normal and a subset of abnormal cases. The performance study thus can be done for three test instances (all, normals, abnormal). In this case both evaluation methods have similar TPR values. The reason lies in the fact that the results from Jiang and Mojon (2003) tend to be thicker than the ground truth. Therefore, as soon as some part of the vessel network is detected, most of the vessel pixels of that part will be marked, leading to a local TPR value near 100% comparable to the local TPR from our evaluation approach. On the other hand, the FPR has much smaller values due to the use of spurious midline pixels only in our approach instead of all spurious vessel pixels.

Piecewise threshold probing of matched filter response (Hoover et al., 2000): For this method only one result per image for a particular parameter set is available. Looking at Fig. 5 (“filter response analysis”), we see that our evaluation method rates the TPR considerably more positive (from lower than 70% to almost 80%). The low TPR value of pixel-wise evaluation is caused by the algorithm’s tendency of not fully marking all local vessel pixels even if the middle part, thus the local network structure, is correctly found. Our structure-based evaluation approach considers the aspects of structure

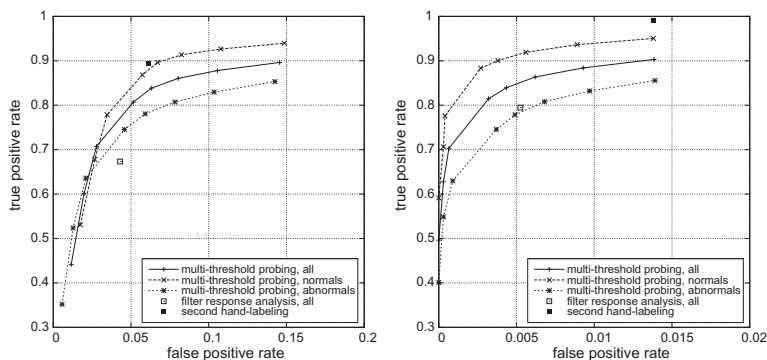


Fig. 5. Evaluation of multi-threshold probing, filter response analysis, and second hand-labeling on STARE database: pixel-wise evaluation (left) and our approach (right). In our case the FPR has much smaller values due to the use of spurious midline pixels only instead of all spurious vessel pixels.

Table 2

Evaluation results for comparing the second labeling in Fig. 3 against the first labeling.

TPR	92.2%
Detection error DE	0.375
Position error PE	1.573
Width error WE	0.634
False positives	955
FP histogram	1–2:90.1%, ≥2:9.9%
False negatives	558
FN histogram	1–2:70.1%, 2–3:5.9%, 3–4:17.4%, ≥4:6.6%

detection and local detection accuracy separately and is therefore able to characterize the behavior of an algorithm more precisely.

Second hand-labeling: In (Hoover et al., 2000; Jiang and Mojon, 2003) the second hand-labeling has been used as “machine-segmented result images” and compared to the first hand-labeling. The detection performance measures are then regarded as a target performance level. In Fig. 5 this level is indicated by an isolated mark in each plot (“second hand-labeling”). Although the second observer masked the vessels more completely, the pixel-wise TPR only amounts to about 90% because the second labeling is partly thinner than the first one. This assessment is obviously against our intuition and expectation. Using our approach the TPR increases to almost 100%.

Table 2 gives the details of this comparison for the retinal image shown in Fig. 3. The pixel-wise evaluation results in a TPR value of only 66.0% for this image. On the other hand, our approach indicates that a much higher rate of 92.2% of the vessel network structure has been correctly segmented by the second observer. The large divergence is caused by the differences in position and width of the marked vessels by the two observers, which is signified by quite large position and width errors in our case. The second observer labels more small vessels. This is documented by the number of false positives, namely 955. Among them 90.1% are midline pixels of thin vessels of one pixel width. This example makes once more our way of assessing the detection quality clear.

4.3. Study of parameter c_w

Our approach has one major parameter c_w . We have conducted a study using varying parameter values based on the multi-threshold probing method and the STARE database; see Fig. 6. The results indicate quite stable behavior for $c_w \in [0.4, 0.7]$. For this

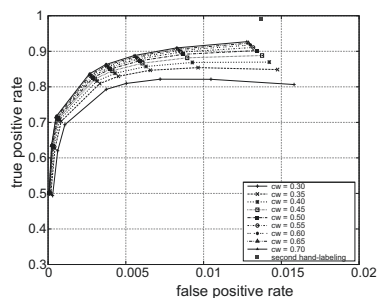


Fig. 6. Performance measure for varying values of parameter c_w (multi-threshold probing, STARE).

reason the parameter setting $c_w = 0.5$ was chosen and used for all experimental results reported in this paper.

4.4. Robustness issues of skeletonization

The basis for our proposed evaluation method is the skeletonization of detected vessel networks. Two robustness issues are considered here.

Image rotation: Ideally, the computed skeleton should be invariant to image translation and rotation. While translation invariance is mostly satisfied, the rotation invariance is more challenging. The relatively simple thinning algorithm from (Cardoner and Thomas, 1997), which is used in our current implementation, produces slightly different results for rotated binary images. To investigate the influence of rotation-dependent thinning results to the performance measures, we conducted the following experiment. The detection result in Fig. 4(c) was compared four times with the GT in Fig. 4(b): The original pair and counterclockwise rotated versions by 90° , 180° , and 270° . The resulting three main rates are listed in Table 3. It can be concluded that the influence of rotation-dependent thinning is not significant and thus using a thinning algorithm like (Cardoner and Thomas, 1997) should not have any impact on ranking vessel segmentation methods.

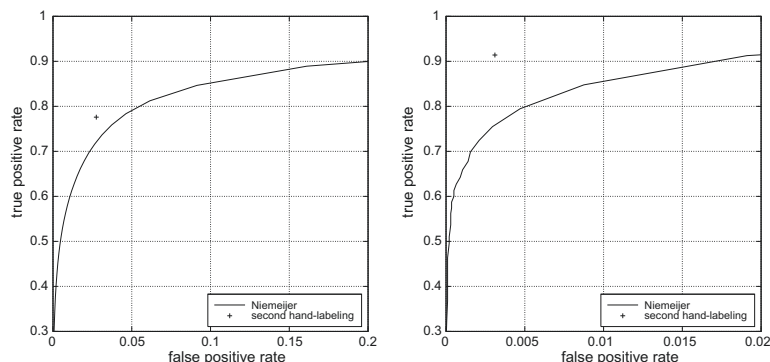


Fig. 7. Evaluation on DRIVE database: pixel-wise evaluation (left) and our approach (right). The term “Niemeijer” denotes the vessel detection method from Niemeijer et al. (2004). In our case the FPR has much smaller values due to the use of spurious midline pixels only instead of all spurious vessel pixels.

Spurious branches: Skeletonization potentially produces minor spurious branches. This phenomenon is not only observable in the relatively simple thinning algorithm (Cardoner and Thomas, 1997) used in our current implementation. Even more sophisticated thinning methods like (Tang et al., 2010) cannot fully avoid it. For the task under consideration, however, these spurious branches are of such small proportion that they should not significantly influence the performance measures. This expectation has been well confirmed by our experiments, in which such spurious branches were manually removed from both detection result and GT. The performance measures with and without the manual removal of spurious branches hardly differ.

4.5. DRIVE database

The DRIVE database consists of 40 images (available at <http://www.isi.uu.nl/Research/Databases/DRIVE/>, 768 × 584 pixels, 8 bits per color channel). The pixel classification approach to vessel detection from Niemeijer et al. (2004) has been evaluated using both methods, see Fig. 7. Similar to the evaluation of the verification-based adaptive local thresholding (Jiang and Mojon, 2003) on the STARE database, the performance measure TPR does not change too much in this case. Large changes occur, however, if we compare two different hand-labelings. Similar arguments apply here as well. Our structure-based evaluation indicates more precisely the structure detection rate.

4.6. Computational time

For comparing a pair of GT and MS image, the skeletonization takes about 2.5 s (STARE) and 1.6 s (DRIVE), respectively, on a standard notebook. In both cases the matching time is about 1.5 s. No code optimization has been tried so far. For the non-realtime task of performance evaluation the computation time is acceptable.

Table 3
Performance measures of rotated detection results.

	Original	90°	180°	270°
TPR	89.28%	89.22%	89.40%	89.14%
FPR	0.10%	0.10%	0.10%	0.09%
FNR	10.72%	10.78%	10.60%	10.86%

5. Discussions and conclusion

Compared to other commonly used features such as edges and regions, there is relatively little work on performance evaluation of algorithms for curvilinear structure detection. In this paper we have proposed a novel structure-based methodology for this purpose. We consider the two aspects of performance, namely detection rate and detection accuracy, separately, in contrast to their mixed handling in earlier approaches that typically produces biased impression of detection quality. By doing so, the proposed performance measures give us a more informative and precise performance characterization. The detailed information about width, for instance, helps the user select a suitable algorithm for a particular application as discussed in Section 2.

Both synthetic and real examples have been used to demonstrate the advantages of our approach. In fact, the use of our evaluation approach may change our thinking about the relative performance of algorithms. Concerned with the vessel detection method from Hoover et al. (2000), for instance, although the evaluation could only be done for one parameter setting, the results in Section 4.2 (i.e., the increased TPR value according to our structure-based evaluation) indicate that this algorithm has a higher “intrinsic” detection rate than assumed so far based on the pixel-wise comparison. The superior performance of the algorithm (Jiang and Mojon, 2003) is at least partly caused by its tendency of producing thicker vessels. More experiments (using vessel detection results not available to us yet) will be needed to fully clarify this point. But this incomplete comparison shows the potential of our approach to directing the evaluation to the intrinsic algorithmic performance.

It is not our intention in this work to conduct a rather complete performance evaluation for a large number of algorithms. Instead, the experiments reported in Section 4 are shown to demonstrate the principal usefulness of our structural performance evaluation. We will do the evaluation work in future by involving other researchers in a joint effort.

The description of the evaluation methodology and the experimental work have been embedded in the context of blood vessel detection in retinal images. It is important to point out that our approach is applicable in the general context of the evaluation of curvilinear structure detection algorithms. In particular, extraction of airway tree and other thin structures in volumetric data (Holtzman-Gazit et al., 2006; Tschirren et al., 2005) is a challenging task

and our evaluation technique will help assess the algorithm performance as well.

Acknowledgments

The authors want to thank A. Hoover, M. Niemeijer and his colleagues for making their retinal image databases and the ground truth data publicly available. They also provided us some of their experimental results. A. Goldberg and R. Kennedy kindly provided their CSA implementation for public use. Thanks also go to Peter Larysch for his support in experimental work.

References

- Bharkad, S.D., Kokare, M., 2011. Performance evaluation of distance metrics: Application to fingerprint recognition. *Int. J. Pattern Recog. Artif. Intell.* 25 (6), 777–806.
- Cardoner, R., Thomas, F., 1997. Residuals + directional gaps=skeletons. *Pattern Recog. Lett.* 18 (4), 343–353.
- Cardoso, J.S., Sousa, R., 2011. Measuring the performance of ordinal classification. *Int. J. Pattern Recog. Artif. Intell.* 25 (8), 1173–1195.
- Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., Goldbaum, M., 1989. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Trans. Med. Imag.* 8 (3), 263–269.
- Fang, B., You, X., Tang, Y.Y., Chen, W.S., 2005. Morphological structure reconstruction of retinal vessels in fundus images. *Int. J. Pattern Recog. Artif. Intell.* 19 (7), 937–948.
- Frame, A., Undrill, P., Cree, M., Olson, J., McHardy, K., Sharp, P., Forrester, J., 1998. A comparison of computer based classification methods applied to the detection of microaneurysms in ophthalmic fluorescein angiograms. *Comput. Biol. Med.* 28, 225–238.
- Gabow, H., Tarjan, R., 1989. Faster scaling algorithms for network problems. *SIAM J. Comput.* 18 (5), 1013–1036.
- Goldberg, A., Kennedy, R., 1995. An efficient cost scaling Algorithm for the assignment problem. *Math. Prog.* 71, 153–178.
- Heath, M.D., Sankar, S., Sanocki, T., Bowyer, K.W., 1997. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Trans. PAMI* 19 (12), 1338–1359.
- Holtzman-Gazit, M., Kimmel, R., Peled, N., Goldsher, D., 2006. Segmentation of thin structures in volumetric medical images. *IEEE Trans. Image Process.* 15 (2), 354–363.
- Hoover, A., Goldbaum, M., 2003. Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels. *IEEE Trans. Medical Imag.* 22 (8), 951–958.
- Hoover, A., Kouznetsova, V., Goldbaum, M., 2000. Locating blood vessels in retinal images by piece-wise threshold probing of a matched filter response. *IEEE Trans. Med. Imag.* 19 (3), 203–210.
- Jiang, X., 2005. Performance evaluation of image segmentation algorithms. In: Chen, C.H., Wand, P.S.P. (Eds.), *Handbook of Pattern Recognition and Computer Vision*, 3rd ed. World Scientific, pp. 525–542.
- Jiang, X., Mojon, D., 2002. Supervised evaluation methodology for curvilinear structure detection algorithms. In: *Proc. 16th Int. Conf. on Pattern Recognition*, vol. 1, 2002, pp. 103–106.
- Jiang, X., Mojon, D., 2003. Adaptive local thresholding by verification-based multi-threshold probing with application to vessel detection in retinal images. *IEEE Trans. PAMI* 25 (1), 131–137.
- Jiang, X., Marti, C., Irniger, C., Bunke, H., 2006. Distance measures for image segmentation evaluation. *EURASIP J. Appl. Signal Processing*, Special Issue on Performance Evaluation in Image Processing, 1–10.
- Jiang, X., Lambers, M., Bunke, H., 2011. Structure-based evaluation methodology for curvilinear structure detection algorithms. In: Jiang, X., Ferrer, M., Torsello, A. (Eds.), *Graph-Based Representations in Pattern Recognition*, LNCS, vol. 6658. Springer, pp. 305–314.
- Lam, B.S.Y., Yan, H., 2008. A novel vessel segmentation algorithm for pathological retina images based on the divergence of vector fields. *IEEE Trans. Med. Imaging* 27 (2), 237–246.
- Lam, B.S.Y., Gao, Y., Liew, A.W.-C., 2010. General retinal vessel segmentation using regularization-based multiconcavity modeling. *IEEE Trans. Med. Imaging* 29 (7), 1369–1381.
- Lee, S.U., Chung, S.Y., Park, R.H., 1990. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing* 52, 171–190.
- Martinez-Perez, M., Hughes, A., Stanton, A., Thom, S., Bharath, A., Parker, K., 1999. Scale-space analysis for the characterization of retinal blood vessels. In: *Proc. Medical Image Computing and Computed Assisted Intervention (MICCAI)*, 1999, pp. 90–97.
- Maurer, C.R., Qi, R., Raghavan, V., 2003. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. PAMI* 25 (2), 265–270.
- Niemeijer, M., Staal, J.J., van Ginneken, B., Loog, M., Abramoff, M.D., 2004. Comparative study of retinal vessel segmentation methods on a new publicly available database. In: Fitzpatrick, J., Sonka, M. (Eds.), *SPIE Medical Imaging*, vol. 5370, 2004, pp. 648–656.
- Niemeijer, M., van Ginneken, B., Cree, M.J., et al., 2010. Retinopathy online challenge: Automatic detection of microaneurysms in digital color fundus photographs. *IEEE Trans. Med. Imag.* 29 (1), 185–195.
- Romero, F., Ruos, L., Thomas, F., 2000. Fast skeletonization of spatially encoded objects. In: *Proc. 15th Int. Conf. on Pattern Recognition*, vol. 3, 2000, pp. 510–513.
- Rothaus, K., Jiang, X., Rhiem, P., 2009. Separation of the retinal vascular graph in arteries and veins based upon structural knowledge. *Image Vision Comput.* 27 (7), 864–875.
- Shufelt, J.A., 1999. Performance evaluation and analysis of monocular building extraction from aerial imagery. *IEEE Trans. PAMI* 21 (4), 311–326.
- Staal, J., Abramoff, M.D., Niemeijer, M., Viergever, M.A., van Ginneken, B., 2004. Ridge-based vessel segmentation in color images of the retina. *IEEE Trans. Med. Imag.* 23 (4), 501–509.
- Tang, Y., Bai, X., Yang, X., Lin, L., Liu, S., Jan Latecki, L., 2010. Skeletonization with particle filters. *Int. J. Pattern Recog. Artif. Intell.* 24 (4), 619–634.
- Tschirren, J., Hoffman, E.A., McLennan, G., Sonka, M., 2005. Intrathoracic airway trees: segmentation and airway morphology analysis from low-dose CT scans. *IEEE Trans. Med. Imaging* 24 (12), 1529–1539.
- Zana, F., Klein, J., 1999. A multimodal registration algorithm of eye fundus images using vessels detection and Hough transform. *IEEE Trans. Med. Imag.* 18 (5), 419–428.
- Zana, F., Klein, J., 2001. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Trans. Med. Imag.* 20 (7), 1010–1019.

A stereoscopic movie player with real-time content adaptation to the display geometry

Sylvain Duchêne^a, Martin Lambers^b, Frédéric Devernay^{*a}

^aPRIMA team, INRIA Grenoble Rhône-Alpes, France

^bComputer Graphics and Multimedia Systems Group, University of Siegen, Germany

ABSTRACT

3D shape perception in a stereoscopic movie depends on several depth cues, including stereopsis. For a given content, the depth perceived from stereopsis highly depends on the camera setup as well as on the display size and distance. This can lead to disturbing depth distortions such as the cardboard effect or the puppet theater effect. As more and more stereoscopic 3D content is produced in 3D (feature movies, documentaries, sports broadcasts), a key point is to get the same 3D experience on any display. For this purpose, perceived depth distortions can be resolved by performing view synthesis. We propose a real time implementation of a stereoscopic player based on the open-source software Bino, which is able to adapt a stereoscopic movie to any display, based on user-provided camera and display parameters.

Keywords: Stereoscopy, 3D video, depth perception, depth-preserving disparity mapping, view synthesis, real time, stereoscopic player

1. INTRODUCTION

Many media are now available in stereoscopic 3D without offering the best viewing condition due to the depth distortion that appear on different displays due to the variety of screen sizes and distances. However this distortion may be reduced through view synthesis methods, which usually involve three steps¹: computing the stereo disparity, applying a disparity-dependent mapping to the original views and compositing the resulting images. In this paper, we do not focus on the first step: while stereo disparity computation is still an active research topic, state-of-the-art methods give results that are satisfying for our application.

In the first part of this paper, we focus on the main constraints on the camera setup used to shoot a stereoscopic movie for a given display configuration. Then, we discuss the choice of our view synthesis method through a view synthesis model built on shooting and viewing geometries. We make the assumption that the movie is correctly rectified: a pixel from the left image corresponds to a pixel on the same horizontal line in the right image, i.e. there is no vertical disparity or vertical misalignment. In the second part, we present our pipeline and an efficient way to implement it on the GPU using several render passes satisfying video frame rate performance and high quality. Finally, we discuss our results, the problems that may appear, such as visual artifacts, and how to solve them.

Our main research goals and contributions are to propose a stereoscopic movie player performing real-time content adaptation to the display geometry with a framework using both CPU and GPU.

2. SHOOTING CONSTRAINTS DESCRIPTION AND CONTENT ADAPTATION

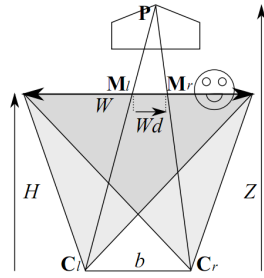
Projecting a stereoscopic movie on different screen sizes and distances will produce different perceptions of depth, which implies that a movie is shot for a particular display configuration. If a stereoscopic movie is viewed without modification, three main issues have to be considered: eye divergence, image scaling and roundness factor².

Figure 1 shows the various geometric parameters describing the camera and display configurations.

*frederic.devernay@inria.fr; phone +33 4 76 61 52 58; www.inrialpes.fr

This work was done within the 3DLive project, supported by the French Ministry of Industry www.3dlive-project.com

S. Duchêne, M. Lambers and F. Devernay "Stereoscopic movie player with real-time content adaptation to the display geometry", Stereoscopic Displays & Applications XXIII, Woods, Holliman, Favalora (eds.), Proc. SPIE 8288, paper 8288-15, 2012. ©2012 Society of Photo-Optical Instrumentation Engineers. Systematic or multiple reproduction, distribution, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper are prohibited.



Symbol	Camera	Display
C_l, C_r	camera optical center	eye optical center
P	physical scene point	perceived 3-D point
M_l, M_r	image points of P	screen points
b	camera interocular	eye interocular
H	convergence distance	screen distance
W	width of convergence plane	screen size
Z	real depth	perceived depth
d	left-to right disparity (as a fraction of W)	

Figure 1. Shooting and viewing geometries can be described using the same small set of parameters.

Since triangles $M_l P M_r$ and $C_l P C_r$ are homothetic, the disparity d can be written as a function of the real depth Z :

$$d = \frac{b Z - H}{W Z} , \text{ or } Z = \frac{H}{1 - \frac{dW}{b}} \quad (1)$$

2.1 Eye divergence

The perceived depth can be expressed by the same way since the horizontal disparity in the images and the screen disparity respectively expressed as a fraction of image width and fraction of screen width are equal: $d = d'$. This results in:

$$Z' = \frac{H'}{1 - \frac{dW'}{b'}} \quad (2)$$

By eliminating the disparity d , from (1) and (2), we obtain:

$$Z' = \frac{H'}{1 - \frac{W'}{b'} \frac{bZ - H}{WZ}} \quad (3)$$

Eye divergence occurs when $Z' < 0$, i.e. $d' > bW'$; object placed at infinity in the real scene ($Z \rightarrow \infty$) will cause eye divergence if and only if $W'/b' > W/b$.

2.2 Scale ratio

The image scale ratio σ' represents how much an object placed at depth Z seems to be enlarged (eg. $\sigma' > 1$) or reduced (eg. $\sigma' < 1$) in X and Y direction with respect to objects present in the convergence plane $s = \frac{H}{Z}$.

$$\sigma' = \frac{s'}{s} = \frac{H'}{Z'} \frac{H}{Z} = \frac{1 - dW'/b'}{1 - dW/b} \quad (4)$$

On-screen objects get a scale ratio equal to 1 since the disparity is 0. Moreover the relation between Z and Z' is non linear except if $\frac{W'}{b'} = \frac{W}{b}$.

2.3 The roundness factor

The roundness factor ρ measures how much the object proportions are affected for an object of dimensions $\delta X, \delta Z$ in the width and the depth directions at a depth Z , perceived as an object of dimensions $\delta X', \delta Z'$ at depth Z' .

$$\rho = \frac{\delta Z' / \delta X'}{\delta Z / \delta X} = \frac{\delta Z' / W'}{W' / s'} = \sigma' \frac{W}{W'} \frac{\delta Z'}{\delta Z} \quad (5)$$

The roundness factor of an object on the screen plane ($Z = H$ and $Z' = H'$) is:

$$\rho_{screen} = \frac{W}{W'} \frac{\delta Z'}{\delta Z} = \frac{b H'}{H b'} \quad (6)$$

Note that despite common belief, the on-screen roundness factor does not depend on the screen size, but on the screen distance. This on-screen roundness factor is equal to 1 if and only if $b'/b = H'/H$.

In order to get perfect shape reproduction, the roundness factor should be equal to 1 everywhere. This implies that the geometric parameters have to satisfy $b'/b = W'/W = H'/H$. Consequently, the only camera configurations that preserve the roundness factor everywhere are scaled versions of the viewing geometry².

2.4 View synthesis model

A generalized depth preserving view synthesis model built on shooting and viewing geometries parameters which satisfies both described constraints can be written as follows³.

A pixel in the left image which has coordinates (x_l, y) and disparity d_l maps to the following pixel in the interpolated view:

$$(x_c + (x_l + wd_l - x_c)\sigma''(d_l) + w(d''(d_l) - d_l), y_c + (y - y_c)\sigma''(d_l)), \quad (7)$$

where w is the image width in pixels, d'' is the mapping from the original disparity to the synthesized disparity:

$$d''(d) = \frac{H'b}{(HW' - H'W)d + H'b} \quad (8)$$

and σ'' is the disparity-dependent image scaling:

$$\sigma''(d) = \frac{H'b}{(HW' - H'W)d + H'b}. \quad (9)$$

Similarly, a pixel in the right image which has coordinates (x_r, y) and disparity d_r maps to the following pixel in the interpolated view:

$$(x_c + (x_r - x_c)\sigma''(d_r) + w(d''(d_r) - d_r), y_c + (y - y_c)\sigma''(d_r)) \quad (10)$$

Note that σ'' and d'' depend on the parameters that describe the shooting and viewing geometries.

As we only need to synthesize one view to maximize the quality³, we prefer a simpler depth-preserving disparity mapping synthesis method over view synthesis method. *Depth-preserving disparity mapping* preserves the depth proportions with a roundness factor equal to 1 for on-screen object, and it avoids eye-divergence, although the disparity-dependent image scaling σ'' is not applied, resulting in off-screen objects having a roundness different than 1.

Expressions (7) and (10) become:

$$(x_l + w * d''(d_l), y) \quad (11)$$

and

$$(x_r + w * (d''(d_r) - d_r), y), \quad (12)$$

and the pixel-dependent blending factors to be applied to the two warped images are:

$$\alpha_i^l = |d_i^r| / (|d_i^l| + |d_i^r|), \quad \alpha_i^r = 1 - \alpha_i^l \quad (13)$$

The computation of the disparity d'' at each pixel can be easily parallelized on a GPU architecture, since it is a closed-form computation that only depend on values at that pixel.

3. RENDERING PIPELINE OVERVIEW

Most view synthesis algorithms require a stereo pair and the associated disparity map(s), but rendering a movie at video frame rate imposes some constraints on the performance required. Our approach can be divided into a sequence of steps:

1. We make sure than the stereo input video sources are rectified. This ensures that each pixel in the left image corresponds to a pixel on the same horizontal line in the right image.
2. We compute the disparity maps from this stereo sequence using a coarse-to-fine method. This method can be implemented on the GPU⁴ and run almost at video-rate on today's GPUs. Then we encode the disparity map in a video stream using the luminance channel.

3. Both the disparity and the video streams are sent to the stereoscopic movie player. The CPU decodes each stream and the two decoded image pairs (left and right images, left-to-right and right-to-left disparity maps) are sent to the GPU.
4. Two views are rendered from the stereo pair displaced using vertex buffer grid geometry, and depth discontinuities are handled by using transparency.
5. We then composite these views to obtain a stereo pair adapted to the viewing conditions and sent to the display using a standard stereoscopic display stream format (Top/bottom, Above/Below, Checkerboard...)
6. An optional pass can be inserted before conversion to the stereoscopic format, in order to remove artifacts using a confidence map³.

Figure 2 describes graphically the data flow from the original images and disparity maps to the synthesized image pair.

3.1 Implementation framework

To implement our real-time solution we used the framework provided by the open-source stereoscopic movie player Bino⁵ which already implements many required features in a unified framework, for example multithreaded and synchronized video stream decoding. It also supports different stereoscopic display formats, such as top/bottom, above/below, and checkerboard, and offers a linear OpenGL pipeline. The decoding is processed through the open-source library FFmpeg, which supports a large number of codecs as well as multi-threaded decoding on the CPU.

Moreover, the framework takes care of using linear RGB pixel values over the pipeline, which is essential during all the process⁶. Intermediate color buffers may lose precision in lower value ranges when stored as 8-bit linear images. For this reason, either 8-bit sRGB images or 16-bit linear-RGB images are used as intermediate buffers.

Our disparity map was produced with a coarse-to fine dense stereo matching algorithm, which is suitable for real-time performance⁴ to be as close as possible to an embedded solution in a stereo movie player. Both left and right disparity maps are encoded in a gray scale video. All the videos were encoded using the H.264 codec for its quality, performance and popularity as it's widely used by many streaming Internet sources. As a disparity can be either positive or negative, an encoding range needs to be chosen to define the 0 disparity.

3.2 Asymmetric precomputation pass

The first rendering pass converts the disparity video input stream by applying the correct scale and offset, and performs the computation of the texture element displacement. Since recent OpenGL versions allow the use of multiple render targets, left and right streams can be processed on the same pass sharing the same GLSL shader using two framebuffer to store the result. This pass does not use any diverging branches, which makes it very fast. xd_l^i and xd_r^i represent the displacement from a point x from L to I and from R to I.

Input Textures: left and right disparity maps
 Input parameters: viewing and shooting geometries parameters

$$\begin{aligned} xd_l^i &= w * d''(d_l) \\ xd_r^i &= w * (d''(d_r) - d_r) \end{aligned}$$

$$\begin{aligned} \alpha_l^i &= |d_r^i| / (|d_l^i| + |d_r^i|) \\ \alpha_r^i &= 1 - \alpha_l^i \end{aligned}$$

Output_buffer [0]: xLI, $d''(d_l)$, d_l , α_l^i
 Output_buffer [1]: xRI, $d''(d_r) - d_r$, d_r , α_r^i

Note that the current pixel position isn't applied yet.

All the computation is done in the fragment shader, since the vertex shader only needs to transfer the texture coordinates.

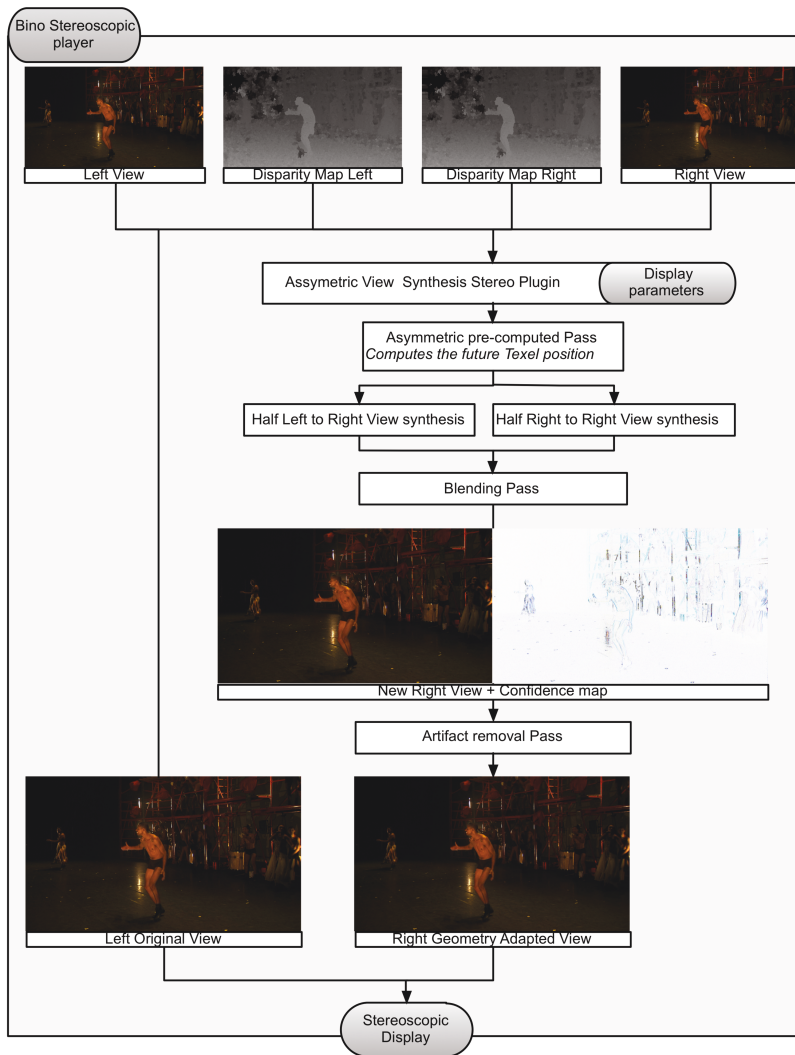


Figure 2. Rendering pipeline overview: the two views and the two disparity maps are uploaded as textures on the GPU, and all subsequent processing is performed on the GPU.

3.3 Rendering half views

Image based rendering can be performed with explicit geometry or with implicit geometry. Our use case is closer to view interpolation than to free viewpoint methods, as our goal is to adapt the content using disparity mapping⁷, which can be seen as a depth-dependent baseline interpolation method.

Using a grid model where each vertex is displaced by the OpenGL pipeline rasterizer makes sense, since we can also handle occlusions that way, by using the disparity itself as the z-component and using a Z-buffer. We preferred using standard OpenGL shaders rather than Cuda or OpenCL in the pipeline, because switching the GPU between the OpenGL and the Cuda/OpenCL modes takes too much time on consumer-level graphics cards. By using only one large indexed vertex buffer object grid in HD (1920*1080) made from a single triangle strip, we can render both view in two passes. The grid is initialized with texture coordinates corresponding to an identity transform.

Input Textures: `assymetric_framebuffer (xdil, dr(di), di, αil)`
Input parameters: `threshold (1.0)`

$$x_i^l = gl_{Vertex}.x + xd_i^l$$

$$gl_{Position} = gl_{ModelViewProjectionMatrix} * (x_i^l, gl_{Vertex}.y, -d^r(d_i), 1.0)$$

The vertex shader applies the displacement on the x-axis for each vertex from the grid using the mapping from the previous pass but also on the z-axis using only the synthesized disparity to handle self-occlusions of the displaced grid.

Detecting depth discontinuities is easy as the previous buffer can be sampled in its neighborhood: when $||xd_i^l - x_{-1}d_i^l|| > \text{threshold}$, the blending factors has to be set to 0.0. By default the threshold has to be 1.0 to detect any elongated triangles in the displaced grid.

Output [0]: RGBvalues from left view, blending factor

The fragment shader just applies the color from the texture through the textures coordinates attached to each displaced vertex. Note that left and right borders must be refilled using the last candidate with a blending factor set to 0.0. Figure 3 shows the intermediate result with the two rendered half-views.

3.4 Blending

The two synthesized views are blended together using the asymmetric disparity mapping blending factors, combined with the discontinuity-detection blending factors.

Input : left and right contribution to synthesize the new view

$$\alpha_i^l = |d_i^r| / (|d_i^l| + |d_i^r|) \quad \alpha_i^r = 1 - \alpha_i^l$$

Output: compose views using blending factors

4. RESULTS

The resulting software plays videos fluently at Full HD resolution (1080p25) on a quad-core 2.8GHz with a GeForce GTX480 GPU. The overall quality is good, but some visual artifacts may appear because of errors present in the disparity maps. Many of these artifacts can actually be detected by image processing⁸, and we are working on GPU-based artifact removal methods that could be used for real-time rendering.



Figure 3. Left view mapped to the synthesized view (top), right view mapped to the synthesized view (bottom).

5. CONCLUSION AND FUTURE WORK

We presented a real-time stereoscopic movie player with content adaptation to the display geometry through a framework that unifies video streams decoding on the CPU and depth preserving view synthesis into a set of chained algorithmic building blocks on the GPU. Future work should be focus on real-time artifact removal methods, and on the computation and transmission of the disparity maps.

The disparity maps should be computed on the fly from the left and right video streams, and can either be transmitted together with the left and right videos, or could be computed on the client side (for example on a set-top box).

Another subject of interest is to detect which shots can be left untouched (so that artifacts caused by view synthesis can be avoided completely), based on the amount of divergence and on the global amount of depth distortions caused by viewing the unmodified shot.



Figure 4. New composed view with depth-preserving synthesis method.

REFERENCES

- [1] Sammy Rogmans, Jiangbo Lu, Philippe Bekaert, and Gauthier Lafruit, "Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity GPUs," *Signal Processing: Image Communication*, 24(1-2):49–64 (2009). ISSN 0923-5965. doi: 10.1016/j.image.2008.10.005. Special issue on advances in three-dimensional television and video.
- [2] Frédéric Devernay, Sylvain Duchêne, and Adrian Ramos-Peon, "Adapting stereoscopic movies to the viewing conditions using depth-preserving and artifact-free novel view synthesis," *Stereoscopic Displays and Applications XXII*, volume 7863 (2011). SPIE. doi: 10.1117/12.872883.
- [3] Frédéric Devernay and Sylvain Duchêne, "New view synthesis for stereo cinema by hybrid disparity remapping," *Proc. International Conference on Image Processing (ICIP)*, pages 5–8, Hong Kong, (2010). doi: 10.1109/ICIP.2010.5649194.
- [4] M. Sizintsev, S. Kuthirummaly, S. Samarasekeray, R. Kumary, H. S. Sawhney, and A. Chaudhry, "GPU accelerated realtime stereo for augmented reality," *Proc. Intl. Symp. 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.
- [5] Martin Lambers, "Bino: free 3D video player" <http://bino3d.org/> Accessed: december 2011
- [6] Larry Gritz and Eugene d'Eon, "The Importance of Being Linear," *GPU Gems 3*, Chapter 24, Hubert Nguyen (ed.), Addison-Wesley (2007), ISBN 0321515269
- [7] Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus Gross, "Nonlinear disparity mapping for stereoscopic 3D," *ACM SIGGRAPH 2010 papers, SIGGRAPH '10*, pages 75:1–75:10 (2010). ISBN 978-1-4503-0210-4. doi: 10.1145/1833349.1778812.
- [8] Frédéric Devernay and Adrian Ramos-Peon, "Novel view synthesis for stereoscopic cinema: detecting and removing artifacts," *Proc. 1st international workshop on 3D video processing, 3DVP '10*, pp. 25–30 (2010). ACM. ISBN 978-1-4503-0159-6. doi: 10.1145/1877791.1877798.