Multi-Sensor State Estimation for Autonomous Navigation of Micro Aerial Vehicles in GNSS Reception Deprived Environments

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von
M.Sc. Nasser Gyagenda

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2022

Betreuer und erster Gutachter
Prof. Dr.-Ing. Dr. h. c. Hubert Roth
Universität Siegen

Zweiter Gutachter
Prof. Vadim Zhmud
Novosibirsk State Technical University (NSTU)

Tag der mündlichen Prüfung
30.09.2022

Abstract

Navigation is a key capability of Micro Aerial Vehicles (MAVs). It includes perception, localization, motion control, cognition and obstacle avoidance as the main competences. It may be accomplished by an external operator or onboard flight management system, known as remotely piloted and autonomous systems respectively. The agility of MAVs and complexity of their operating environments have favoured autonomous navigation solutions, which are predominantly Global Navigation Satellite System (GNSS)-based, over remotely piloted solutions. But GNSS technology is susceptible to intentional and unintentional interferences, which challenges have motivated the quest for GNSS-independent autonomous navigation solutions.

Although navigation is supported by several competences, this research focuses on cognition, specifically, the problem-solving intellectual function. Within the navigation task, one of the main problem solvers is the path planner. The key requirement of any path planner is the ability to find feasible paths. Additionally, for MAVs with the inability to conserve power while searching for a path online, the planner ought to be fast and scale well with the environment. This led to the first goal of developing an online path planner with such performance. Another problem to solve arises when the planned path length exceeds the MAV's endurance, a case common in coverage tasks. For this, a coverage path planner capable of accounting for vehicle endurance in relation to path length and environment size is required. Lastly, autonomous functioning has enabled deployment of mobile robots on our world and beyond. But knowing the right amount of autonomy required to complete a given task is still a challenge. Several autonomy evaluation frameworks have been proposed over the last three decades, but most of these offer a low resolution categorical output or have inconsistent metrics, raising the need for a better autonomy framework.

A path planning ensemble consisting of three concurrently executed single query randomized sampling-based path planners has been proposed for online path planning. A partitioning path planner capable of exact cellular decomposition of large areas of interest into manageable cells using Voronoi decomposition, planning coverage paths and scheduling them on a MAV or a fleet of either homogeneous or heterogeneous MAVs has been proposed as well. Last but not least, a set of four autonomy evaluation metrics, namely capabilities, trust factor, performance capacity and environmental complexity, and their associated mathematical models have also been proposed.

Tested in a physics supported graphical simulator, the proposed path planning ensemble demonstrated query adaptability, a high path finding success rate and a short path planning time, suitable for online path replanning with allowance for path smoothing. Also, the lack of implicit environment representation by sampling-based planners meant that the ensemble planner scales well with the environment. The plausibility of the proposed large-scale coverage path planner has been ascertained through a Software-In-the-Loop (SIL) test. Such a planner guarantees coverage, ensures proper resource management and proper mission planning. The autonomy evaluation framework has been tested on three case studies, which together ascertained the plausibility of its models. This framework provides a systematic approach for development and regulation of autonomy.

Zusammenfassung

Die Navigation ist eine der wichtigsten Fähigkeiten von Mikro-Luftfahrzeugen (MAVs). Sie beinhaltet die Hauptkompetenzen Wahrnehmung, Lokalisierung, Stabilitätskontrolle, Kognition und Hindernisvermeidung. Die Navigation kann von einem externen Bediener oder einem bordseitigen Flugmanagementsystem ausgeführt werden, die als ferngesteuerte bzw. autonome Systeme bekannt sind. Die Agilität von MAVs und die Komplexität ihrer Betriebsumgebungen haben autonome Navigationslösungen, die überwiegend auf dem Globalen Navigationssatellitensystem (GNSS) basieren, gegenüber ihren ferngesteuerten Pendants bevorzugt. Aber die GNSS-Technologie ist jedoch anfällig für beabsichtigte und unbeabsichtigte Störungen. Diese Herausforderungen haben die Suche nach GNSS-unabhängigen autonomen Navigationslösungen motiviert.

Obwohl die Navigation durch mehrere Kompetenzen unterstützt wird, liegt der Schwerpunkt hier auf der Kognition, Problemlösungskompetenz. Im Rahmen der Navigationsaufgabe ist einer der wichtigsten Problemlöser der Pfadplaner. Die Hauptanforderung an jeden Pfadplaner ist die Fähigkeit, realisierbare Pfade zu finden. Da MAVs jedoch nicht in der Lage sind, während des Flugs Energie zu sparen, ist auch ein schneller Online-Planer erforderlich, der ebenfalls unabhängig von der Kartengröße ist. Da die Flugzeit von MAVs begrenzt ist, wird ein möglichst schneller Online-Planer benötigt. Desweiteren soll die Berechnungsdauer möglichst unabhängig von der Kartengröße sein. Daraus wurde das erste Ziel abgeleitet: die Entwicklung eines Online-Pfadplaners mit oben genannten Fähigkeiten. Ein weiteres Problem ergibt sich, wenn die geplante Pfadlänge die Flugzeit des MAVs übersteigt, was bei Aufgaben, in denen ein Gebiet überwacht werden muss, häufig der Fall ist (engl. "coverage tasks"). Hierfür wird ein Pfadplaner benötigt, der die Flugzeit des Fahrzeugs in Abhängigkeit von der Pfadlänge und der Größe der Umgebung berücksichtigt. Schließlich hat die autonome Funktionsweise den Einsatz von mobilen Robotern in unserer Welt und darüber hinaus ermöglicht. Es ist jedoch nach wie vor eine Herausforderung, das richtige Maß an Autonomie zu finden, das ein System zur Erfüllung einer bestimmten Aufgabe benötigt. In den letzten drei Jahrzehnten wurden mehrere Methoden zur Bewertung der Autonomie vorgeschlagen, aber die meisten von ihnen bieten eine gering aufgelöste kategorische Ausgabe oder haben inkonsistente Metriken, was den Bedarf an einem besseren Framework erhöht.

Für die Online-Pfadplanung wurde ein Pfadplanungs-Ensemble vorgeschlagen, das aus drei gleichzeitig ausgeführten Pfadplanern besteht, die zur Kategorie der single-query random sampling Planern gehören. Dies ist ein partitionierender Pfadplaner, der in der Lage ist, eine exakte Zerlegung von großen Interessenbereichen in handhabbare Teilstücke zu zerlegen, indem eine Voronoi-Zerlegung angewendet wird. Anschließend erfolgt die Pfadplanung, die zur Abdeckung notwendig ist. Möglichkeiten zur Anwendung dieser Pläne auf ein einzelnes MAV oder homogene oder heterogene MAVs werden vorgeschlagen. Nicht zuletzt wurden auch vier Autonomiemetriken vorgeschlagen, nämlich Fähigkeiten, Vertrauensfaktor, Leistungskapazität und Umgebungskomplexität mit ihren dazugehörigen mathematischen Modelle.

In einem physikalisch unterstützten grafischen Simulator getestet, zeigte das vorgeschlagene Pfadplanungs-Ensemble die Anpassungsfähigkeit von Abfragen, eine hohe Erfolgsrate bei der Pfadfindung und eine kurze Pfadplanungszeit, die für die Online-Planung unter Berücksichtigung der Pfadglättung geeignet ist. Das Fehlen einer impliziten Umgebungsdarstellung bei stichprobenbasierten Planern bedeutete auch, dass der Ensemble-Planer gut mit der Umgebung skaliert. Die Plausibilität des vorgeschlagenen Pfadplaners mit großer Abdeckung wurde durch einen Software-In-the-Loop (SIL)-Test nachgewiesen. Ein solcher Planer garantiert die Abdeckung, sorgt für ein angemessenes Ressourcenmanagement und eine korrekte Missionsplanung. Die vorgeschlagene Methode zur Bewertung der Autonomie wurde an drei Fallstudien getestet, die zusammen die Plausibilität der Modelle bestätigten. Diese Bewertungsmethode bietet einen systematischen Ansatz für die Entwicklung und Regulierung von Autonomie.

Acknowledgements

The research work presented in this writing is a conglomeration of project results conducted at Lehrstuhl Regelungs- und Steuerungstechnik (RST) in the department of Electrical Engineering and Computer Science of the University of Siegen. Throughout the period of this research work, the author has interacted with numerous great people, who have influenced the results presented herein in more ways than can be described in words and deserve acknowledgement.

To start with, the author would like to express his utmost gratitude to his main supervisor Prof. Dr.-Ing. Dr. h. c. Hubert Roth for the opportunity to be part of his RST team and unwavering support, without which this work would not have been possible. The author would also like to thank Prof. Vadim Zhmud of the department of Automatics, Faculty of Automatics and Computer Techniques of Novosibirsk State Technical University (NSTU) for co-supervising this research work.

During his time at RST, the author conducted several research visits to Lahore University of Management Sciences (LUMS) under the invitation of Dr.-Ing. Ahmad Kamal Nasir, who throughout the PhD duration has been a constant source of advice and guidance on a number of robotics and machine learning related subjects.

The author has also received enormous support and assistance from the RST team members of the years 2016 through 2021. The author is grateful to each and every one of you. He thanks you all for your efforts and making his time at RST very rewarding and memorable. Furthermore, the author is grateful for the support he received from other members of the University of Siegen, with whom countless discussions directly related to this work and otherwise, were conducted.

It would be a great omission not to mention the anonymous reviewers whose insightful comments always opened ways for new perspectives that enriched the research results presented herein and author's personal experience.

Last but definitely not the least, the author would also like to express his deepest gratitude to his brothers and sisters, and friends scattered around the globe for their continued emotional and social support.

Nasser Gyagenda

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ACL	Autonomy Control Level	8
ACPP	Aerial Coverage Path Planning	57
AGAST	Adaptive and Generic Accelerated Segment Test	18
AGL	Above Ground Level	17
AI	Artificial Intelligence	26
AIS	Artificial Immune System	14
ALFUS	Autonomy Levels for Unmanned Systems	8
ANFIS	Adaptive Neuron Fuzzy Inference System	24
AOI	Area of Interest	57
ATRA	Autonomy and Technology Readiness Assessment	26
BiT-RRT	Bidirectional Transition-based Rapidly-exploring Random Tree	50
BRIEF	Binary Robust Independent Elementary Features	74
CAD	Computer-Aided Design	12
CPP	Coverage Path Planning	57
CUDA	Compute Unified Device Architecture	75
DARPA	Defense Advanced Research Projects Agency	43
DCM	Direction Cosine Matrix	31
DGPS	Differential Global Positioning System	20
DoA	Degree of Autonomy	36
DTMs	Digital Terrain Models	12
EKF	Extended Kalman Filter	24
ESA	European Space Agency	26
FAST	Features from Accelerated Segment Test	15
FMCW	Frequency Modulated Continuous Wave	17
GNN	Global Nearest Neighbour	17
GNSS	Global Navigation Satellite System	iii
GPS	Global Positioning System	2
IMU	Inertial Measurement Unit	11
INS	Inertial Navigation System	3
LoA	Level of Autonomy	36

LOS	Line-Of-Sight	3
MAV	Micro Aerial Vehicle	4
MEMS	Micro Electro-Mechanical Systems	11
MTOW	Maximum Take-Off Weight	29
NASA	National Aeronautics and Space Administration	26
NIST	National Institute of Standards and Technology	8
NLOS	Non-Line-Of-Sight	2
OMPL	Open Motion Planning Library	52
ORB	Oriented FAST and Rotated BRIEF	15
OS-CFAR	Ordered Statistics-Constant False Alarm Rate	17
PARS	Phased Array Radio System	14
PDST	Path-Directed Subdivision Tree	50
PNT	Positioning, Navigation and Timing	2
PRM	Probabilistic Roadmap Method	21
PRN	Pseudorandom Noise	3
RANSAC	Random Sample Consensus	74
RDTs	Rapidly-exploring Dense Trees	50
ROS	Robot Operating System	69
RRG	Rapidly-exploring Random Graph	22
RRT	Rapidly-exploring Random Tree	21
SCER	Security Code Estimation and Replay	3
SIFT	Scale Invariant Feature Transform	15
SIL	Software-In-the-Loop	64
SLAM	Simultaneous Localization and Mapping	12
SURF	Speeded Up Robust Feature	15
SWaP-C	Size, Weight and Power - Cost	11
TDOA	Time Difference of Arrival	19
TRL	Technology Readiness Level	26
T-RRT	Transition-based Rapidly-exploring Random Tree	50
TWR	Two-Way Ranging	19
UAS	Unmanned Aerial System	26
UAV	Unmanned Aerial Vehicle	1
UGV	Unmanned Ground Vehicle	9
UKF	Unscented Kalman Filter	67
UWB	Ultra-Wide Band	11
VTOL	Vertical Take-Off and Landing	21
VIO	Visual Inertial Odometry	14
VO	Visual Odometry	14

Notation

Matrices

Matrices are represented by uppercase alphabets with an underline. For example $\underline{A}$ represents a matrix in variable A. In this case $\underline{A} \in \mathbb{R}^{m\times n}$,

$$\underline{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} = (a_{ij}), \quad i = 1, \cdots, m \text{ and } j = 1, \cdots, n.$$

Vectors

Vectors are represented by lowercase alphabets with an underline. For example, $\underline{a}$ represents a vector in viable a. In this case $\underline{a} \in \mathbb{R}^2$,

$$\underline{a} = \begin{pmatrix} x \\ y \end{pmatrix}, \text{ or } \underline{a} = (x, y)^T$$

All vectors in this work are *column* vectors.

Sets

The concept of sets appears in a few parts of this work. In which case, curly brackets have been used to represent a set of elements. For example, $\{a_1, a_2, ...a_n\}$ is a set of n elements. Unlike in vectors, in sets the order of elements is not important. Some of the special sets used include $\mathbb{R}$ for a set of scalars, $\mathbb{R}^n$ for a set of n-dimensional vectors and $\mathbb{R}^{m\times n}$ for a set of $m \times n$ matrices.

Functions

Functions are represented by $f(\cdot)$. For example, $f(a, b) \mapsto c$ is a function that maps a set of parameters $\{a, b\}$ into a scalar value c.

Derivatives

Herein Newton's notation (the dot notation) has been adopted to represent derivative. For example, $\dot{x}$ represents the time derivative of a time function $x(t)$.

Direction Cosine Matrix

A Direction cosine matrix (DCM) is a matrix belonging to the special orthogonal group, which represents transformations between two reference frames. The matrices are represented with uppercase character $\underline{R}$ with a trailing superscript and a leading subscript. For example, ${}^{B}\underline{R}_{A}$ is a transformation from frame $\{A\}$ to frame $\{B\}$.

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) are members of the mobile robot family capable of six degrees of freedom motion in the aerial domain. These vehicles find applications in a number of fields that listing them all here would span quite a footprint. Interested readers are therefore refer to articles [7] and [8] dedicated to this subject. Being mobile robots, navigation is the most important competence of any Unmanned Aerial Vehicle (UAV) flight management system. It is a meta-capability, which includes perception, localization, motion control, cognition, obstacle avoidance and optionally mapping competences.

Having a human perform the main navigation subtasks with support from sensors constitutes the simplest navigation solution, but the expanding application possibilities have pushed UAVs into fields whose control demands surpass the capability of human-in-the-loop control mechanisms. This expansion has motivated the quest for human-independent navigation solutions, also known as autonomous navigation techniques. The UAV autonomous navigation problem has been of interest since 2007 [9] with the initial focus on GNSS-based solutions for open outdoor areas. But GNSS technology is vulnerable to spoofing, jamming, environmental effects [10], and absent or unreliable in indoor, underground, urban canyons, natural canyons and forest understories [11, 12]. These vulnerabilities then motivated the quest for GNSS-independent variants of human-independent navigation solutions. This is known as the GNSS-independent UAV autonomous navigation problem and is the focus of this work.

Solutions to the GNSS-independent UAV autonomous navigation problem have been mostly supported by technological breakthroughs in sensor miniaturization and processor performance, and algorithm development. This work reviews the existing solutions, their technology maturity level, and also proposes complete indoor navigation frameworks. The contribution in these frameworks is on the cognition sub-capability, represented by the path planning problem solver within the navigation task. Developing autonomous systems requires replacement of human skilled operators with electronic modules capable of delivering similar or better levels of performance. This requires characterization of tasks to enable establishment of a set of quantitative metrics associating task to operator capability requirements and then task to robot capability requirements. Unfortunately, no such metrics with consistent outputs exist. Therefore, this work additionally seeks to determine a set of metrics and their associated combination models for autonomy estimation. If available, such metrics would facilitate autonomy advancement and regulation.

Autonomy is purposive; it is defined with respect to a specific task. Herein, the focus is on autonomous navigation as a task. Therefore, autonomy in this work is studied with respect to the navigation task. The navigation problem is defined as follows: given a starting and a goal configuration or a set of goal configurations defined in the same frame of reference, a system should use prior knowledge if available or accumulated knowledge to plan and execute a feasible trajectory from a start to a goal configuration.

1.1 Motivation

Several solutions for the UAV autonomous navigation problem have been proposed over the years, but most reliable among them are GNSS technology supported. As mentioned earlier, GNSS technology is susceptible to intentional and unintentional interferences. These vulnerabilities are explained in details in Sect. 1.1.1, and are the motivating factors for research into GNSS-independent UAV navigation solutions, which includes the research presented herein. Furthermore, this work acknowledges the need for large-scale coverage path planners in Sect. 1.1.2, for fast online path planners in Sect. 1.1.3, and autonomy assessment for UAV autonomous navigation systems in Sect.1.1.4, and additionally, establishes the technology readiness level so far achieved after over a decade of research effort invested into solving the GNSS-independent UAV autonomous navigation problem. The latter is necessary in justifying the need for any additional research effort and pointing out the directions in which it is best suited. Overall, this work examines the status quo of GNSS-independent navigation solutions, as well as improving on their cognitive intellectual function.

1.1.1 GNSS Vulnerabilities

There are four main satellite constellations providing global navigation services namely, Global Positioning System (GPS), GLONASS, Galileo and BeiDou. The main services provided by these satellite systems include Positioning, Navigation and Timing (PNT), which support governments and consumer industries including aviation, automobile, agriculture, survey, military, railway, maritime, telecommunication, financial market, national security and energy sectors [13].

The wide spread applications of GNSS for civilian and military sectors have made it a potential target for malicious attacks. This claim is supported by the existence of special bands and encrypted codes like P(Y) code on L1 and L2 GPS bands, and Galileo Public Regulated Service (PRS) that were put in place to withstand malicious attacks and minimize the likelihood of GNSS blackouts for these bands. Actual evidence of adversarial attacks on GNSS systems have been presented by the German Aerospace Center in [14] and a survey on GNSS spoofing in [15]. Also, a GNSS spoofing demonstration on a Hornet Mini UAV is presented in [16].

Besides intentional malicious attacks, environmental and atmospheric effects like scintillation, solar activities, multiple paths and shadowing may attenuate the GNSS signal leading to breakage of the receiver-signal lock as a result of reduced signal-to-noise ratio [10]. The two main environmental effects are known as Non-Line-Of-Sight (NLOS) propagation for the case when a GNSS receiver receives reflected signals due to lack of direct paths between the satellites and receiver, and multipath propagation when a GNSS receiver receives signals propagated through two or more direct and indirect paths [17]. Zhang and Hsu demonstrated the multipath effects on GNSS positioning with a commercial GNSS receiver operating in urban Hong Kong in [18]. The next section provides an in-depth account of the intentional and unintentional GNSS interferences.

Intentional and Unintentional GNSS Interferences

Herein, the GNSS interferences are classified into two main categories namely, intentional and unintentional interferences. Intentional interferences result from devices and/or structures whose sole purpose is to impede proper functioning of GNSS receivers, and may take the form of jamming or spoofing. Jamming actions impede acquisition of a signal lock and/or breaks existing signal locks while spoofing actions generate counterfeit GNSS signals intended to manipulate the belief of a receiver. On the other hand, unintentional interferences are non-deliberate negative effects on GNSS signal quality resulting from other purposeful devices and/or structures within the vicinity of the GNSS receivers [13].

GNSS signals are very weak. Taking an example of GPS, it has a signal power of about 1.6×10^{-16} W. In typical environments, the background noise is on orders of magnitude greater than this. As a result, typical jammer-to-signal-ratio is usually greater than one, $J/S > 1$ [15]. Most jammers emit a continuous chirp signals at GNSS frequencies boosting the background noise substantially, which makes it difficult to detect and lock onto a satellite signal. Additionally, empty GNSS Pseudorandom Noise (PRN) codes could be broadcast to penetrate anti-jamming filters [15, 13]. High power jammers are easily identifiable from their high-energy signature, but the low power jammers are quite challenging to identify because this type of jamming could be a result of unintended signal blockage by structures like trees and buildings in the environment.

Unlike jamming which is a brute force approach, spoofing takes a deceptive approach. Spoofing signal generators broadcast imitative GNSS signals at a suitable strength and frequency to break the receiver-signal lock. These imitative signals are normally simulated or derived from legitimate signals with varying degrees of manipulation. Manipulation could take the form of capture-delay-retransmit or capture-selective delay-retransmit also known as meaconing and Security Code Estimation and Replay (SCER) or selective delay respectively [15]. Like low power jamming, signal delays might result from spectral signal reflection off structures like buildings and the ground during low altitude flights.

But as reported in [15, 16, 13], spoofing is a very complicated process requiring precise knowledge of the position and velocity of the target to be able to generate spoofing signals that are synchronised with GNSS received at the target. This makes it a less likely threat for the highly agile UAVs. For military applications, spoofing is even less likely, thanks to the availability of anti-spoofing capable GNSS military receivers with encrypted P(Y) signal. Jamming on the other hand is an eminent threat as this could even occur naturally from structural shielding and radio frequency interference. To address these problems, several GNSS interference countermeasures presented in the next section are commonly employed by the GNSS community.

GNSS Interference Countermeasures

GNSS interference countermeasures (specifically for radio interferences) have been launched on two fronts namely, interference detection and interference mitigation fronts [13]. The former focuses on development of algorithms for recognizing irregularities in the GNSS signal that are attributable to interferences. The latter category includes interference mitigation techniques, which can be classified into four categories namely, signal processing, cryptographic, antenna technology and correlation with other positioning systems [15].

The mitigation techniques listed above are effective in scenarios where access to legitimate GNSS signal is still possible. Following this assumption, GNSS augmentation solutions have been proposed over the years, for example GPS/Inertial Navigation System (INS) [19, 20, 21]. Like most radio-based systems, GNSS operates under Line-Of-Sight (LOS) assumption—LOS assumption is violated in urban canyons, natural canyons, forests understories, tunnels and indoors as these environments may occlude GNSS signals partially or completely. In case of complete GNSS occlusions, the above-mentioned mitigation strategies fail [22]. This case calls for GNSS-independent navigation solutions, which are the focus of this work. Furthermore, with the increasing likelihood of GNSS blackouts for civilian users, it has become imperative to provide a redundant GNSS-independent localization system even for GNSS-based navigation implementations for increased reliability.

1.1.2 Large-Scale Coverage Path Planning

According to the PwC global report on commercial applications of unmanned aerial technology [1], the two leading UAV application industries namely, infrastructure and agriculture account for more than a half of the global market share Fig. 1.1. Clearly, all these applications are large-scale and coverage in nature. Large-scale coverage industrial applications often exceed the coverage capability of most modern Micro Aerial Vehicle (MAV)s. Luckily, the market price of Micro Aerial Vehicles (MAVs) have dropped substantially over the years enabling acquisition of multiple platforms. The aggregate capability of a fleet of such MAVs can easily satisfy most of these large-scale coverage applications. It should be noted that even with a single platform, multiple flights can be conducted in a short period of time, owing to the manoeuvrability and low operating costs of these aerial platforms. To harness the cumulative power in numbers, coverage path planners with integrated partitioning schemes are necessary for systematically partitioning large areas into manageable portions, generating coverage paths for each partition and then scheduling them on a fleet of MAVs or flying them with one platform multiple times.

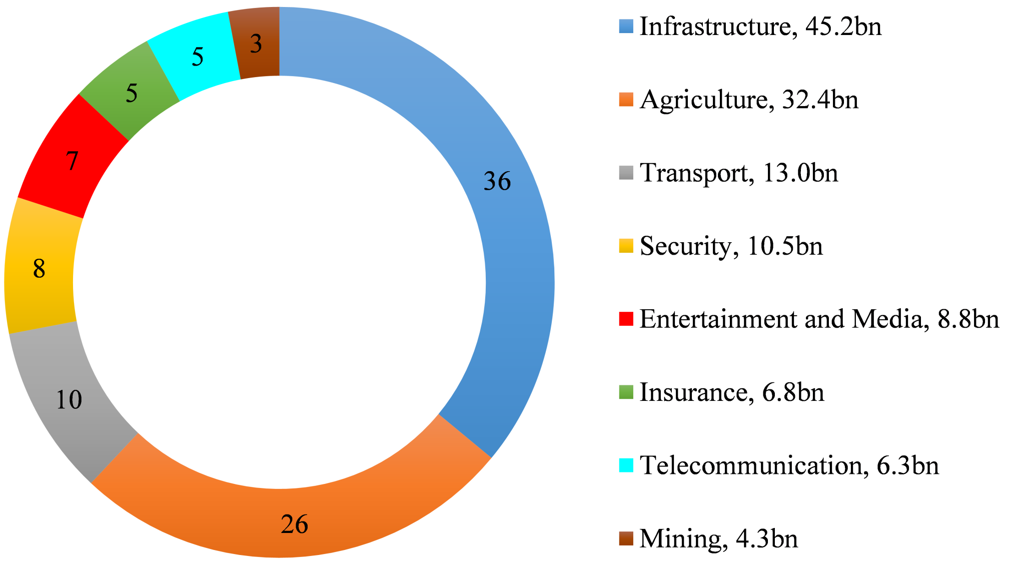


Figure 1.1: Predicted market value of UAV powered industrial solutions in US dollars [1].

1.1.3 Fast Online Path Planning

Different UAV operating environments afford different navigation solutions. This work focuses on building-like environments, which provide a structured environment. The structured nature comes at a cost of narrow pathways, increased collision possibilities and access points whose state can change from one moment to the next. An example of the latter is a doorway, which could be open in one moment and closed in the next. From a path planner perspective, such state superposition is impossible to work with without real-time environmental state updates. This can be circumvented by providing the planner with an environmental model whose doorways are in an “open” state, then the vehicle replans online on observing the door as impassable. This necessitates availability of a fast online path replanning algorithm. Of course, the MAV vehicle class considered in this work is capable of hovering to wait for path replannig to complete, but this waiting time has to be minimized to avoid wasting of the already scarce onboard power. The planner should be able to plan directly in 3D, enabling traversal of multi-floor building environments.

1.1.4 Autonomy Assessment

Autonomous functioning has enabled deployment of highly dynamic robotic systems in dangerous and complex areas on our world and other worlds like Mars. Knowing the right amount of autonomy required for a system to complete a given task in such environments with or without human intervention is a challenging necessity. A number of methods for quantifying autonomy have been proposed over the last three decades, but most of these either offer a low resolution categorical output (level classification) or have poor metrics with inconsistent outputs. Therefore, there is a need for a set of quantitative autonomy metrics that is easily measurable, broad enough to capture autonomy evolution in a system and yields consistent outputs with good output resolution.

The deficiency of the existing frameworks can be exemplified by the SAE International's levels of driving automation published in SAE J3016 [23]. This framework classifies driving automation systems into six levels, from level 0 that supports no driving automation to level 5 that supports full driving automation. The assessment process asks at most five yes-no questions, with answers based on qualitative performance knowledge for that vehicle. If the vehicle does not fulfil the performance requirement of level 5, it is assessed for level 4, and so on until a yes is obtained or level 0 is reached. Such a framework assumes that all vehicles of a specific level have the same degree of performance, but this is not true as technology advancements and economic factors may lead to varying performance even at a similar level. Against this background, it is asserted that autonomous systems differ functionally as well as performance wise. Therefore, this works seeks to establish a set of autonomy correlated metrics covering both functionality and performance, their mathematical representations and a mapping function into a single measure of autonomy.

1.2 Research Scope

This research studies the autonomous navigation problem. The navigation problem is addressed differently in different operating domains. This work focuses on navigation in the aerial domain. Depending on the structure delimiting the aerial domain, different structural configurations afford different navigation technologies. For example, outdoor open spaces afford GNSS navigation, built-up areas provide perspective line features that afford visual navigation and indoor areas are dominated by vertical walls that afford map-based localization, to mention but a few. This work focuses on building-like indoor environments, for which the prominent GNSS navigation technology is not applicable, due to its absence or unreliability.

Navigation is a meta-capability of mobile robots, supported by a number of competences including perception, localization, cognition, motion control, obstacle avoidance and optionally mapping. For UAVs, the availability of off-the-shelf autopilots is testimonial to the achieved superiority in motion control. Hence, this competence is not addressed in this research work. For perception and obstacle detection, several sensor technologies are deployed, with disregard for their working principles, but their environmental effects are considered. Therefore, the two main competences of focus are cognition and localization. It should be noted that mapping is used in this work for visual localization.

For GNSS-challenged environments, a number of autonomous navigation solutions have been proposed over the years. The techniques applied in these solutions are studied and mapped back onto the general autonomous navigation problem structure to tell the level of technology maturity so far achievable, as well as revealing any open questions, for which answers are then proposed. These questions range from structural properties, algorithmic to overall performance of GNSS-independent navigation solutions. The questions guiding this research include the following:

- How to develop complete indoor GNSS-independent UAV autonomous navigation solutions?
- How to achieve fast online motion planning and replanning?
- How to generate motion plans for large-scale aerial coverage scenarios?
- How to develop high fidelity MAV navigation simulators?
- How to measure the autonomy of a UAV for a specific task?
- What is the actual technology readiness maturity level of GNSS-independent UAV autonomous navigation solutions?

1.3 Outline of the Text

The work presented herein covers a wide range of topics centred around the subject of GNSS-independent UAV autonomous navigation. For the interest of understandability, the presented body of text has been organized into eight coherent chapters. Besides the current chapter, **Chapter 1**, the content of the other seven chapters is as follows:

Chapter 2—puts this thesis work in context with existing and related literature. For the interest of organisation, the literature has been clustered into informative sections including autonomy, UAV navigation, localization, motion planning, integrity monitoring, technology readiness assessment of navigation systems and trends in GNSS-independent navigation for UAVs.

Chapter 3—focuses on aerial platforms, taking the reader through the journey of classifying UAVs and selecting the most suitable class and configuration for this work. This chapter is also divided into sections including UAV classification, vehicle selection, and quadcopter dynamic and kinematic motion models.

Chapter 4—this chapter describes the thought pattern applied in the selection of autonomy metrics and designing of mathematical models for mapping them into a single measure of autonomy. Here autonomy is defined as a two-part measure, with the constituent parts being level of autonomy and degree of autonomy. The chapter content is presented in two main sections namely, autonomy measurement and autonomy framework evaluation.

Chapter 5—here, the focus is on motion planners capable of generating feasible paths for special scenarios like fast online path planning in multi-floor building environments, and large-scale coverage path planning with a single vehicle or fleets of homogenous and heterogeneous vehicles. The chapter content is divided into two main sections namely, randomized sampling-based path planners and large-scale aerial coverage path planning.

Chapter 6—this chapter describes the localization techniques implemented and tested as part of the proposed complete navigation frameworks. These include ultra-wideband-aided inertial localization and visual odometry-aided inertial localization. In the chapter are also their associated implementation descriptions, simulation and experimental results.

Chapter 7—this is a chapter on modelling and simulation. It offers a description of mathematical and geometric models for the different sensors and vehicle platforms applied in this work. Additionally, it describes modelling and embedding of the test environment model into the Unity software framework, enabling utilization of its realistic physics rigid-body behaviour in simulations.

Chapter 8—this is the last chapter of this body of text. It provides a summary of concluding remarks of the works presented as well as proposals for future courses of action.

Chapter 2

Literature Review

If I have seen further it is by standing on the shoulders of Giants.

Sir Isaac Newton

This chapter covers literature related to the research problems addressed in this thesis, which are related to the navigation problem in autonomous aerial vehicles. As the title suggests, it intersects a number of study fields including autonomy, state estimation, localization, path planning, obstacle avoidance, and Unmanned Aerial Vehicle (UAV) classification, control and navigation simulation. The subsequent sections cover literature related to each of these study fields, starting with autonomy.

2.1 Autonomy

Autonomous robots are sought after in a number of applications due to their benefits, among which include handling of complex applications and attaining time critical responses [24], replacing humans [25], operational domain qualification and delineation [26], enhance system capabilities, basis for control architecture selection [27], safety risk assessment, operator training and licensing [6], characterizing autonomous technologies and assess technology maturity [28], and reducing human factor effects and operating mission costs while maximising mission success [29]. These have motivated a global surge in the development and deployment of autonomous systems, resulting in a diverse selection of autonomous systems with a wide range of capabilities, raising regulatory, safety and ethical concerns. These concerns are solvable through proper engineering practices and regulatory laws, but their enforcement requires the ability to characterize and/or measure autonomous functioning.

A number of methods for quantifying autonomy have been proposed over the last three decades. The derivations of these frameworks are founded on the researcher's definition of autonomy, which differs for different research groups and institutions. The next paragraphs report on some of such frameworks found relevant and interesting with respect to the problem statement of this work.

One of the earliest works on autonomy we could find is in [24], where autonomy is defined as a measure of supervision. Based on this definition, they measured autonomy using communication channel bandwidth. The logic of the bandwidth metric is as follows, the less external commands the system requires during operation and the less data the system sends to an external controller for interpretation, the more control tasks it is capable of handling onboard, hence, the higher its level of autonomy. This approach assumes all commands to have equal importance to the task, which might be true only for very simple tasks. Additionally, the entropy of each command may differ resulting in varying amounts of information per command.

Autonomy Levels for Unmanned Systems (ALFUS) workgroup of National Institute of Standards and Technology (NIST) defines autonomy as a system's ability to use its root autonomous capabilities to achieve its assigned goal [30]. Their proposed framework differentiates between autonomy and level of autonomy, where the latter is a measure of the degree of human intervention and the former is a function of human independence, mission complexity and environmental complexity scores. To be more precise, they define level of autonomy as a set of progressive indices that determine the ability of a system to perform a given task with or without human intervention, which they quantify as a measure of human independence. In this framework, neither explicit metrics nor mathematical models for their usage are defined.

Autonomy has been likened to intelligence and performance by some researchers, which is misleading. Usage of the label "misleading" is justified by the differences in the definitions of these terms, where intelligence is defined as the ability to acquire and/or use knowledge and performance as the ability to achieve a desired goal. Intelligence is an aspect of autonomous functioning that facilitates decision making and learning. Unlike autonomy, performance has no regard for consequences of the choices leading to the achievement of a goal. An example is the work presented in [26], in which autonomy is defined as a measure of performance with two measurement metrics namely, environmental complexity and information. The author agrees with this work on the point of environmental complexity being a key metric in determining autonomy and adapts the grid decomposition approach taken in determining local obstacle densities in the operating environment.

Another level of autonomy framework, but designed with military application in mind is the Autonomy Control Level (ACL) chart. It is based on four metrics namely, perception, analysis, decision making and capability [25]. These metrics were inspired by the observe–orient–decide–act loop. The ACL chart divides autonomy into eleven levels ranging from zero autonomy to full autonomy. A similar eleven level chart was proposed in [30], but based on mission complexity, task complexity and human independence groups of metrics. In chapter 4, a similar elven-level chart with level descriptions based on generic UAV control strategies, where level 0 corresponds to remotely controlled systems with no autonomy, level 10 to fully autonomous systems and levels $1-9$ to semi-autonomous systems is proposed.

The frameworks in [25], [28], [30] and seven others were evaluated on six UAVs in a case study presented in [6]. The study revealed that only three of the frameworks were able to classify all six vehicles, but only one of these was able to unambiguously classify all the six. Unfortunately, even the one that unambiguously classified the vehicles, classified two vehicles, one with some unsupervised capabilities and the other purely remote controlled in one class. This highlights the need to continue the quest for better autonomy evaluation frameworks.

Unlike all the previously presented approaches that focus on a discrete measure of autonomy in form of levels of autonomy, the work presented in [27] proposed a continuous measure of passive autonomy known as degree of autonomy. This is an important addition as the difference between autonomous systems is not only of kind (functionality), but also of degree (performance capacity). Their framework was based on the premise that autonomous systems have an automated problem-solving process, so they measured autonomy as the levels of automation of the problem-solving capability using two formulations presented in Eq.2.1 and Eq.2.2. Level of automation is a simple metric, but it does not account for the quality of decisions, which is a problem in complex systems. Other propositions of a continuous autonomy measure include [31], where autonomy is given by

Eq. 2.3 and [32], where autonomy is given by Eq. 2.4.

$$\alpha = \frac{\delta_D + \delta_E + \delta_S + \delta_I + \delta_V}{n} \tag{2.1}$$

where n is the number of autonomous capabilities considered and $\delta_D, \delta_E, \delta_S, \delta_I$ and δ_V are levels of automation for definition (D), exploration (E), selection (S), implementation (I) and verification (V) capabilities respectively.

$$\alpha = \frac{\frac{\eta_{D,act}}{\eta_{D,std}} * 10 + \frac{\eta_{E,act}}{\eta_{E,std}} * 10 + \frac{\eta_{S,act}}{\eta_{S,std}} * 10 + \frac{\eta_{I,act}}{\eta_{I,std}} * 10 + \frac{\eta_{V,act}}{\eta_{V,std}} * 10}{n} \tag{2.2}$$

where the η quotients are the ratios of actual system behaviour to standard behaviour for definition (D), exploration (E), selection (S), implementation (I) and verification (V) capabilities.

$$\alpha = \int_{T_i}^{T_f} \int_{V_i}^{V_f} \int_{PU}^{PL} (Human\_effort)\, dp\, da\, dt \tag{2.3}$$

where p, a and t are performance, area and time respectively.

$$\alpha = C_n \left(\frac{C_{bits}}{S_{msg}}\right)^{-i} \left(\frac{t_{cont}}{t_{miss}}\right)^{-j} \tag{2.4}$$

where C_{bits} is the number of control bits, S_{msg} is the total message size, t_{cont} is the contact time, t_{miss} is the total mission time, C_n, i and j are constants.

Unfortunately, all of the reviewed approaches either offer a categorical measure with poor output resolution or yield inconsistent metrics outputs. From this, it is clear that a set of autonomy metrics that is easily measurable, broad enough to capture autonomy evolution in a system and generates outputs with good output resolution is still lacking.

2.2 UAV Navigation

The navigation capability is supported by perception, localization, cognition and motion control competences. Additionally, environmental and mission complexity may necessitate inclusion of obstacle avoidance and mapping. UAV navigation has received wide spread interest since 2007 with the initial focus on outdoor applications [9], thanks to availability of GNSS services. Soon applications expanded to indoor and other GNSS-denied environments, which necessitated a paradigm shift in the field of navigation.

Starting with the Unmanned Ground Vehicle (UGV) class, for which GNSS-denied navigation techniques are more mature, attempts to apply these same navigation solutions directly to aerial navigation were limited by the fact that the former operates in a 2D world while the latter operates in a 3D world. Although, with the assumption of a quasi-fixed operating altitude, 2.5D navigation solutions can be developed, enabling operation of UAVs in a 3D world with only three controlled degrees of freedom (2D horizontal position and heading) as presented in [12, 33], which is comparable to the planar motion of UGVs.

Common to all aspects of aerial navigation is the reliance on state information, which could be available prior to mission, e.g. initial state, a path, landmark positions and a map, and/or acquired during mission execution, e.g. distance to approaching obstacles. Localization requires actual state observations and absolute or relative initial state information for absolute and relative localization respectively. Motion planning in a closed world requires only a map, whereas motion planning in an open world requires a map for global path generation and real-time local state observations for environmental uncertainty minimization. Motion control requires real-time state estimates to stabilize the vehicle.

Velocity, heading, altitude, position and attitude are not only the most important states, but also constitute a sufficient input set for navigation.

Autonomous navigation demands the interaction of all navigation components, but due to the sheer complexity of the navigation problem, researchers have focused on addressing individual components as indicated in Figure 2.1, with localization being the most studied field, followed by approach and landing. The downside to this divide-and-conquer approach is that once the individual navigation components are solved, formulating a full navigation solution by aggregation risks exceeding computation, power and hardware capability of any off-the-shelf deployment system. There have also been attempts to solve the full navigation problem, which totalled to 16% of the reviewed literature, showing that there is still a need for full GNSS-independent UAV navigation solutions.

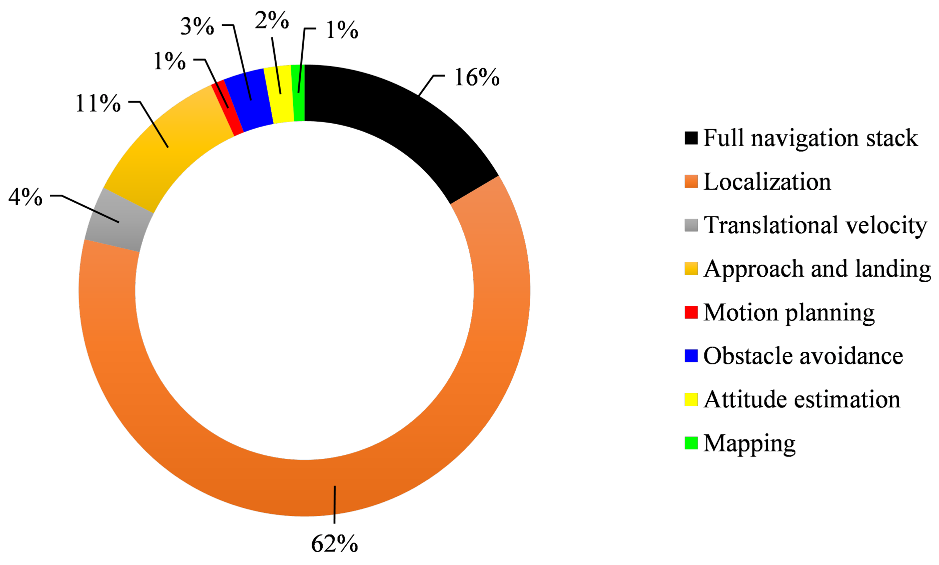


Figure 2.1: Distribution of GNSS-independent UAV navigation research.

2.2.1 Navigation Sensors

UAV navigation sensors for GNSS-denied environments can be categorized into attitude, localization, altitude, mapping and obstacle detection functional categories. Though, some sensors serve more than one category. These sensors can also be classified by technology, which includes inertial, vision, LiDAR, radio and acoustic. Another way to characterize sensors is as active or passive depending on whether they emit energy into the world or not. Again, some sensors are both passive and active like RGB-Depth sensors. Generally, passive sensors are lighter, cheaper and consume less power, but active sensors offer increased measurement accuracy, higher signal-to-noise ratio and increased robustness against environmental effects. Regardless, it is seldom that a single sensor performs satisfactorily over long periods, hence the need for sensor characterization with the effort to discover complementary sensors. One strategy for achieving this is to identify sensors that are superior at measuring some degrees of freedom, but inferior at others. Other criteria include:

- Error dynamics
- Reliability
- Robustness
- Computational complexity
- Exteroceptive or proprioceptive
- Relative or absoluteness
- Representational richness
- Passive or active

Sensor Selection

Selecting the right sensors is a daunting task given its criticality as it lays the foundation for the choice of algorithm, resource usage and performance. Sensor selection criteria include dynamic range, error characteristics, bandwidth, vehicle payload constrains, resolution, response time, operating environment, flight settings, computational resources and power availability. The most common UAV navigation sensors as of the preliminary review include GNSS receivers, inertial sensors, vision sensors, LiDAR sensors, acoustic sensors, radar sensors and Ultra-Wide Band (UWB) sensors.

For non-tactical civilian applications, inertial navigation systems employ small form factor strap-down Micro Electro-Mechanical Systems (MEMS)-Inertial Measurement Unit (IMU). A standard IMU consists of a triaxial rate gyroscope and a triaxial accelerometer. The key IMU features are dynamic range, gyroscope in-run bias stability, accelerometer in-run bias stability, gyroscope angular random walk and accelerometer velocity random walk. Inertial sensors are applied to localization, attitude, altitude and velocity estimation.

The key vision sensor selection features include resolution, shutter type, maximum frame rate, field-of-view and weight. Spatial deformation and blurring may occur in images as a result of camera motion induced by vehicle motion, vibration and moving objects. Rolling shutter cameras are more affected by this in comparison to global shutter cameras, but could be improved by rolling shutter modelling as demonstrated in [34]. Vision sensors from the preliminary review included monocular, stereo, depth, thermal and catadioptric cameras. Vision sensors have been applied for localization [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54], localization and mapping [33, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72], translational velocity estimation [73, 74, 75], landing pad detection [76, 77], obstacle detection [59, 78], attitude estimation [79, 47], and landing pad detection and landing [80, 81, 82, 83, 84, 76, 77].

LiDAR rangefinders have either a single fixed or a rotating beam. For the former, important is the range and accuracy. For the latter it is range, accuracy, scan angle, angular resolution and scan speed. LiDAR rangefinders have been applied for height estimation[37, 85], obstacle detection and mapping.

The commonly applied acoustic sensor is ultrasonic rangefinder, which is characterized by range, open angle, frequency and accuracy. Ultrasonic sensors operate on time-of-flight principle with range given as $0.5 \times C \times t$, where t is time of flight and C is the speed of sound in the transmission medium. Since C is affected by temperature, pressure and humidity, with temperature having the greatest influence of all [86], it is necessary to account for them to obtain accurate ranging.

Radar sensors have in the past been limited by their relatively large size, weight and power consumption to normal and large UAVs, but with recent technological breakthroughs, miniaturized radar modules fit for small and micro UAVs have started emerging. These sensors have the longest line-of-sight measurement range, are robust against changes in weather conditions, dust and lighting conditions. But relative to other sensor modalities, the current miniaturised modules are still large and heavy, for example the on-board radar dome in [87] has a volume of 0.280 m $\times$ 0.290 m $\times$ 0.285 m and mass 1.78 kg.

Generally, inertial sensors are proprioceptive, have high update rates of up to 1 kHz [88], score highly on Size, Weight and Power - Cost (SWaP-C) scale, but drift. Vision sensors are less effective for long-range perception, in low illumination and low textured environments [89], exhibit lower update rates, normally under 100 Hz [88], but capture diverse information, are lighter and relatively cheaper. LiDAR rangefinders are relatively expensive, heavier and exhibit degraded performance under bad weather conditions and direct sunlight [90], but deliver more accurate, direct and longer-range high frequency measurements. Radar sensors are relatively bulky and consume the most power, but deliver the longest-range and are insensitive to environmental conditions.

Motion Capture Systems

When sensing is not the focus, motion capture systems can track and deliver direct state information. They consist of synchronized active-passive optical sensors strategically positioned to accurately track objects within a pre-defined measurement volume. The three ways in which motion capture systems were used in the reviewed literature include localization, absolute position initialization and ground truth measuring for performance evaluation of state estimators . The downsides to these systems are high cost and limited measurement volume [57]. The different motion capture systems from literature are presented in Table 2.1, all of which are marker-based.

Table 2.1: Motion capture systems.

Motion capture system	**Usage in Literature**	**Papers**
Vicon	Performance evaluation	[33, 91]
OptiTrack	Performance evaluation	[36, 92]
	Localization and attitude estimation	[75, 50]
Qualisys	Performance evaluation	[93]
Motion analysis	Performance evaluation	[62]
	Position initialization	[62]
In-house system (five Kinect sensors network)	Localization	[65]

2.3 Localization

Localization answers the “where am I?” question by estimating the position of a vehicle. This capability is not only necessary for navigation, but also for object manipulation, multi-robot coordination, exploration and mapping. As revealed in Figure 2.1, it is the most researched navigation competence.

Robots are situated agents [94, 95]. This situatedness supports perception affordance for inertial, visual, radar, radio, LiDAR and acoustic-based localization. As presented in Table 2.2, each localization technique exhibits benefits and drawbacks. Drawing from complementarity, localization solutions combining multiple techniques have yielded better performance both in terms of accuracy and robustness.

Localization techniques require environmental representations, which could be made available beforehand in form of geo-referenced aerial images [39, 96], Computer-Aided Design (CAD) models [97, 98, 99], Digital Terrain Models (DTMs) [47], occupancy grid maps and feature maps [37, 100], or built and used simultaneously, as in Simultaneous Localization and Mapping (SLAM) [95].

Environmental representations could be generated simply by embedding artificial beacons at known positions in the environment [45, 101], which are then tracked by on-board sensors. The opposite is also possible, where beacons are mounted on the vehicle and sensors fixed in the environment [50, 65] as is the case with marker-based motion capture systems. This concept has been extended to multi-robot localization, where a robot with reliable localization capability provides a localization reference for other robots. An example is [102] where a vehicle in a GPS-denied environment localizes itself relative to GPS-enabled vehicles.

Aerial images are the most popular environmental representations, but finding image representations that are illumination invariant and tolerant to slight view-point changes is challenging. Proposed solutions to address the former include image gradient patterns [40] and learned features [39], and appearance-based matching methods [103] for the latter. For a summary of localization techniques see Table 2.3 and for the distribution of localization techniques among the reviewed full navigation solutions, see Figure 2.2.

Table 2.2: Pros and cons of the different localization technologies.

Mode	**Pros**	**Cons**
Inertial	• Very low SWaP (Size, Weight and Power) • Low cost • Very high update rate • Very short start-up time • Jam proof	• Drifts over time • Low signal-to-noise ratio at low accelerations and angular velocities • Overall accuracy dependent on the accuracy of the initial state
Vision	• Light weight • Semantic information • Jam proof	• Affected by weather and illumination • Indirect state observation • Computationally intensive
Radar	• Longest operating range • Robust against illumination and weather • Direct measurements • Range independent resolution	• Costly • Bulky • High power consumption • May require environmental accessories like transponders or reflectors
UWB	• Weather independent • Provides a communication channel • Low power consumption	• Interference from conductive materials • May require environmental installations • Limited operating volume and range • Low update rate circa 10 Hz
Magnetic	• No environmental installations required • Very low power consumption • Low cost	• Susceptible to magnetic interference • Requires delicate calibration • Low precision
LiDAR	• No environmental installations required • Very precise • Direct measurements • High update rate	• Very costly • Degraded by direct sunlight • Influenced by reflecting surface properties, like colour • Limited range, up to 100 m

2.3.1 Inertial Localization

UAV inertial localization operates on the principle of dead reckoning, where accelerations and/or velocities are integrated over time to obtain vehicle position and orientation with respect to an inertial reference frame. As mentioned in Sect. 2.2.1, inertial systems employ

Table 2.3: Practically tested localization solutions.

Localization	**Aiding Technology**	**Papers**
LiDAR SLAM	N/A (not available)	[33, 89, 92, 104, 105, 106, 107, 108]
Visual SLAM	N/A	[55, 11, 56, 57, 59, 21, 62, 64, 98, 97, 68, 69, 70, 71]
Visual Inertial Odometry (VIO)	N/A	[41, 60, 44, 48, 51, 63, 109, 66, 110, 67, 54, 72]
	GPS	[43]
Air-to-air cooperation	GPS and vision-based bearing	[102]
INS	Visual Odometry (VO)	[19, 111]
	Phased Array Radio System (PARS)	[112]
	Artificial markers	[35, 38, 42, 91, 52, 101]
	Machine learned vehicle model	[113]
	LiDAR scan matching	[12]
	UWB and vision	[43]
	Artificial Immune System (AIS)	[114]
Beacons	Visual beacons	[36, 53]
	Radar beacons	[87]
	UWB	[115]
Knowledge-based localization	Radar propagation channel response	[100]
	Satellite image	[39]
	CAD model	[99]
Radar odometry	N/A	[116]
Motion capture system	N/A	[50, 65]

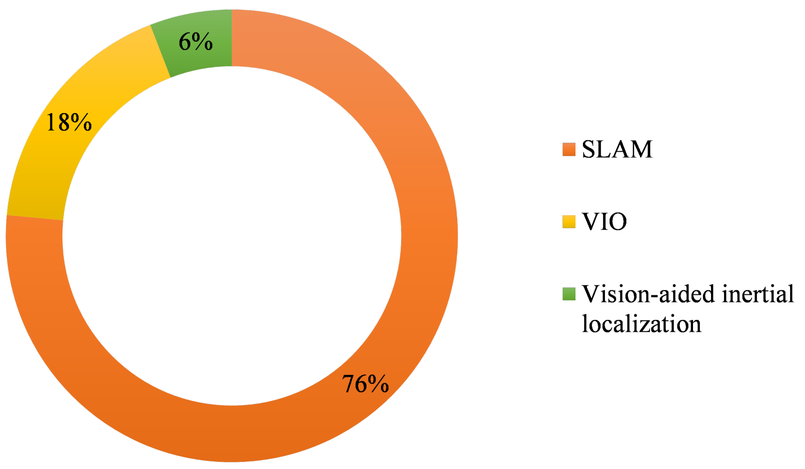


Figure 2.2: Distribution of localization techniques in reviewed full navigation solutions.

IMUs, which exhibit high update rates making them suitable for agile vehicles like UAVs. However, integration causes drift.

To address drift, it is customary to fuse inertial with other complementary techniques like vision, for example VIO as described in [88], visual odometry and optical flow, PARS [112], LiDAR [12] and UWB [117] through state estimators like Kalman or particle filters. Fusion could happen at raw data or state level, known as tightly coupled and loosely coupled schemes respectively. Disparate performance studies have showed tightly coupled schemes to be more accurate and robust than loosely coupled schemes [88, 118], but more complex and inflexible [116].

Results from the preliminary study accentuated the indispensability of inertial localization techniques in any modern GNSS-independent navigation solution, a conclusion drawn based on their presence in 100% of the reviewed full navigation solutions as indicated in Table 2.4. In all cases, these inertial techniques were aided by other techniques to minimize drift. It is also evident that vision is the most popular inertial aiding method with over 94% presence. From these observations, it is asserted that inertial-visual integration holds the key to GNSS-independent navigation solutions of the future.

2.3.2 Visual Localization

Visual localization utilizes vision sensors and computer vision algorithms to extract location information from visual scenes. Its popularity is attributed to the low SWaP-C of vision sensors, as well as their rich environmental representation, which includes shapes, colour and texture. However, vision algorithms are affected by illumination variations, low feature density, motion blurring and consume relatively high computational resources.

Vision methods are categorised into indirect aka feature-based and direct aka appearance-based methods [119, 120]. The former abstracts images to features making it faster, while the latter operates directly on pixel intensities, a trait that results in improved robustness against illumination variations. Of the two, feature-based methods are more popular given their lower computational cost, sparsity and robustness against rolling shutter artefacts [120], but exhibit degraded performance in featureless environments. It is also possible to combine indirect and direct methods resulting in semi-direct methods [121, 122].

Essential to indirect methods is the choice of features. Features either occur naturally in the environment or are specially designed and embedded into the environment to ease computer vision processing. Table 2.5 presents a summary of features identified in the reviewed literature and their associated properties.

To localize, correspondence must be established between features in consecutive frames, either by (1) detecting features in one frame and tracking them into a local region in the next frame or (2) detecting features in multiple images and matching them based on local similarity [123]. The former finds application in sparse optical flow, feature-based visual-SLAM and feature-based visual odometry, while the latter finds application in image matching as in establishing correspondence between a large-scale image, e.g. satellite images and a small-scale image for terrain-based localization as applied in [39]. Table 2.6 presents two sets of computational speed tests for Features from Accelerated Segment Test (FAST), Scale Invariant Feature Transform (SIFT), Speeded Up Robust Feature (SURF), Oriented FAST and Rotated BRIEF (ORB), shi-Tomasi, Harris corner detectors, both of which indicate FAST feature detector as the fastest.

As indicated in Figure 2.2, visual localization, specifically SLAM dominates among full navigation solutions. This dominance is attributed to its ability to simultaneously map and localize a vehicle in an unknown or partially known environment. SLAM relies on loop closure to reduce estimation drift [64, 60, 61], however, loop closing reduces on the available overall flight time. It should also be mentioned that SLAM initialization is a delicate step, which if not executed well can affect localization quality. SLAM initialization

Table 2.4: Full navigation solutions.

Paper	Perception	Environment	Localization	Path Planning	Obstacle Avoidance
[33]	Vision, laser rangefinders, IMU	Known indoor	2D SLAM	A*	Online local path planning
[35]	Vision, IMU	Known outdoor	Visual-aided INS	Ground control points	N/A
[55]	Vision, laser rangefinders, IMU	Unknown cluttered outdoor and indoor	2D SLAM	A*	Online local path planning
[11]	Vision, IMU	Unknown, unstructured	Visual SLAM	POMDP[a] motion planner	Ray casting
[59]	Vision, IMU, sonar	Indoor and outdoor	Visual SLAM	A*	Obstacle grid map
[41]	Vision, IMU	Structured known indoor	VIO	POMDP[a] motion planner	Online local trajectory generation
[92]	Vision, laser rangefinders, IMU	Indoor	2D SLAM	N/A	Online local path planning
[62]	Vision, sonar, IMU	Structured indoor	Visual SLAM	Dijkstra's algorithm	Online path re-planning
[105]	Vision, laser rangefinders, IMU	Indoor	2D SLAM	A*	Artificial potential fields
[66]	Vision, sonar, IMU	Unstructured, partially known indoor	Visual SLAM	BI-RRT[b]	Obstacle free global path
[106]	Laser rangefinders, IMU	Unstructured, unknown indoor and outdoor	LiDAR SLAM	N/A	Laser-based obstacle mapping
[69]	Vision and IMU	Complex indoor and outdoor	Visual SLAM	Dijkstra's algorithm	Obstacle free global path
[70]	Vision and IMU	Known indoor	Visual SLAM	Weighted lazy theta*	Online local path planning
[71]	Vision and IMU	Partly known indoor	Visual SLAM	A*	Online local path planning
[108]	Vision, laser rangefinders, IMU	Partly known indoor	LiDAR SLAM	A*	Obstacle free global path
[54]	Vision and IMU	Cluttered indoor and outdoor	VIO	Dijkstra's algorithm	Online local trajectory generation
[72]	Vision and IMU	Cluttered unknown indoor and outdoor	VIO	A*	Online local trajectory generation

[a] Partially Observable Markov Decision Process (POMDP).
[b] Belief and Information based-Rapidly-exploring Random Tree (BI-RRT).

methods found in literature include use of GPS [124], GPS/INS [21], pre-stored map [125], feature points [89] and manual flight initial mapping [81]. More detailed accounts of UAV visual localization techniques can be found in [126, 122].

2.3.3 LiDAR Localization

This navigation technique employs laser rangefinders, which actively illuminate the environment with a collimated light beam. Its measurement principle is based on either time-of-flight or phase-shift between the transmitted and reflected beam. These sensors are capable of measuring depth to a single point in the environment, a feature lacking in wide open angle sensors like ultrasonic. Single beam laser rangefinders have been applied for Above Ground Level (AGL) ranging in [37, 85]. With a rotating or rotating and nodding mechanism, they are capable of generating 2D and 3D localization point clouds respectively.

Unlike vision sensors, laser rangefinders provide direct ranging, but are unable to capture scene texture and colour, making them complementary to cameras that capture such information. This complementarity has been exploited in [33, 55, 92, 89, 104] for UAV localization.

2.3.4 Radio Localization

This section describes localization solutions supported by radio technology, which from the reviewed literature include radar, UWB and PARS.

Radar Localization

Until recently, radar size, weight and power constraints had limited its usage to normal and large aerial vehicles, but with recent technological breakthroughs, miniaturized radar modules fit for small UAVs are starting to emerge, and hence the associated radar localization solutions.

A demonstration of radar transponders deployed by a UAV at the onset of landing phase is presented in [87]. The transponders together with a multi-channel radar sensor constitute a wireless local positioning system. The position of on-board radar sensor is determined from range and elevation measurements to each of the transponders. Although the applied transponders are small in size (0.035 m $\times$ 0.015 m), the on-board dome sensor is quite bulky and heavy at 0.280 m $\times$ 0.290 m $\times$ 0.285 m and 1.78 kg respectively.

Biological inspiration is a driving factor in a number of robotic solutions, from anthropomorphic appearance of humanoid robots to evolutionary algorithms. Inspired by echo location in cave swiftlets, [100] used a Frequency Modulated Continuous Wave (FMCW) X-band radar transceiver mounted on a UAV to map propagation channel responses of the environment pre-flight. On revisiting the area, the vehicle localizes itself through determination of the closest match between the actual and stored channel responses. This localization technique was found to have high sensitivity to antenna orientation differences during reference channel response gathering and localization, making it lacking in robustness.

The proof-of-concept in [116] investigated radar-only 2D-localization of UAVs using radar odometry. An ultralight FMCW radar transceiver with one transmitter antenna and two receiver antennas mounted on a UAV measures azimuth and range to multiple static corner reflectors in the environment. An Ordered Statistics-Constant False Alarm Rate (OS-CFAR) detector is applied for target detection and Global Nearest Neighbour (GNN) algorithm for multiple corner reflector tracking. This technique has difficulty

Table 2.5: Visual features.

Features	**Properties**	**Paper**
FAST/ORB	• Computationally fast • Accurate • Rotation, scale and global illumination invariant • High repeatability	[62, 19, 124]
SIFT	• Scale and rotation invariant • More reliable local features • Computationally slow	[56, 61, 49]
SURF	• Scale and rotation invariant • Computationally slow (faster than SIFT)	[39, 60, 127, 48]
Shi-Tomasi	• Scale and rotation invariant • Based on Harris corners with better selection criteria	[42]
Adaptive and Generic Accelerated Segment Test (AGAST)	• Derived from FAST features • Computationally fast • Dynamically adapting corner detector	[69, 54]
Harris corners	• Rotation invariant • Intensity shift invariant • Intensity scaling invariant • Sensitive to image scaling	[63, 128, 67]
Solid circles	• Easily identifiable • Embed scale information	[36, 129]
Lines (edges or horizon)	• Easy to extract from noisy images • Orientation of lines may be estimated with sub-pixel accuracy • Simple mathematical representation	[12, 47]
Semantic features • Doors • Roads (intersection and centreline)	• Semantic understanding • Complicated mathematical definitions • Strong localization information	[42, 44, 130]
Learned features	• Not limited to human interpretation • Generalize well • Robust across heterogeneous images	[92]

Table 2.6: Feature detector speed tests.

	Time (msec)					
Papers	**FAST**	**ORB**	**SIFT**	**SURF**	**Harris Corners**	**Shi-Tomasi**
[66]	**2.8**	15.6	91.1	112.3	16.7	N/A
[131]	**4.1**	N/A	33.4	23.6	N/A	16

differentiating rotation-only from translation-only motion. Furthermore, despite designation as ultralight, the applied SENTIRE Radar transceiver is relatively heavy, with a mass of 0.186 kg.

Unlike [87] with deployable transponders and on-board multi-channel radar sensor, and [100] requiring prior mapping of the environmental channel response, [116] utilizes a single on-board transceiver with a single transmitting antenna and two receiving antennas. This design is more compact, but still necessitated availing a dedicated battery to cope with the high power demands of radar.

Ultra-Wideband and Phased-Array Radio Localization

Initially conceived for communication, UWB systems have been extended to incorporate ranging functionality [117]. Their precise timestamping and message scheduling supports Time Difference of Arrival (TDOA) and Two-Way Ranging (TWR) between tags and anchors [115], usable for localization. These modules are lightweight, consume relatively less power, but are more suitable for indoor open spaces as clutter leads to non-line-of-sight signal paths resulting in low accuracy.

For 3D position measurement, UWB requires precise installation of anchors in the environment limiting their usage to human accessible structured indoor environments. Important to note is the limited vertical clearance in indoor environments, a factor that results in poor height estimation accuracy relative to horizontal position accuracy. Additionally, UWB update rates scale inversely proportionally to the number of tracked tags.

The first reviewed work [115] demonstrated an indoor 3D UWB positioning system for UAV localization that achieved ± 0.20 m accuracy at 0.95 probability. The setup consisted of eight UWB nodes installed at different heights around the experimental volume. Each node ran a decaWave ScenSor DWM1000 transceiver module. The estimated position is represented as a pseudo-range message to emulate GNSS signals for ease of integration.

Fusion of UWB position measurement and inertial position estimate in an extended Kalman filter for UAV localization demonstrated in [117] achieved an accuracy of ± 0.15 m at 0.95 probability using only four ultra-lightweight UbiSense UWB anchors. UWB positioning together with visual QR-codes placed in the environment provided for inertial drift correction. This approach is highly inflexible since it requires not only installation of anchors at known locations in the environment, but also of QR-codes.

Following a strategy similar to [117], [132] loosely coupled inertial position estimates, UWB position measurements and 3D laser scanner position estimates generated at 100 Hz, 10 Hz and 1 Hz respectively in a Kalman filter. The algorithm assumed knowledge of the initial position and hovering at selected waypoints to enable the slow scanning process to run to completion. Despite intermittent motion, the proposed system achieved sub-centimetre accuracy when tested in simulation.

It is also possible to have UWB modules installed on mobile platforms as in [133] and [134] for air-to-ground cooperative localization. In [133], the UGV estimates its pose in the inertial frame using visual inertial odometry and shares it with the UAV via UWB communication channel. Then the UAV localizes itself in the inertial frame from its relative position to UGV and the shared UGV pose. In [134], the UGV localizes itself via Differential Global Positioning System (DGPS) while the UAV localizes itself from its relative position to the UGV tracked by UWB. The UGV is assumed to operate in a GNSS-enabled environment, while the UAV in a GNSS-denied environment.

PARS uses electronically steered radio waves for detection and ranging. [112] applied PARS and achieved GNSS-level positioning accuracy (6.86 m RMSE). However, unlike GNSS, PARS has a higher signal-to-noise ratio and supports encrypted communication. PARS consists of an on-board module and a ground module at a known location, in this case Radionor CRE2-189 and a 0.295 kg CRE2-144-M2-SMA module respectively. Phase differences of the received signals by the ground receiver are used to resolve azimuth, elevation and position of the vehicle. The authors assert that this aerial-ground radio combination can support up to 114 km line-of-sight ranging.

2.3.5 Above Ground Level Estimation

This degree of freedom differentiates aerial robots from ground robots. AGL measurements support not only take-off, cruise and landing flight phases, but also scale factor determination in monocular vision as in [19, 56, 57] and optical flow algorithms.

Techniques for AGL estimation include altimeters, computer vision (stereo vision and optical flow), inertial odometry and beacons. Altimeters are the most common, supported by sensors like ultrasonic, laser rangefinders, barometers and radar. Less common are inertial and computer vision, supported by IMU, and monocular, stereo-vision and RGB-Depth cameras respectively. Inertial sensors double-integrate vertical IMU acceleration to obtain AGL, but this approach is susceptible to drifting. Noteworthy, laser rangefinders, ultrasonic, radar and vision sensors may generate false AGL in presence of ground obstacles due to their vehicle-centric reference, but barometers, motion capture systems and beacons are immune to this effect, an achievement attributed to their global reference nature.

Researchers have also fused complementary sensors to improve AGL estimation accuracy, robustness and reliability. Examples include IMU and barometer fusion in a complementary filter [135], IMU, 3D laser scanner and UWB fusion [132], IMU and barometer fusion [115], visual odometry fused with UAV and satellite image alignment using a deep convolutional network [39], IMU, ultrasonic and barometer fusion [101], fusion of IMU and visual motion [61], fusion of INS and horizon profile matching [47], and fusion of IMU, barometer and laser rangefinder in an Extended Kalman Filter (EKF) [92].

2.3.6 Operating Environments

Environments can be classified as structured/unstructured, known/unknown/partially known, indoor/outdoor, 1D/2D/3D and static/dynamic. Each class offers different localization affordance. Structured environments may offer artificial markers, indoor environments may offer perspective lines, outdoor environments may offer rich texture, stars etc. Localization requires environment-to-map association through sensors, which should be capable of perceiving either continuous or discrete environmental factors of variation correlated with the position of the vehicle. Unfortunately, in an open world, environmental properties may vary over time, which adds to environmental complexity.

Environmental complexity could be a factor of obstacle density, obstacle dynamics, illumination intensity, spatial symmetry, signal attenuation, turbulence, structure, knowledge of the environment and relative size of the free configuration space. Analysis of this complexity is a key input to the aerial platform and sensor selection process, and is one of the three factors, the others being mission complexity and human independence, that determine the level of autonomy of unmanned systems according to NIST [136]. Therefore, by the law of requisite variety, vehicle capability should match environmental complexity. For indoor, forested and urban canyons, multi-rotor vehicles are a more suitable choice because of their Vertical Take-Off and Landing (VTOL) capability and agility, while for open outdoor and high-altitude flights, fixed-wing vehicles are better suited.

GNSS-denied environments explored in the reviewed literature include indoor corridors [137, 42], underground mines [109], urban canyons [18, 134, 106], forests [85, 55] and outdoors [43]. This work focuses on indoor environments and urban canyons, which have similar properties, i.e., both are bounded by walls and exhibit challenged or completely denied GNSS reception.

2.4 Motion Planning

Situatedness implies that robots are an intrinsic part of their operating environment. This obliges them to respect environmental geometric and kinematic constraints. This compliance is achieved by either manually programming robot task sequences or automatically generating task sequences compliant with such constraints. The former approach is tedious and time consuming, especially in high dimensional spaces, which motivated the latter approach known as motion planning.

Motion planning consists of path and trajectory planning, where the former generates an obstacle free path from a start to a goal configuration with no regard for vehicle dynamical constraints. The latter parametrizes a purely geometric path with time promoting it to a trajectory and enables consideration of differential and torque constraints on the generated path [138]. Since path planning and time scaling of the resulting path are performed sequentially, the resulting trajectory is typically not time-optimal. A time-optimal trajectory can be generated by planning directly in the state space [139].

Path planning may assume precise knowledge of all obstacles in the environment, which assumption is unrealistic when operating in an open world. Path planners founded on such an assumption are known as offline planners. Contrary to these are online planners, which account for incomplete obstacle information.

Paths are searched within partially or fully known workspaces, which is a union of a closed set of obstacle configurations and an open set of free configurations. Configurations in the obstacle set involve collisions that violate geometric constraints, except in cases where such contact is part of the task, e.g. grasping or landing on an object, in which case the reachability structure becomes the closure of the free configuration. To simplify path planning, a robot geometry is equated to a particle. This simplification ignores geometric constraints, which are then incorporated through expansion of all obstacle boundaries by the radius of the robot silhouette. The resulting space with expanded obstacle boundaries is known as configuration space [140]. Paths can be easily planned in this space using one of several path planning paradigms.

Path planning paradigms include sampling-based, combinatorial, potential field [141] and reward-based planners. Sampling-based path planners probabilistically or deterministically sample the configuration space to build a data structure that captures the connectivity of the free configuration space. Deterministic sampling results in a regular data structure, which is resolution complete, while probabilistic sampling results in a random data structure, which is probabilistically complete [138]. Sampling path planners belong to either the Probabilistic Roadmap Method (PRM) multi-query subclass or the Rapidly-exploring Random Tree (RRT) single-query subclass [66]. Combinatorial methods construct a roadmap from intersections of Voronoi roadmap segments. Potential field methods build a mapping from configuration to a potential, known as a potential function. Then techniques like gradient descent can be applied to this function, to guide a system downhill to a goal located at the global minimum. Reward-based planners select actions from a finite set of possible actions to maximize a future reward. As a result of the finite set of possible actions, this approach is suboptimal. A summary of differences between the three classical path planning paradigms namely, combinatorial, sampling-based and potential fields is presented in Table 2.7.

The quest for an optimal and efficient universal path planner has for decades persisted with no solution in sight, i.e., no one path planning paradigm outperforms the others at all planning scenarios [142], but among the four, only combinatorial methods exhibit completeness—complete planners exit in finite time if no path exists. Potential field methods are suitable for online path planning and provide measures for feedback control, but their gradient descent implementations are susceptible to local minima, leading to goal unreachability [53]. Unlike combinatorial and potential field, sampling-based methods apply to a wide range of path planning problems, thanks to their ability to scale to higher dimensional spaces and large problems with tractable computational costs [71, 72], but generally produce non-smooth paths. If time is available, path simplification may be applied to smoothen the path.

Table 2.7: Differences between sampling, combinatorial and potential field path planning paradigms.

Sampling	**Combinatorial**	**Potential fields**
Builds a roadmap or tree	Builds a roadmap	Potential field function
No modelling of the obstacle configuration space	Models the obstacle configuration space	Models the obstacle configuration space
Probabilistically or resolution complete	Generally complete	No solution guarantees[a]
Applicable to higher dimensional configurations	Limited to lower dimensional configurations	Limited to lower dimensional configurations[b]
Online variants exist	Virtually always offline	Online capable
Complexity is independent of environmental complexity [143]	Complexity scales exponentially with problem dimensionality	Complexity scales exponentially with problem dimensionality

[a] Potential field-based methods guarantee no solution except in case of a navigation function or randomized potential planners. Despite completeness, both have a downside of requiring prior complete knowledge of the configuration space [144].
[b] With random walk augmentation as a way to overcome local minimum, this approach was able to plan in up to a 31 dimension configuration space [138].

Looking at Figure 2.3, Combinatorial methods dominate among global path planners, thanks to their completeness, while sampling-based global path planners are the least applied of the three despite their growing attractiveness with properties like relatively fast planning, computational complexity independent of environmental complexity and the ability to handle nonholonomic constraints and high dimensional configuration spaces. Furthermore, researchers have also developed real-time and asymptotically optimal randomized sampling-based planners like real-time capable RRT-Connect [145] and asymptotically optimal Rapidly-exploring Random Graph (RRG), PRM* and RRT* [146]. In Sect. 5.1, it is showed that despite non-optimality, the sampled paths are practical for building-like environments, online implementable and are of acceptable quality if combined with path smoothing post-processing. Additionally, a combinatorial method has been applied for addressing the coverage path planning problem in Sect. 5.2.

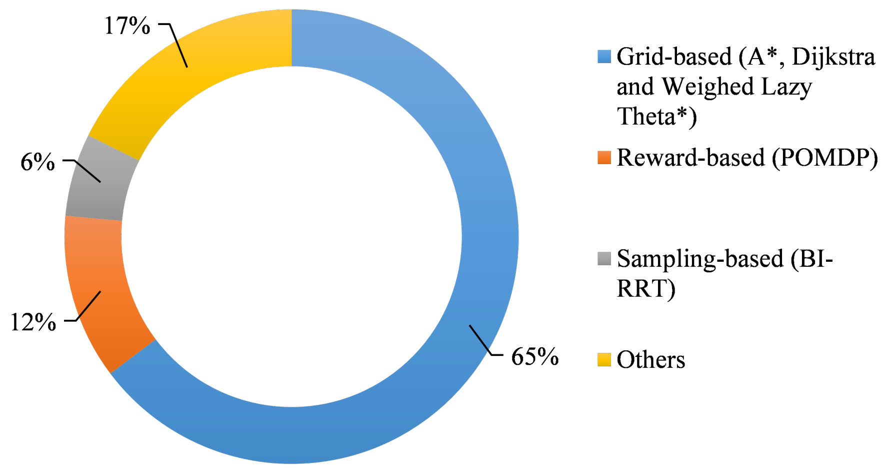


Figure 2.3: Global Path Planners

2.5 Integrity Monitoring of UAV Navigation Systems

UAV mishaps may result in costly property damage and loss of lives especially when operating in urban areas. Failure analysis conducted by United States military on six normal-sized UAVs over a period of 17 years revealed that 57% of the failures were attributed to system design, i.e., propulsion and flight control systems [147]. Navigation, which is the focus of this work, subsumes both propulsion and flight control systems making it a very critical part of any UAV flight management system.

Navigation is supported by sensors, actuators, algorithms and the environment all of which are fault-prone. To ensure reliability, it is imperative to detect and eliminate faults before they compromise the system. In remotely piloted systems, human pilots may easily detect faults and avert impending failure. In autonomous systems, faults are either observed directly through sensors or indirectly through integrity monitoring systems.

Integrity monitors can be classified into model-based, signal processing-based, and knowledge-based approaches [148]. Model-based approaches derive and incorporate equations of motion into either state observers, e.g. Luenberger observer in [4] or state estimators like Kalman filters, e.g. in [149]. Signal processing-based approaches transform signals into representations for which the normal and faulty signals can be separated. Knowledge-based approaches derive a set of rules from expert knowledge capable of distinguishing known faulty from normal states.

A distribution is normally assumed on each state with normality defined to within a standard deviation threshold and any state outside this threshold is considered faulty. Mahalanobis distance function is commonly applied for this check. The threshold value varies depending on the criticality of the system, task and environment. Therefore, different tasks and environments demand different Requirements of Navigation Performance (RNP). A set of metrics including Protection Level (PL), Alert Limit (AL), Time-to-Alert (TTA) and Integrity Risk (IR) has been adopted for integrity monitoring, but the thresholds are not standardized as of this literature survey.

Faults could be detected at sensor or state levels. The former associates each sensor with a fault detector, while the latter fuses sensor measurements together in an estimator and analyses the estimated state for faults. An example of the former is presented in [4], which is a model-based fault detector for an autonomous helicopter. For each sensor and its associated Luenberger observer, a residual is calculated and analysed for any anomalies. The results indicated successful detection of different faults, but also revealed an inverse relationship between drift magnitude and drift fault detectability.

It is also possible to combine mathematical models with data-driven models for fault detection. The work presented in [149] detected faults in a triaxial gyroscope mounted on a quadcopter using a neural network whose weights were updated by an Extended Kalman Filter (EKF). The algorithm successfully detected bias, step and triangular faults.

Others have combined model and knowledge-based approaches for fault detection, like the Adaptive Neuron Fuzzy Inference System (ANFIS) for position related fault detection in a UAV navigation system presented in [150]. The algorithm determines an F-indicator from R-Indicator and C-Indicator, which are functions of Kalman filter residual, and applies it to fuzzy rules for fault detection. The training dataset is continuously updated with every successful detection of a normal or a faulty state. Upon addition of a predefined number of new faulty samples, the neural network is retrained, which then updates the fuzzy rules. The approach is beneficial in terms of learning new rules to detect new faults, a feature that is missing in knowledge-based fault detectors.

Unlike the data-driven models in [149] and [150] that require training, the approach proposed in [151] applies a non-parametric training-free data-driven approach that compares the current input to the previous normal inputs within a window of size n to determine if the current input is faulty. Before analysis, the input is filtered for noise reduction then applied to a Mahalanobis distance function to decide if it falls within the standard deviation threshold. This non-parametric, model free, unsupervised, online approach is applicable to different domains and data sources, and has been tested on a UAV, vacuum cleaning robot and electric power supply for fault detection.

In some cases, it is necessary not only to detect but also to eliminate faults or minimise their effect. Fail-safe through hardware and software redundancy is demonstrated in [152]. Specifically, they demonstrated detection and recovery from actuator malfunction on a coaxial octocopter. The two rotors per arm allow for continued performance in case one of them fails. Rotor failure is observed indirectly through Euler angles measured by an IMU. The detector analyses residuals—discrepancy between sensor and non-linear sliding mode observer. Then the occurrence of a rotor failure is decided by a rule-based inference algorithm. Besides actuator redundancy, software redundancy is also implemented, where a different control law is executed during system recovery to account for changes in total thrust.

Each navigation technique requires a dedicated integrity monitor as these methods exhibit unique dynamics and hence failure modes. Of interest are vision-based navigation systems. As mentioned in Sect. 2.3.1, vision sensors have become an indispensable part of GNSS-denied navigation systems, but unlike GNSS, which exhibit fewer errors, in most cases one per component, visual navigation systems exhibit numerous simultaneous error sources. This complicates the fault identification process. The authors of [153] proposed an integrity monitoring framework for a feature-based (ORB-SLAM2) visual localization system. The framework applies an Iterative Parity Space Outlier Rejection (IPSOR) method that evaluates the distribution of weighted sum of squared residuals to determine whether an outlier measurement is present. This method relies on the assumption that inliers have a high precision. The iterative nature allows for elimination of multiple outliers, but not all outliers. It is assumed that at most a fault will ensue from the undetected outliers. After outlier rejection, protection levels in x, y and z-directions are calculated and used to determine system integrity.

2.5.1 Fault Modes in Navigation Systems

Navigation faults exhibit varying dynamic behaviours like ramp, bias, periodic, step and sporadic. The five commonly tracked faults and their likely causes are presented in Table 2.8, which are classified into point, contextual and collective faults [154]. For point faults, a single value is sufficient to rule on occurrence of a fault, while for a contextual fault, context and value are required. Ruling on collective faults require analysis of a sequence of values.

Table 2.8: Fault types and their likely causes (Adapted from [3, 4]).

Fault Classification		**Fault type**	**Example Causes**
Hard failure	Point fault	Complete failure	Power disruption or communication blackout
Soft failure	Contextual fault	Bias fault	Current or voltage bias due to temperature change and or vibrations
	Collective fault	Drift fault	Internal temperature change or outdated calibration values
	Collective fault	Multiplicative fault	Environmental changes, component ageing
	Collective fault	Periodic fault	Environmental changes
	Collective fault	Outlier fault	Environmental changes

2.5.2 Existing Challenges of Integrity Monitoring Systems

Despite decades of research, there are still a number of challenges that need addressing. Following is a list of open challenges, this list is not in any way comprehensive:

- Multiple fault detection. Most solutions in literature assume a single fault yet the complexity of navigation systems suggests a likelihood of multiple faults.

- Performance requirements. At the time of compiling this literature survey, there were no standardized performance requirements for integrity monitoring in UAV navigation systems.

- Faults elimination techniques. With the exception of obvious cases like outlier rejection and simple hardware redundancy, fault elimination techniques are still lacking.

- Predictive fault monitoring. The existing fault detectors respond to occurrence of a fault, which in some cases might be too late for any correction measures. Therefore, raising a need for predictive fault detection techniques.

- Computational complexity. Integration of integrity monitors into flight management systems may introduce competition for resources that may jeopardize system stability. Hence, the need for low computational complexity integrity monitoring-UAV navigation software architectures.

2.6 Technology Readiness Assessment of UAV Navigation Systems

This section uses a Technology Readiness Level (TRL) framework adapted by European Space Agency (ESA) from the original TRLs developed by National Aeronautics and Space Administration (NASA) and Unmanned Aerial System (UAS)-adapted Autonomy and Technology Readiness Assessment (ATRA) [28] to determine the maturity of GNSS-independent navigation solutions. This assessment serves as a progress evaluation step and a justification for efforts so far invested in addressing this navigation problem. The TRL technology readiness assessment framework consists of nine readiness levels [155].

Motivated by ATRA framework, the nine TRLs are divided into three clusters namely, research and development (levels 1-3), integration, testing and evaluation (levels 4-7), and production and deployment (levels 8-9). Levels 1-3 analyse the feasibility of the underlying theoretical premise. All the reviewed research met this feasibility check. Levels 4-7 test and evaluate navigation technology against environmental complexity and performance metrics. Levels 8-9 look at deployment of the actual complete systems in real scenarios. Table 2.9 presents a summary of TRLs ratings for all the full navigation solutions found in the reviewed literature, from which it is evident that the GNSS-independent navigation technology at the time of this survey is at TRL-6, interpreted as “Successful high fidelity prototype demonstrations in relevant environments” [28]. This observation justifies the need for more effort towards achieving levels 7 through 9 navigation solutions.

2.7 Future Trends in the Field of GNSS-Independent UAV Navigation

As revealed herein, navigation has not reached full technology maturity. This section highlights the likely future trends in sensing technology, algorithms, communication technology and simulation.

With the prevalence of off-the-shelf autopilots, it is asserted that UAV flight controllers and their associated structural designs have reached full technology maturity. So, the focus is now concentrated on perception, localization, high fidelity physics simulations, fast online motion planning and replanning, and communication.

Towards perception, future research is expected to employ more advanced sensors like signal-of-opportunity sensors, event cameras and miniaturized radar. With advanced sensors also comes the need for advanced algorithms. Active vision, machine learning techniques especially physics-informed neural networks and other adaptive Artificial Intelligence (AI) techniques are expected to become popular solutions for addressing sensor error modelling and semantic scene understanding. Last but not least, 5G communication technology is expected to become a key enabler for on-board data-driven navigation systems with its high volume real-time data streaming capability.

Going beyond TRL-6 demands deployment and testing of the proposed navigation solutions in their expected operating environments, a step that comes with great risks that might ensue from mishaps. Therefore, it is imperative to test out these systems on high fidelity simulators before deployment. Hence, the likely prevalence of high-fidelity UAV navigation physics simulators featuring more physics-accurate vehicles and environment behaviours. A solution to this has been presented in Chapter 7 of this writing.

Looking at GNSS-based navigation systems that support launch-and-forget, GNSS-independent navigation solutions are soon to move towards such integrity, availability and continuity levels. Of course, this will be preceded by clear definition and measures of autonomy, and standardization of performance requirements for GNSS-independent UAV

Table 2.9: TRL rating of the full navigation solutions from the reviewed literature.

Paper	**Perception**	**Solution Description**	**TRL**
[33]	Laser rangefinder, IMU and vision	On-board 2D SLAM, tested on a prototype in a partially known relevant environment	6
[35]	Vision, IMU	Visual-aided INS, tested in a simulated known environment (Software-in-the-loop)	5
[55]	Laser rangefinder, IMU and vision	On-board LiDAR 2D SLAM, tested on a prototype in an unknown cluttered relevant environment	6
[11]	Vision, IMU	3D visual SLAM, tested in a simulated unknown unstructured environment	5
[59]	Vision, IMU, sonar	On-board 3D visual SLAM, tested on a prototype in relevant environment	6
[41]	Vision, IMU	Off-board VIO, tested on a prototype in a known structured relevant environment	5
[92]	Laser rangefinder, IMU and vision	On-board 2D SLAM, tested on a prototype in a relevant environment	6
[62]	Vision, sonar, IMU	On-board 3D visual SLAM, tested on a prototype in a known structured relevant environment	6
[105]	Laser rangefinder, IMU and vision	On-board 3D SLAM, tested on a prototype in a relevant environment	6
[66]	Vision, sonar, IMU	Off-board 3D visual SLAM, tested on a prototype in a partially known unstructured environment	6
[106]	Laser rangefinder, IMU	Off-board 2D LiDAR SLAM, tested on a prototype in an unknown unstructured relevant environment	6
[69]	Vision and IMU	On-board 3D visual SLAM, tested on a prototype in a relevant environment	6
[70]	Vision and IMU	On-board 3D visual SLAM, tested on a prototype in a known relevant environment	6
[71]	Vision and IMU	On-board 3D visual SLAM, tested on a prototype in a partially known relevant environment	6
[108]	Laser rangefinder, IMU and vision	On-board 3D LiDAR SLAM, tested on a prototype in a partially known relevant environment	6
[54]	Vision and IMU	On-board 3D visual SLAM, tested on a prototype in a cluttered relevant environment	6
[72]	Vision and IMU	On-board 3D visual SLAM, tested on a prototype in a cluttered unknown relevant environment	6

navigation, which did not exist at the time of compiling this literature survey. The former has been addressed in Chapter 4 of this writing.

As indicated in Figure 2.3, sampling-based path planners were the least applied despite their growing attractiveness, with the development of real-time and asymptotically optimal randomized sampling-based planners like real-time capable RRT-Connect [145] and asymptotically optimal RRG, PRM* and RRT* [146]. This attractiveness suggests a likely prevalence in their application. In Sect. 5.1, randomized sampling path planners are parallelized in an ensemble path planner resulting in a high success rate and fast planner capable of supporting real-time global path planning and replanning.

The future trends presented in this section formed the basis for derivation of some of the objectives addressed in this work. These objectives have been addressed through several projects, whose results are presented in the subsequent chapters herein.

2.8 Chapter Summary

This chapter presents the existing scientific contributions related to the problems at hand. It starts by looking for answers to the question of "How to assess the autonomy of a system?", for which two kinds of solutions have been proposed over the years namely, an ordinal scale measure that assigns a level index to each autonomous system, and a ratio scale measure that combines quantitative metrics into a single value measure of autonomy. Then talks about existing navigation sensor technologies, which include GNSS, inertial, vision, LiDAR, radio and acoustic, and their associated benefits and drawbacks. Then reviews motion capture systems, commonly used as substitutes for onboard perception or for performance evaluation of navigation solutions. These systems are characterized by high accuracy and high frequency state information.

The GNSS-independent UAV navigation problem has been addressed to varying degrees. Of all reviewed works, full navigation solutions constituted only 16%, with majority focusing on localization at 62% followed by approach and landing, translational velocity estimation, obstacle avoidance, attitude estimation, mapping and motion planning at 11%, 4%, 3%, 2%, 1% and 1% respectively. Amongst the reviewed full navigation solutions, SLAM-based solutions dominated at 76% followed by visual inertial odometry and vision-aided inertial solutions at 18% and 6% respectively. SLAM dominates the localization techniques, followed by visual inertial odometry and visual-aided inertial localization. This reveals optical and inertial as the dominant technologies. With the reliability and high update rates of inertial systems, and rich representation power of vision systems, it is conjectured that inertial-optical hybridizations hold the key to high performance GNSS-independent UAV autonomous navigation solutions.

The last two sections look at integrity monitoring and technology readiness of GNSS-independent UAV navigation solutions, where the former aims at detection and elimination of faults through application of either model-based, signal processing-based or knowledge-based approaches for fault detection, and software and/or hardware redundancy for fault recovery. The latter applies TRLs for technology maturity assessment of GNSS-independent UAV autonomous navigation solutions with a result of TRL-6, interpreted as "Successful high fidelity prototype demonstrations in relevant environments".

Chapter 3

Unmanned Aerial Vehicles

UAVs (Unmanned Aerial Vehicles) have generated widespread interest in academia, industry and military due to their applicability to remote sensing, logistics, inspection, search and rescue to mention but a few. More applications can be found in dedicated articles like [7, 8]. This chapter covers classification of UAVs in Sect. 3.1 , vehicle selection in Sect. 3.2 and mathematical modelling of quadrotor-type UAVs in Sect. 3.3. The former two facilitate UAV regulation and vehicle selection, while the latter describes the equations governing the motion of quadrotor Micro Aerial Vehicles (MAVs), which are applied later on in Chapter 7 for vehicle simulation and analysis. With proper UAV classification schemes in place, the task of matching appropriate vehicle performance to task requirements is made simpler. Furthermore, this classification also serves as a basis for establishment of regulations governing UAV usage, which is evident as national and international regulations target not a single vehicle, but groups thereof. As the title of this thesis suggests, the vehicle class of interest is MAV. The next section describes the relations between MAV class and other UAV classes.

3.1 UAV Classification

This sub-section introduces a two-stage UAV classification process. The first stage classifies UAVs based on their aerodynamical configurations, then the second stage classifies the different aerodynamical classes based on Maximum Take-Off Weight (MTOW). The different aerodynamical configurations include blimps, flapping-wing, fixed-wing, rotorcrafts and ducted-fan vehicles, and combinations thereof constructible known as hybrid vehicles.

Blimps overcome gravity by regulating the density and pressure of the gas filling their hull making them lighter than ambient air. They are capable of very long endurances, but are less manoeuvrable and bulky [156]. Fixed-wing vehicles regulate their speed and the shape of their aerofoil wings to generate lift. They are capable of long endurance and high cruise speeds, but require continuous forward motion to stay airborne [157]. Rotorcrafts overcome gravity by regulating the rotational speed of their propeller blades, resulting in a lift force along the rotation axis in the direction established according to the right-hand convention. This locomotion modality enables them to achieve Vertical Take-Off and Landing (VTOL), hovering and high manoeuvrability, but consume more power. Flapping-wing vehicles mimic birds and are normally characterised by very lightweight chassis supported by relatively large wings. As mentioned in the title, this thesis focuses on GNSS-denied environments, which are normally characterized by partially or fully enclosed spaces with obstacles. This makes rotorcrafts the best choice given their manoeuvrability, hovering and VTOL flight capability, hence, the vehicle of choice for this work.

The existing MTOW classification schemes as presented in [158] only support micro, mini, small, lightweight, normal and large vehicle classes. But the recent surge in interest for nano vehicles both for civilian and military applications raises the need for their inclusion as an independent class separate from micro vehicles. Equipped with this motivation, a new MTOW classification scheme presented in Table 3.1, which is an adaptation of the classification schemes from [159], [160] and [161] has been proposed. The weight range for the nano class has been inspired by the existing nano UAVs from the preliminary literature review.

Table 3.1: UAV classification.

Category	**Maximum Take-off Weight (kg)**
Nano	≤ 0.25
Micro	(0.25, 5.0]
Mini	(5.0, 30]
Small	(30, 100]
Lightweight	(100, 250]
Normal	(250, 5000]
Large	> 5000

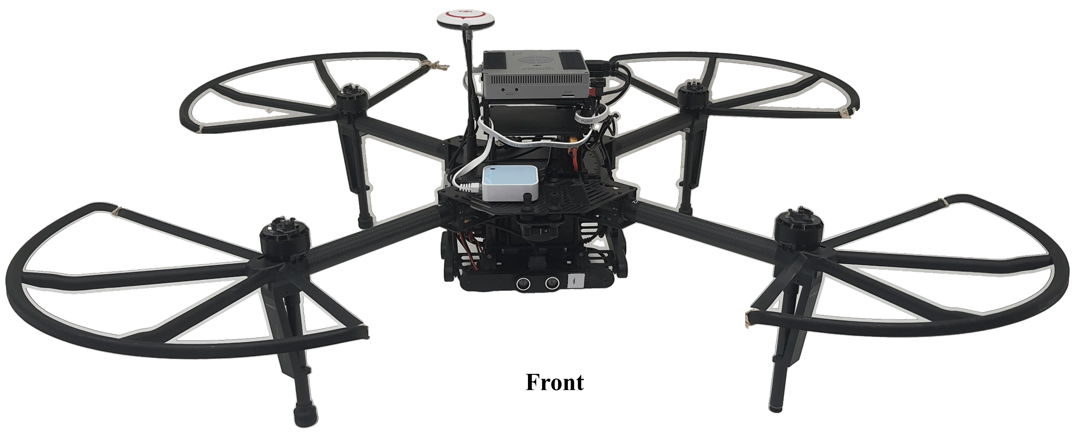
3.2 Vehicle Selection

A vehicle is an integral part of the environment in which it operates. Equipped with the knowledge of the intended operating environment, i.e., enclosed with obstacles, rotorcrafts made the best choice. In particular, quadrotor vehicles were selected. These vehicles are characterised by low inertia and simplistic structural design. Quadcopter frames come in two types namely, $\times$-configuration and $+$-configuration. The difference between the two is that the former engages all four rotors during pure roll and pure pitch manoeuvres, while the latter engages a different pair in each case. This puts the $\times$-configuration at an advantage of increased agility. Therefore, for all the projects conducted in this research work, two micro quadcopters of $\times$-configuration have been applied.

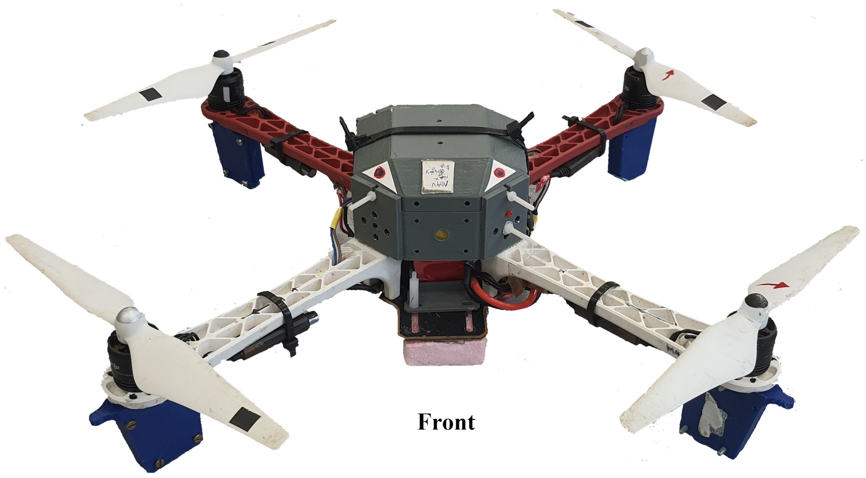
The two vehicles included an off-the-shelf DJI Matrice 100 developed by SZ DJI Technology Co., Ltd in Fig. 3.1a and a custom built quadcopter based on F450 chassis in Fig. 3.1b, weighing $2.4 - 3.4$ kg and $1.0 - 1.5$ kg respectively. The actual weights depend on the payload, which varied depending on the project. DJI Matrice 100 is a developer platform, supporting custom built navigation applications to interface with the inbuilt autopilot. F450 comes with a chassis and propulsion system, but no sensors and autopilot. This allowed for custom integration of sensors and autopilot. More details on the two vehicles can be found in Appendix A. Now that we know the configuration, class and sizes of UAVs used in this research, the next section builds on this by discussing the equations governing the motion of these types of vehicles.

3.3 Quadcopter Dynamics and Kinematics

This section deals with derivation of the equations that define the motion of a quadcopter. These equations are divided into three categories namely, dynamic translational, dynamic rotational and kinematic equations. Details for each of the three categories are described in the subsequent sub-sections.



(a)



(b)

Figure 3.1: MAVs applied in the different research projects presented herein.

3.3.1 Dynamic Motion Model

The equations presented here describe the motion of a quadrotor-type rigid-body in relation to the forces causing this motion. The equations describing the translational and rotational motion of a generic rigid-body are derived from translational and rotational variants of Newton's second law of motion respectively, restricting their validity to an inertial frame of reference. The inertial frame adopted is the North-East-Down (NED) frame, fixed on the surface of the earth with the assumption that the earth's acceleration is relatively negligible. Besides properties like gravity and absolute position that are measured in the inertial frame of reference, other states like acceleration and rotational speeds are measured by strapdown IMU sensors in the body frame of reference with its origin coincident with the centre of mass of the vehicle. The inertial and body frames are indicated as $\{I\}$ and $\{B\}$ in Fig. 3.2 respectively. Since the inertial frame does not rotate, it is evident that the relative orientation of the two frames changes during operation, raising the need to transform states from one frame of reference to another, which is achieved through a Direction Cosine Matrix (DCM) ${}^{to}\underline{R}_{from}$. Together this information yields the translational equation of motion, which is presented in Eq. 3.1,

$$\ddot{\underline{p}} = \frac{{}^{I}\underline{R}_{B}\underline{f}_{B}}{m} + \underline{g} \tag{3.1}$$

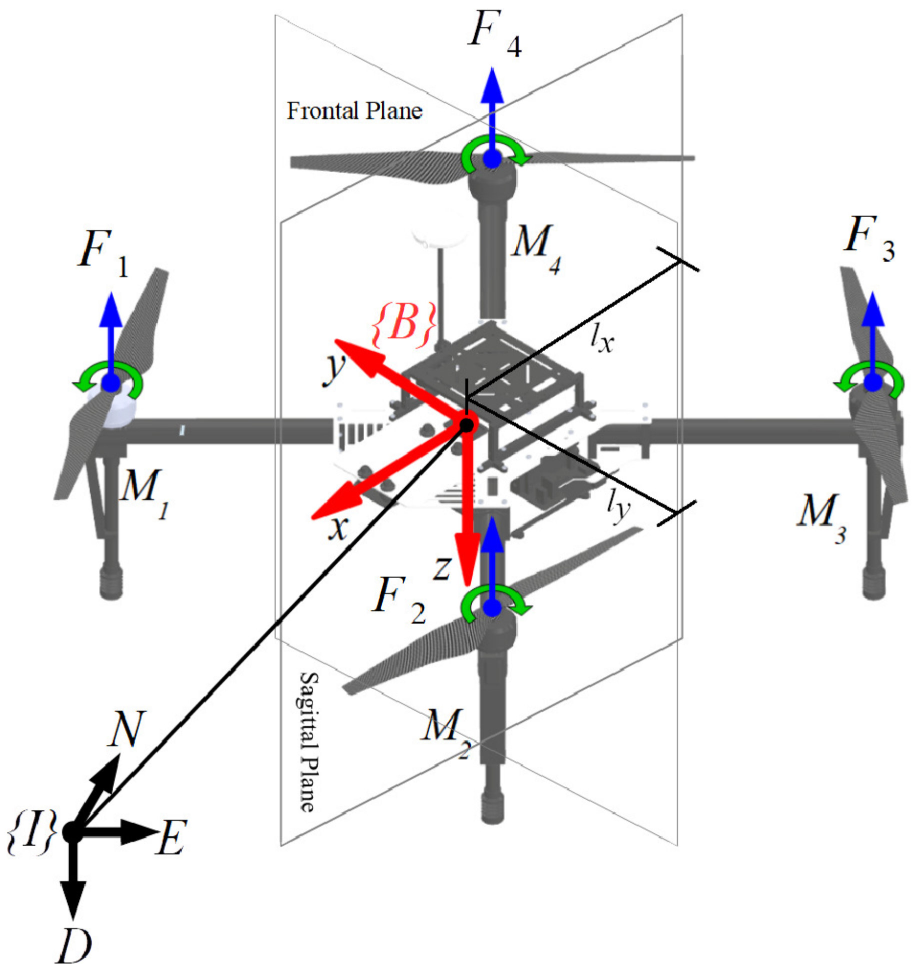


Figure 3.2: Vehicle reference frame, forces and moments.

where $\underline{p}$ is the position vector, ${}^{I}\underline{R}_B$ is the DCM for body to inertial transformation, $\underline{f}_B$ is total thrust generated by the propulsion system and $\underline{g}$ is the gravitation vector.

The rotational equation of motion is given by Eq. 3.2,

$$\dot{\underline{\omega}}_B = -\underline{J}^{-1}\left(\underline{\omega}_B \times (\underline{J}\ \underline{\omega}_B)\right) + \underline{J}^{-1}\underline{M}_B \tag{3.2}$$

where $\underline{\omega}_B$ is the body angular velocity vector, $\underline{J}$ is the moment of inertial matrix and $\underline{M}_B$ is a vector of moments.

Derivation of these equations is based on two main assumptions namely, gravitation attraction is constant given by its value at sea-level and at that latitude. For low altitude flights, the curvature of the earth can be considered insignificant, hence, assuming the earth to be locally flat (flat-Earth assumption) [162]. The resulting equations, Eq. 3.1 and Eq. 3.2 are thus known as the flat-Earth equations of motion.

3.3.2 Kinematic Motion Model

This section introduces the equations describing the motion of a quadrotor-like rigid-body with disregard for forces producing this motion. These equations define the attitude of the body frame relative to inertial frame. The aerospace $Z-Y-X$ rotation convention is adopted for inertial to body transformation, where $Z-Y-X$ correspond to ψ degrees rotation around the z-axis, θ degrees rotation around the y-axis and ϕ degrees rotation around the x-axis. The resulting inertial-to-body DCM is presented in Eq.3.3.

$$
{}^{B}\underline{R}_{I} = \begin{pmatrix} c(\theta)c(\psi) & s(\phi)s(\psi) & -s(\theta) \\ s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & s(\phi)c(\theta) \\ s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) & c(\phi)c(\theta) \end{pmatrix} \tag{3.3}
$$

where $c := cos$ and $s := sin$.

Applying ${}^{B}\underline{R}_{I}$ to the strapdown equation ${}^{B}\underline{\dot{R}}_{I} = -\underline{\omega}_{B} \times {}^{B}\underline{R}_{I}$ and performing coefficient comparison results in the sough after kinematic equations, presented here in Eq. 3.4, where $\underline{\omega}_{B} = (p, q, r)^{T}$. For numerical stability, the pitch angle is limited to the range $\theta \in (-90^{\circ}, 90^{\circ})$.

$$
\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & tan(\theta)sin(\phi) & tan(\theta)cos(\phi) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)/cos(\theta) & cos(\phi)/cos(\theta) \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.4}
$$

Together the translation equations Eq. 3.1, rotational equations Eq. 3.2, kinematic equations Eq. 3.4 and navigation equation Eq. 3.5 are used to derive a set of first order differential equations Eq. 3.6 used to simulate the quadcopter dynamics.

$$
\underline{\dot{x}} =^{I} \underline{\dot{R}}_{B}\underline{v}_{B} \tag{3.5}
$$

where $\underline{x}$ is the state vector and $\underline{v}_{B}$ is the vehicle translational velocity measured in the body fixed frame.

$$
\left.\begin{aligned}
&\text{Force equations} \\
&\dot{v}^{b}_{x} = -gs\theta + rv^{b}_{y} - qv^{b}_{z} \\
&\dot{v}^{b}_{y} = gs\phi c\theta - rv^{b}_{x} + pv^{b}_{z} \\
&\dot{v}^{b}_{z} = -\frac{U_1}{m} + gc\phi c\theta + qv^{b}_{x} - pv^{b}_{y} \\
&\text{Navigation equations} \\
&\dot{x} = v^{b}_{x}c(\psi)c(\theta) + v^{b}_{y}\left(c(\psi)s(\psi)s(\theta) - c(\phi)s(\psi)\right) \\
&\qquad + v^{b}_{z}\left(s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)\right) \\
&\dot{y} = v^{b}_{x}c(\theta)s(\psi) + v^{b}_{y}\left(c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)\right) \\
&\qquad + v^{b}_{z}\left(c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi)\right) \\
&\dot{h} = v^{b}_{x}s(\theta) - v^{b}_{y}\left(c(\theta)s(\phi)\right) - v^{b}_{z}c(\phi)c(\theta) \\
&\text{Moment equations} \\
&\dot{p} = \frac{(J_{yy} - J_{zz})qr}{J_{xx}} + \frac{U_2}{J_{xx}} \\
&\dot{q} = \frac{(J_{zz} - J_{xx})pr}{J_{yy}} + \frac{U_3}{J_{yy}} \\
&\dot{r} = \frac{(J_{xx} - J_{yy})pq}{J_{zz}} + \frac{U_4}{J_{zz}} \\
&\text{Kinematic equations} \\
&\dot{\phi} = p + qt(\theta)s(\phi) + rt(\theta)c(\phi) \\
&\dot{\theta} = qc(\phi) - rs(\phi) \\
&\dot{\psi} = qs(\phi)/c(\theta) + rc(\phi)/c(\theta)
\end{aligned}\right\} \tag{3.6}
$$

where $c := cos$, $s := sin$ and $t := tan$.

The model in Eq. 3.6 assumes symmetry in the frontal and sagittal planes of the vehicle, as shown in Figure. 3.2. The state vector $\underline{x} \in \mathbb{R}^{12}$ consists of three position coordinates specifying the potential energy, three translational velocities specifying translational kinetic energy, three angular velocities specifying rotational kinetic energy and three Euler angles specifying the attitude of the vehicle [162]. The inputs U_1, U_2, U_3 and U_4 are given as indicated in Eq. 3.7.

$$\left.\begin{aligned} U_1 &= F_1 + F_2 + F_3 + F_4 \\ U_2 &= (F_2 + F_3)\, l_y - (F_1 + F_4)\, l_y \\ U_3 &= (F_1 + F_2)\, l_x - (F_3 + F_4)\, l_x \\ U_4 &= (M_1 + M_3 - M_2 - M_4) \cdot \sqrt{\left(l_x^2 + l_y^2\right)} \end{aligned}\right\} \tag{3.7}$$

where F_i is the thrust for rotor i, M_i is the moment of rotor i, and l_x and l_y are the x and y displacements of the rotors from the centre of mass respectively.

Designing simulations for a MAV employs such equations of motion together with control algorithms. The control strategies require careful simulation before they can be deployed on actual platforms, on which they take on the form of embedded implementation of their discrete equivalents.

3.4 Chapter Summary

This chapter presents an extension of the existing MTOW UAV classification schemes in Sect. 3.1. The extended scheme includes the nano class, which was subsumed by the micro class in the previous schemes, bringing the number of classes to seven namely, nano, micro, mini, small, lightweight, normal and large aerial vehicle classes. Then discusses the vehicle selection process that resulted in selection of $\times$-configuration micro quadcopters as the vehicles of interest in this research work, a choice attributed to their low inertia and simplistic structural design. The chapter also introduces the equations of motion in Sect. 3.3 that have been applied in numerical and graphical simulation of the two selected quadcopters. The flat-Earth assumption made in the derivation process of these equations limits their suitability to low altitude flights, which makes them appropriate for the studies in this work since the applied quadcopters belong to the “Open” category of unmanned aircraft systems according to the European Union Aviation Safety Agency (EASA) categorization, which are legally restricted to AGL not greater than 120 m [163].

Chapter 4

Autonomy

Nevertheless the difference in mind between man and the higher animals, great as it is, is certainly one of degree and not of kind.

Charles Darwin

Robotic systems are finding applications in more complex and highly dynamic territories whose control demands exceed human regulation capability rendering such systems under exclusive human control unstable under reasonable operating conditions. One possible solution to this instability is to slow down the controlled system's dynamics. But a more lucrative solution would be to move some or all the decision making onboard the vehicle. Such a move would require platforms to have the ability to make one of three deliberate choices namely, (1) choice of a goal or state, (2) choice of action for achieving a desired goal or state (3) choice of goal-action pair (behaviour). Systems with such decision-making capability are known as autonomous systems.

The ability of autonomous functioning is known as autonomy, which is defined here as the system's ability to select an intermediate goal and/or course of action to achieve that goal, as well as approve or disapprove any previous and future courses of action while achieving its overall goal. The goals are the main drivers of any robot action and may originate from an operator or from system constraints set by the designer. According to the etymology of the word "robot", it means a servant and that is what robots are intended to be. Therefore, the possibility of systems generating their own ultimate goals and constraints, which falls outside this definition is excluded. This means absolute autonomy [24] is beyond such machines, hence unless otherwise stated, herein autonomy implies relative autonomy.

The attractiveness of autonomy has resulted in a diverse selection of autonomous systems with a range of capabilities, raising the need to characterize and ultimately regulate such systems. To quote H. James Harrington, "Measurement is the first step that leads to control and eventually to improvement. If you can not measure something, you can not understand it. If you can not understand it, you can not control it. If you can not control it, you can not improve it." Therefore, regulating autonomy requires the ability to measure autonomy. Unfortunately, a set of autonomy metrics that is easily measurable, broad enough to capture autonomy evolution in a system and with good output resolution is still lacking.

This chapter introduces a set of proposed autonomy metrics and associated mathematical models for mapping them to autonomy measures. The metrics were derived from robot task characteristics, which themselves were determined by relating robot task characteristics to human job characteristics as applied in industries. It has also been realized herein that autonomy is purposive and environmental specific, where the former means autonomous functioning is designed to address a specific goal, while the latter highlights the fact that autonomous functioning is tied to a predefined world setting. Therefore, complete designation of autonomy for a system is defined with respect to a purpose, environment and additionally, performance. The next section discusses proposed approaches for quantifying autonomy.

4.1 Autonomy Measurement

The first step towards building an autonomous system is establishment of lower-level capabilities to build into the system. The low-level capabilities interact to form and/or support higher-level capabilities. Engineering specifications associated with these capabilities then provide the structure and control system architectural design foundation for the robot. Against this background, it is asserted that capabilities determine the fundamental structure of an autonomous system. This observation points to the fact that fully autonomous systems ultimately differ in appearance from their manually controlled counterparts. It should also be pointed out that to the end-users and regulatory authorities, it is not the precise technology, but functional capabilities that are of at most importance.

Two categories of capabilities have been proposed namely, behavioural and cognitive capabilities. Behavioural capabilities constitute the low-level behaviours of a robot, while the cognitive capabilities include the intellectual functions of a robot. To claim autonomous functioning, robots must possess both sets. Cognitive capabilities are implemented in the planning layer or for a fixed plan, in the executive layer of the control architecture, while behavioural capabilities span between behavioural and executive control layers as indicated in Fig. 4.1. Cognitive response is conditioned on knowledge that shapes the belief of a system upon which decisions are made. Behavioural capabilities depend on actuator, sensor and data-link technology, while cognitive capabilities depend on microprocessor technology, and algorithm performance and computational complexity.

For a specific task in a specific environment, a list of capabilities is not sufficient for determining the autonomy of a system. Additional information like the associated performance capacity of the capabilities is also necessary, as autonomous systems differ not only functionally (level of autonomy), but also performance wise (degree of autonomy). This means that systems with similar capabilities may exhibit varying performances as a matter of technological and economical factors. This raises a need for additional autonomy correlated information sources. Level of Autonomy (LoA) is an ordinal scale that indicates a system's independence from external intervention when in operation and ranges from teleoperated or remotely operated to fully autonomous systems. Degree of Autonomy (DoA) is a ratio scale measure of performance superiority of a system at its level of autonomy. This distinction is important because it emphasizes the difference in performance that exists among systems at a similar level of autonomy, which may result from technology, structural design or system software and algorithmic differences.

After knowing the distinction between these two classes of autonomy, the next question to ask is "how to measure LoA and DoA for a system?". These classes can be assessed from contextual perspective—derived from robotic platform, task and environmental characteristics, or a non-contextual perspective—derived from only the robotic platform characteristics [164]. Measuring LoA and DoA requires finding associated metrics or at least metrics correlated with each class. These metrics ought to be easily measurable, broad enough to capture autonomy evolution in a system, with good output resolution [25], highly sensitive and generalizable to a broad range of systems.

Since the performance lower-bound of autonomous systems is human operator performance, it is logical to start by analysing human-based jobs. Table 4.1 lists the four major human-job characteristics applied in industries. In the rightmost column is a mapping of the human-job characteristics to robot-task characteristics. These robot-task characteristics namely, capabilities, trust factor, performance capacity and environmental complexity formed the basis upon which the proposed metrics were derived. In the review on autonomy assessment criteria presented in [27], it is indicated that 25% of the considered twenty frameworks viewed autonomy from environment, mission and self-autonomy perspectives. This in itself shows that the four selected characteristics have merit, but the fact that only 25% of the frameworks shared this view shows how oblivious the previous works were to their strength.

Table 4.1: Human-job characteristics mapped to robot-task characteristics. The human-job characteristics and importance weights are adopted from [5].

Human worker Characteristics	Importance (%)	**Robot worker Characteristics**
Skills	50	Capabilities
Responsibility	25	Trust factor
Effort	15	Performance capacity
Working conditions	10	Environmental complexity
Total	**100**	

4.1.1 Autonomy Metrics

This section describes the four autonomy characteristics mentioned in Table 4.1, their justification and mathematical representations. These characteristics form the basis for autonomy measurement metrics.

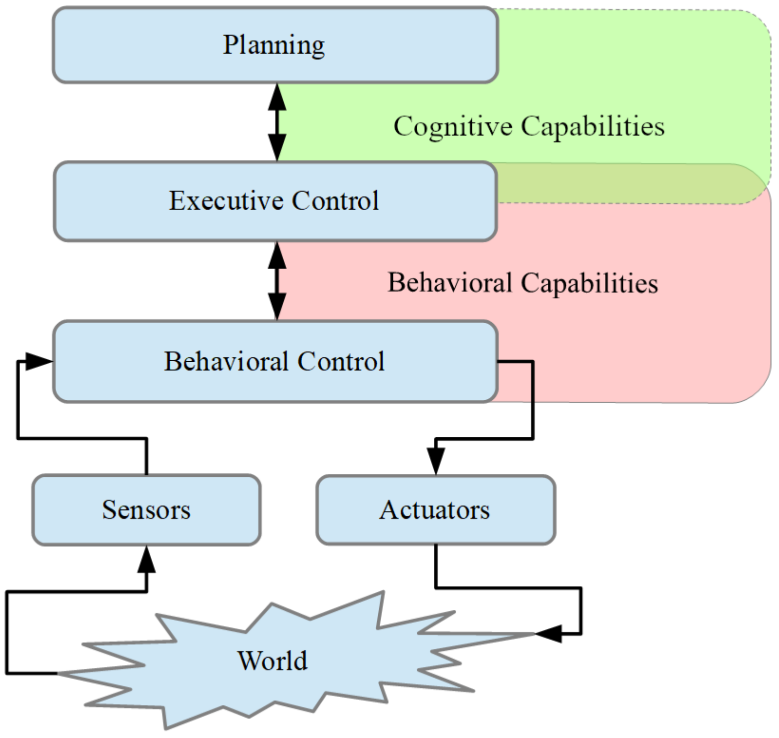


Figure 4.1: Three-tiered architecture overlaid with capability classes. The intersection between capability classes represents the fact that these classes can influence each other.

Capabilities

Having control over the existing large variety of autonomous robotic systems is a long-standing challenge, which is of interest to autonomous robot manufacturers and regulatory authorities. The good news is that these autonomous systems operate within physical limits and what they do in terms of behaviour is largely determined by the human operators or developers during operation and system design respectively. As indicated in Table 4.1, capabilities are the most important of all characteristics. Having a grasp of their influence on the behaviour of autonomous systems is essential to achieving the benefits of regulated autonomous functioning.

Capabilities are the most researched aspects of autonomous systems in relation with autonomy evaluation. A review of twenty autonomous assessment frameworks presented in [27] revealed that capabilities are the dominant approach followed by mission, interaction, environment, etc. The same study listed the dominant capabilities as problem solving, motion, perception, communication, acquisition and self-preservation. These capabilities are mappable to our proposed capability classes of cognitive and behavioural capabilities, with problem solving and perception belonging to cognitive capabilities, while motion, communication, energy gathering and self-preservation belonging to behavioural capabilities. This list is not exhaustive, so, a more comprehensive list of capabilities as of the current level of research and technology in the field of robotics is provided, which includes cognitive and behavioural capabilities on the left and right respectively:

- Basic perception
- Situation awareness
- Natural language processing
- Machine vision
- Learning
- Decision making or planning

- Mobility
- Object manipulation
- Communication
- Energy gathering
- Self-preservation

For humans, each job is associated with a set of skills. This concept is extendable to robot tasks by associating each task with a set of capabilities. This associated set of capabilities then becomes a tool for scoring the capability aspect of any robotic agents.

Trust Factor

Trust factor defines the risk level/uncertainty or performance tolerances acceptable during execution of a specific capability. It encapsulates criticality of consequences that may ensue as a result of execution mishaps by the robotic agent. If a system is built for a specific task or a set of tasks, the trust factor can be relaxed or tightened by adjusting the expected precision, hence admitting lower levels of autonomy and higher levels of autonomy respectively.

This outlook on trust factor evaluation is beneficial in a way that it eliminates the subjective urge to compare different capabilities with the aim of determining one as more trustable compared to another. This changes the focus of trust factor assessment from the capability itself to the required performance precision. An example of this would be lane following by an autonomous vehicle. This task can be made more complex by demanding the vehicle to follow the lane to within a tolerance of $\pm 4\%$ of the lane width or less complex by demanding the vehicle to track the lane to within $\pm 20\%$ of the lane width. It should be noted that in so doing, the underlying lane tracking technology enabling achievement of the set tolerance is irrelevant. The actual precision achievable by the system and the desired precision are two different things, but their ratio encodes trust and responsibility in the system for this particular task. Therefore, trust factor metric is represented as (C_{TF}), which is the ratio of actual precision to desired precision of a system for a capability i. Mathematically, it is expressed as indicated in Eq. 4.1, where σ^2_{des} is the desired variance that could be set by the operator or regulatory authorities for a particular task in a specific environment and σ^2_{act} is the actual variance of the system that is empirically determined.

$$C_{TF,i} = \frac{\sigma^2_{des,i}}{\sigma^2_{act,i}} \tag{4.1}$$

Generally, the expected value of a random variable is a location parameter, making variance and trust factor location invariant. For events that involve observations of the probability of success p of independent experimental trials, $\sigma^2 = p\,(1-p)$. When $\sigma^2_{des,i} = 0$, two trust factor outcomes are possible as indicated in Eq. 4.2, which means the system is either not trustable at all since it has a non-zero variance for the first case or is infinitely trustable when its variance is zero. These cases are very unlikely as real systems generally do not exhibit 100% certainty under natural operating conditions.

$$C_{TF,i} = \begin{cases} 0 & \sigma^2_{act,i} \neq 0, \sigma^2_{des,i} = 0 \\ \infty & \sigma^2_{act,i} = 0, \sigma^2_{des,i} = 0 \end{cases} \tag{4.2}$$

Performance Capacity

Capabilities can be developed to varying capacities which collectively determine the responsiveness of a robotic system. The observable factor for this characteristic is response time. This observable factor incorporates two important performance characteristics namely, drive force and spatial extent of the system. The performance metric is derived from response time as the capability execution temporal rate, representing a fraction of capability executed in one second. Here we assume capability execution has a linear temporal relationship. This performance capacity metric is represented by Eq. 4.3.

$$C_{PC,i} = \frac{1}{T_{cap,i}} \tag{4.3}$$

where $T_{cap,i}$ is time in seconds needed to execute capability i. The resulting performance capacity has units of s^{-1}.

Environmental Complexity

Robotic platforms are part of the environment in which they operate, a concept known as situatedness [94, 95]. Environments afford perception, localization, obstacle detection and robot locomotion. But these perceptible affordances are dynamic which complicates the associated processes. Environments impose both kinematic and geometric constraints on the vehicle motion, which further exacerbate the situation. These factors together define the environmental complexity.

Performance and environmental complexity have an inverse relationship [26]. In away, the environment implicitly imposes performance requirements on any system that operates on it, and autonomous systems are no exception. This inverse relationship is also mirrored in the law of requisite variety, which is restated here as “only variety can control variety” and is related to Shannon’s theorem, which is identical to entropy [165]. Hence, the use of entropy to model environmental complexity in this work.

Herein, environmental complexity is inferred from spatial entropy as a measure of certainty of finding a free space within a region delineated by a mask. The mask could be shaped as in Fig. 4.2, i.e., of one, two or three dimensions with size 1×3, 3×3 and $3 \times 3 \times 3$ units respectively. This complexity analysis assumes a simple environment with only static obstacles. To evaluate environmental complexity, assuming a planar environment, the first step is to divide the world into a regular grid with squares equal to twice the radius of the robot on the side. Any square partially or fully occupied by an obstacle is considered blocked. This results in an approximate decomposition of the environment, an example of which is shown in Fig. 4.3a, in which the dark cells are blocked and the light cells are free. Fig. 4.3b shows a 3×3 mask in the upper left corner, used to determine the local probability also known as the sampling density of the free space.

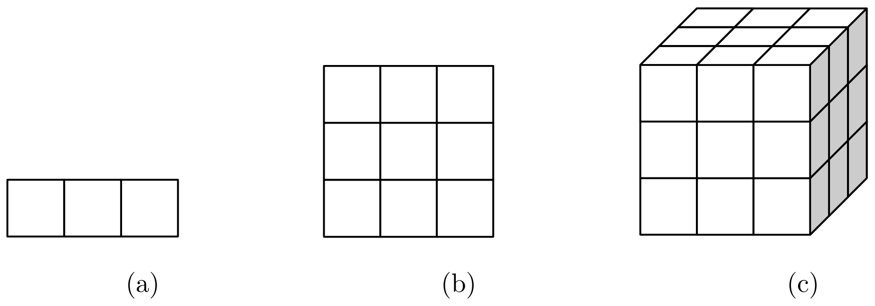


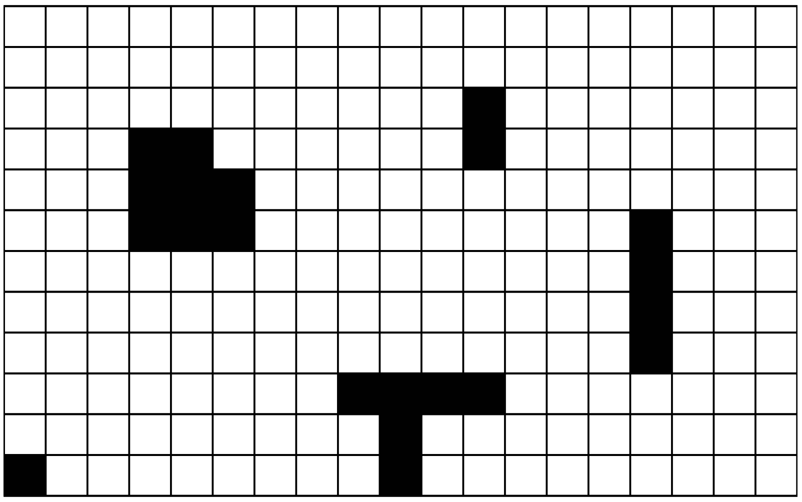
Figure 4.2: (a) One-dimensional mask. (b) Two-dimensional mask. (c) Three-dimensional mask.

The environmental complexity C_{EC} is specific to a vehicle, i.e., disparate vehicles would view a similar environment differently as far as its complexity is concern. For this 2D case, the applied 3×3 mask is associated with ten possible sampling densities as indicated in Table 4.3. The sampling densities represent the probability of free squares within any masked region with the assumption of uniform distribution. Each sampling density has an associated entropy $E_{\rho_i} = -\rho_i \cdot log(\rho_i)$, which indicates its local complexity. The overall environmental complexity is then measured as the expected entropy resulting from sliding the mask through the environment. This is mathematically expressed as in Eq. 4.4.

$$C_{EC} = \begin{cases} \sum_{i=0}^{r=9} P(\rho = \rho_i) \cdot E_{\rho_i} & \text{if } P(\rho = \rho_0) \neq 1 \\ \text{undefined} & \text{if } P(\rho = \rho_0) = 1 \end{cases} \tag{4.4}$$

where r is the mask size, $P(\rho = \rho_i)$ is the probability of observing sampling density ρ_i in the environment, determined by the frequency of this density during sliding of the mask through $(n-2) \times (m-2)$ grid steps contained within the dashed polygon of Fig. 4.3b. It should be noted that for the case $P(\rho = \rho_0) = 1$, C_{EC} is undefined as for such a case the size of the free space is zero. This demonstration is for a 2D case, but the steps can be applied to 1D and 3D environments as well, in which cases the masks are also 1D and 3D in shape respectively. The sampling densities and associated entropies for the 1D and 3D cases are indicated in Table 4.2 and Table 4.4 respectively. This environmental complexity model accounts only for geometry related complexity resulting from presence of static obstacles. Factors associated with environmental affordance like changes in lighting conditions and weather affect only specific sensing technologies, hence, do not affect environmental complexity in general.

In summary, the four metrics can be applied for the purpose of non-contextual or contextual autonomy assessment as indicated in Table 4.5, where the former defines autonomy with no regard for environmental complexity, trust factor and performance capacity, while the latter makes no such assumptions. These are also known as passive and active autonomy respectively [27].


(a)

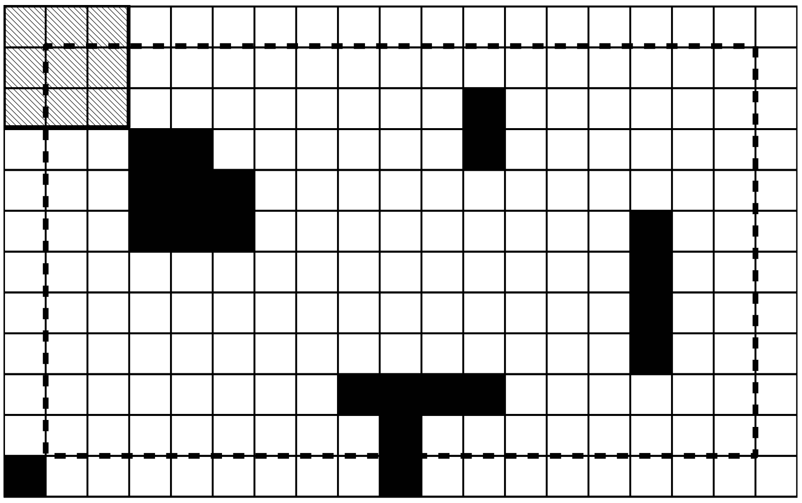

(b)

Figure 4.3: (a) Approximate decomposition of a robot environment. (b) A mask in the upper-left corner used to measure sampling density. The mask is moved step-wise to all squares within the dashed polygon.

Table 4.2: One-dimensional mask sampling densities and associated spatial entropies.

	Sampling Densities			
ρ_i	0/3	1/3	2/3	1
E_{ρ_i}	0.000	0.528	0.390	0.000

Table 4.3: Two-dimensional mask sampling densities and associated spatial entropies.

	Sampling Densities									
ρ_i	0	1/9	2/9	3/9	4/9	5/9	6/9	7/9	8/9	1
E_{ρ_i}	0.000	0.352	0.482	0.528	0.520	0.471	0.390	0.282	0.151	0.000

Table 4.4: Three-dimensional mask sampling densities and associated spatial entropies.

	Sampling Densities						
ρ_i	0.000	1/27	2/27	3/27	4/27	5/27	6/27
E_{ρ_i}	0	0.176	0.278	0.352	0.408	0.451	0.482
ρ_i	7/27	8/27	9/27	10/27	11/27	12/27	13/27
E_{ρ_i}	0.505	0.520	0.528	0.531	0.528	0.520	0.508
ρ_i	14/27	15/27	16/27	17/27	18/27	19/27	20/27
E_{ρ_i}	0.491	0.471	0.447	0.420	0.390	0.357	0.321
ρ_i	21/27	22/27	23/27	24/27	25/27	26/27	1
E_{ρ_i}	0.282	0.241	0.197	0.151	0.103	0.052	0.000

Table 4.5: LoA and DoA characteristics.

Characteristics	**LoA**	**DoA**
Capabilities	✓	✓
Trust factor		✓
Performance capacity		✓
Environmental complexity		✓

4.1.2 Level of Autonomy and Degree of Autonomy

LoA is inherently non-contextual while DoA is inherently contextual. LoA considers only presence/absence of mandatory autonomous capabilities, while DoA considers not only capabilities, but also trust factor, performance capacity and environmental complexity.

The LoA assessment procedure applied here is similar to the commonly applied method of capability comparison, where a list of required capabilities for each autonomy level for a particular task are compared with the vehicle's capabilities and the system is assigned to the highest matched level. This approach applies to any autonomous system, but MAVs are used for demonstration purposes as they are the platform of focus in these projects.

This work adopts a discretization resolution of eleven levels of autonomy as in [25] and [28], ranging from remotely controlled systems at level 0 to fully autonomous systems at level 10. The eleven levels and their descriptions are presented in Table 4.6. Level 0 systems are continuously externally controlled, levels $1-2$ systems provide assistance to the external controller, levels $3-4$ systems share control (through cooperation or collaboration) between the external controller and onboard system, levels $5-7$ are fully autonomous but still require presence of external supervision, while levels $8-10$ systems are fully autonomous with no need of external supervision, but levels 8 and 9 offer only conditional autonomous functioning.

Each level in Table 4.6 is associated with a set of capabilities for a particular task, which act as a reference for LoA assessment. LoA for any system at performing a particular task is given by the level whose capabilities' control strategy matches that of the system.

Degree of autonomy (DoA) is determined from trust factor and performance capacity using Eq. 4.5.

$$DoA = \sum_{i=1}^{n} (C_{TF,i} \times C_{PC,i}) \tag{4.5}$$

where $C_{TF,i}$ is the trust factor for capability i, $C_{PC,i}$ is performance capacity for capability i and n is the number of relevant capabilities.

Table 4.6: Level of autonomy chart.

Level	Description
10	Fully autonomous.
9	Environment-dependent full autonomy.
8	Limited-performance full autonomy.
7	Fully autonomous with on-request supervision or consultation.
6	Fully autonomous with more unsupervised than there are supervised modules.
5	Fully autonomous with at least as many supervised as there are unsupervised modules.
4	Shared control (externally controlled and unsupervised modules, collaborative).
3	Shared control (externally controlled and supervised modules, cooperative).
2	Assisted external control with basic actuation.
1	Assisted external control with actuation proposition.
0	Externally controlled (Remote control).

Therefore, defining only the level of autonomy as it is being done to date is not sufficient for defining autonomy of a robot. Herein, a three-part designation of autonomy including level of autonomy, degree of autonomy and their associated environmental complexity is proposed. Therefore, autonomy for any robotic system is specified as "α DoA at β LoA in an environment of γ complexity", where α and β, and γ are the DoA model output, LoA level index and environmental complexity model output respectively.

4.2 Autonomy Framework Evaluation Results

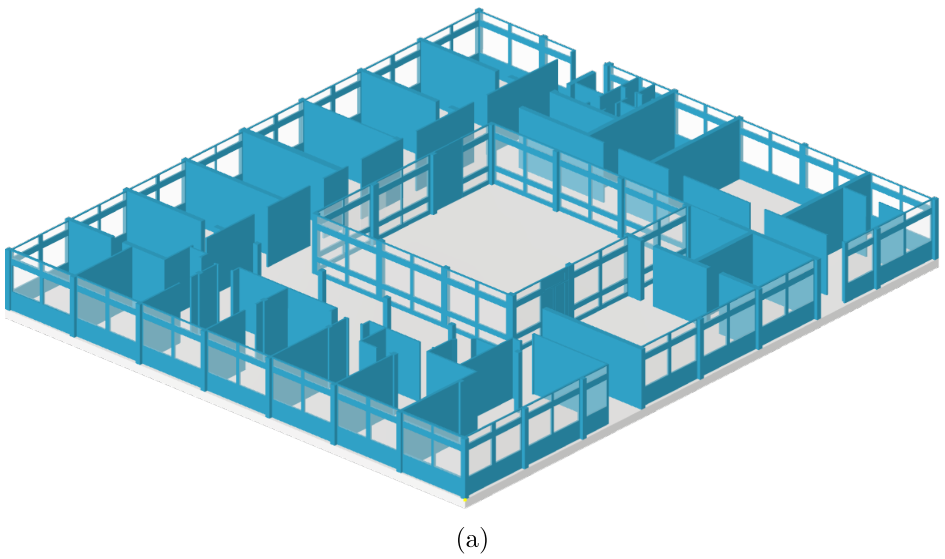
In this section, application demonstrations of the proposed autonomy framework are presented. The demonstrations include a case study of a hypothetical MAV navigation scenario, Defense Advanced Research Projects Agency (DARPA) subterranean challenge competition rules analysis and a case study with six unmanned aerial systems used previously to evaluate 10 existing autonomy evaluation frameworks in [6].

4.2.1 MAV Demonstration

Here, a generic autonomous MAV with the following capabilities (such capabilities would normally be listed in the vehicle's datasheet) is deployed for the task of indoor navigation:

- Supervised path tracking.
- A GNSS-independent onboard localization system with a standard deviation of 0.1 m.
- Supervised static obstacle avoidance.
- Unsupervised path planning.
- Unsupervised battery level monitoring and reporting.

The navigation environment is an office building showed in Fig. 4.4a with hallways wide enough for this $\varnothing 1.0m \times 0.35m$ airframe MAV to fly through. The goal here is to assess the level of autonomy and degree of autonomy of this vehicle in this specified environment.



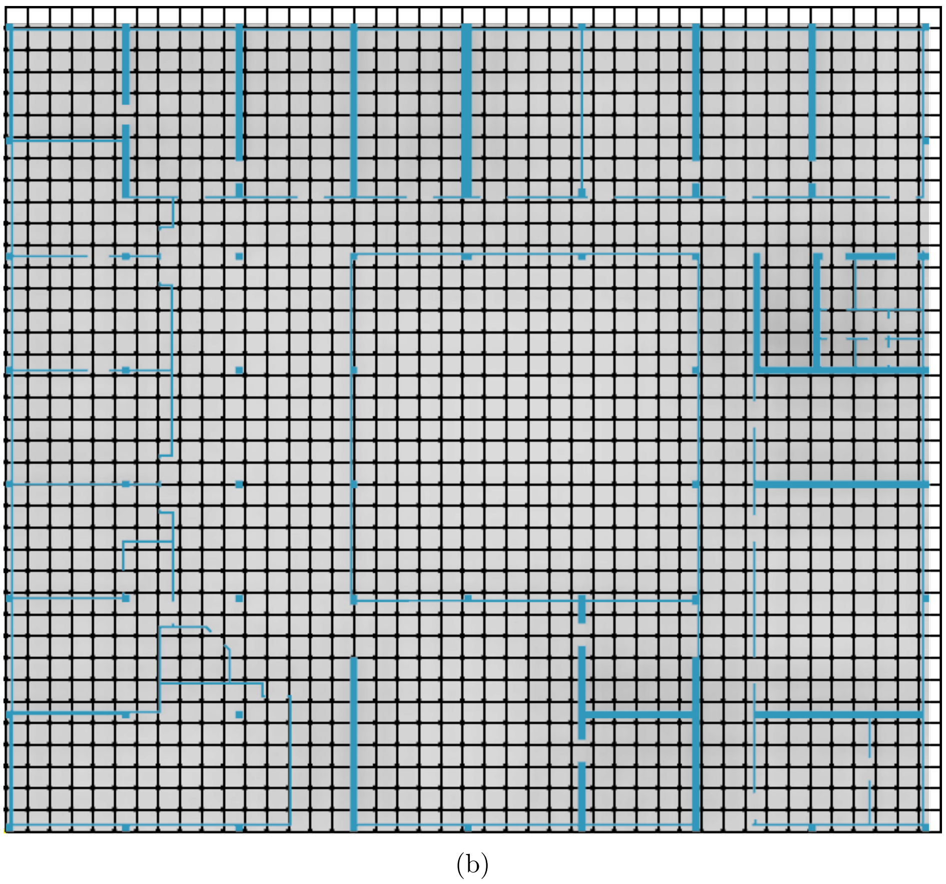


Figure 4.4: (a) Isometric view of the test environment. (b) Top view of the test environment with an overlaid approximate decomposition grid.

Level of Autonomy Determination

LoA assessment is a two-step process. The first step compares vehicle capabilities to the core navigation capabilities to ensure navigation viability. If passed, the control strategies of the vehicle capabilities are then compared to Table 4.6 for level determination. As evident from Table 4.7, the MAV possesses all the core navigation capabilities. But some of them require supervision, which will impact the vehicle's LoA. With reference to Table 4.6, the MAV is of level of autonomy six (LoA 5).

Table 4.7: List of navigation capabilities for LoA assessment.

Navigation Core Capability	**MAV Navigation Capability**	**Operator Involvement**
Path planning	✓	Unsupervised
Localization	✓	Unsupervised
Path tracking	✓	Supervised
Obstacle avoidance	✓	Supervised

Environmental Complexity Determination

Since the environment contains only vertical obstacles of similar height, sampling density probabilities can be easily calculated by sliding a 3×3 mask at any fixed height, instead of a $3 \times 3 \times 3$ mask at progressive height steps. The results of this process are showed in Table 4.8, where environmental complexity is the sum of the right most column entries, which in this case totals to 0.292 bits.

Table 4.8: Environmental complexity assessment.

ρ	$\mathbf{E}$	$\mathbf{P}(\rho = \rho_\mathbf{i})$	$\mathbf{P}(\rho = \rho_\mathbf{i}) \cdot \mathbf{E}$
0	0.00000	0.00542	0.00000
1/9	0.35221	0.00136	0.00048
2/9	0.48221	0.02439	0.01176
3/9	0.52832	0.04539	0.02398
4/9	0.51997	0.11314	0.05883
5/9	0.47111	0.06775	0.03192
6/9	0.38998	0.39295	0.15324
7/9	0.28200	0.02575	0.00726
8/9	0.15104	0.02981	0.00450
1	0.00000	0.29404	0.00000
		1.00000	C_{EC} = **0.29197**

Trust Factor Determination

The trust factor determination process starts by assessing the trust factor of each of the core navigation capabilities namely, path planning, path tracking, localization and obstacle avoidance. Assuming the following desired performance requirements, 70% path planning success rate, 90% path tracking success rate, 0.05 m localization standard deviation and 80% obstacle avoidance success rate.

Several test trials have been conducted with the aim of obtaining actual measurements corresponding to the desired properties. The outcomes of these tests are summarised in Table 4.9. For path planning, optimality is not considered since the path planner of choice is a sampling-based path planner. Additionally, the path planning process was allowed a maximum planning and path simplification duration of 1 sec. The actual localization system has a standard deviation of 0.1 m. For this evaluation, ten navigation trials were conducted with the same starting configuration, but random goal configurations in each of the thirty rooms and one hallway in the test environment.

Table 4.9: Trust factor assessment.

Capability	$\sigma^2_{\mathtt{des}}$	$\sigma^2_{\mathtt{act}}$	$\mathtt{C_{TF,i}}$
Path planning	0.2100	0.2415	0.868
Localization	0.0025	0.0100	0.250
Path tracking	0.0900	0.1600	0.563
Obstacle avoidance	0.1600	0.1476	1.084

Performance Capacity Determination

Performance capacity metric is the execution frequency of a capability, number of execution cycles in a second. In other words, it is the reciprocal of execution time $T_{cap,i}$. After running several simulation trials, an average path planning time of 0.407 seconds was obtained, the localization module published vehicle locations at a frequency of 20 Hz, obstacle avoidance took a minimum of 0.92 seconds from stimulus to response execution and path tracking nominal speed is 0.4 m/s. Applying these measurements to the proposed performance capacity metric in Eq. 4.3 yielded the results summarised in Table 4.10.

Table 4.10: Performance capacity assessment.

Capability	$\mathtt{C_{PC,i}}$ (Hz)
Path planning	2.458
Localization	20.000
Path tracking	0.400
Obstacle avoidance	1.087

Degree of Autonomy Determination

Now that the trust factors and their associated performance capacities are known, these are then applied to Eq. 4.5 to obtain the degree of autonomy of this MAV, which in this case as indicated in Table 4.11 is 8.537 Hz. This is interpreted as efficiency adjusted performance rate.

Table 4.11: Degree of autonomy assessment.

Capability	$\mathtt{C_{TF,i}}$	$\mathtt{C_{PC,i}}$	$\mathtt{C_{TF,i}} \times \mathtt{C_{PC,i}}$
Path planning	0.868	2.458	2.137
Localization	0.250	20.000	5.000
Path tracking	0.563	0.400	0.225
Obstacle avoidance	1.084	1.087	1.178
		Total	**8.540** Hz

Therefore, this MAV has a level of autonomy LoA 5, and a degree of autonomy of 8.540 Hz in an environment of complexity 0.292 bits.

4.2.2 DARPA Subterranean Challenge

This section compares the recently concluded DARPA subterranean (SubT) challenge scoring criteria with the metrics proposed in this work. It also shows how application of these metrics can provide a self-evaluation measure indicative of a team's likelihood of qualifying for such competitions prior to actual qualification rounds. DARPA SubT challenge was a three-year (September 2018 - August 2021) robotics competition organised by the DARPA to motivate development of state-of-art mapping, navigation and spatial search solutions for dynamic complex unknown subterranean environments. The competing systems demonstrated autonomous functioning, perception, mobility and networking capabilities. [166]. The challenge included two competitions namely, systems competition and virtual competition, where the former involved actual hardware operating in actual environments and the latter was exclusively virtual, with simulated vehicles operating in graphically simulated environments.

The scoring objective for this competition was the total number of accurately reported artefacts within a limited time window. Since the artefacts were spatially distributed within the subterranean environment, and the exploration time limited, this implicitly bounded the operating speed. Validity of an artefact detection depended on its accurate identification and estimation of its 3D position to within ± 5 m. The systems competition was evaluated based on one final run, which provided no statistical possibility of accounting for performance variability. The decision of scoring only on the final round was justified by avoidance of having to run a statistically significant number of trials in the real environment that would require a lot of resources. In the virtual competition however, the final score was averaged on m scenarios and n trial runs per scenario to account for random variability in performance.

Looking at each of the SubT challenge meta-capabilities namely, mapping, navigation and spatial searching from an autonomy assessment point of view revealed the following: (1) Any mapping framework with a trust factor $C_{TF} < 1.0$ has an increased probability of reporting artefacts with position errors outside the allowed error range of ± 5 m assuming $3\sigma_{act} \approxeq 5$ m. (2) A mapping framework with $C_{PC} < 0.1$ s^{-1} would not qualify for the competition. (3) The final event had forty and twenty artefacts, but allowed only forty-five and twenty-five reports per run for the systems and virtual events respectively. This meant that for any system to stand a chance of searching and reporting all artefacts (assuming perfect artefact position estimation and sufficient exploration speed), it needed to exhibit an identification success rate variance of $\sigma^2_{act} \leq 0.160$ and $\sigma^2_{act} \leq 0.099$ for systems and virtual competitions respectively. (4) For navigation, a speed reference value was neither specified nor inferable from the run duration without environmental knowledge. This should have made it difficult for the competitors to select a suitable exploration speed for the vehicles to maximize coverage with no speed reference.

In summary, a procedure similar to that described in Sect. 4.2.1 can be applied to a vehicle's mapping, navigation and searching capabilities to determine its level of autonomy and degree of autonomy for each respective task. LoA determination is necessary for the systems competition since human intervention was allowed, hence the likelihood of systems with different levels of autonomy. DoA would differentiate the superiority of systems at similar levels of autonomy for the different subdomains.

4.2.3 Other Case Studies

Here, the proposed LoA chart in Table 4.6 is applied to the case study of six UASs that were used to evaluate performance of the ten frameworks in [6]. A more detailed description of the six vehicles is presented in Table B.1 of Appendix B. According to the source article, UAS A is capable of unsupervised execution, but planning is done by an external operator. This is indicative of a collaborative control strategy, hence belongs to level of autonomy LoA 4. UAS B is continuously under external control, hence belongs to LoA 0. UAS C has deterministic supervised behaviours, an external operator performs planning and replanning, and the system executes the plan under supervision. This is indicative of a cooperative control strategy, hence this vehicle is of LoA 3. UAV D has nondeterministic supervised behaviours, system plans, replans and executes all under supervision. This makes it a LoA 5 system. UAS E and UAS F are both fully autonomous systems of LoA 10. This is because autonomous systems are purposive, so, they are not expected to set their own goals and constraints. This shows that the presented LoA chart is well formulated and detailed enough to allow unambiguous classification of robotic systems than existing frameworks. The complete autonomy level classification results of the proposed framework and ten existing frameworks on these case studies is presented in Table B.2 of Appendix B.

4.3 Chapter Summary

Autonomy is a term that has been used on several occasions to describe performance of products like automobiles, cranes, service robots to mention but a few. But what autonomy means quantitatively is undefined or defined ambiguously. This chapter provides a clear description of autonomy and quantitative metrics for measuring it. Any robot task is characterized by capability requirements, trust factor, performance capacity and environmental complexity. Capabilities define a robot's LoA for a particular task. Capabilities, trust factor and performance capacity define a robot's DoA in a specific environment. Besides providing context, environmental complexity is also indirectly integrated into DoA through desired trust factor requirements. Mathematical functions for the metrics and DoA have also been proposed. Unlike the existing measures of autonomy that report only LoA, the proposed framework outputs a three-part autonomy designation, resulting in a less ambiguous, high resolution characterization of a system's autonomy. The framework output specifies the system's DoA score, the associated LoA and complexity of the expected operating environment. Finally, demonstration results of the proposed autonomy evaluation framework on three case studies are presented. The case studies include a generic MAV, DARPA challenge SubT competition and a case study with six UASs that was used to evaluate the performance of ten existing autonomy frameworks in [6].

Chapter 5

Path Planning

This chapter introduces a key competence of navigation known as path planning. The output of any path planning process is a connectivity graph through the free space in the operating environment. There are several approaches for generating such paths, but the most popular are combinatorial, sampling-based and potential field-based path planners. As mentioned in Sect. 2.4, combinatorial and potential field-based approaches do not scale well to higher dimensional and large spaces, while sampling-based approaches scale well to such spaces, but are incomplete in the general sense and produce poor quality paths. Furthermore, potential field-based approaches render themselves well to online implementation, but their gradient descent implementation is susceptible to local minima.

The work in this chapter aims at addressing two main goals namely, online implementation of sampling-based path planning with enhanced success rate and implementation of a combinatorial coverage path planner for planning in large-scale scenarios. Sampling-based path planners have in the past been labelled as having poor quality paths and generally incomplete. This has contributed to their persistent unpopularity despite their growing attractiveness that is attributed to properties like relatively fast planning, computational complexity independent of environmental complexity, ability to handle nonholonomic constraints and high dimensional configuration spaces. Sect. 5.1.1 presents a high success rate, relatively fast, online and adaptive sampling path planner with acceptable path quality. Sect. 5.2 presents a solution to the large-scale coverage path planning problem, which is a combinatorial type path planner, specifically, coverage path planner with exact cellular decomposition functionality.

5.1 Randomized Sampling-based Path Planners

Combinatorial (generally complete) motion planners have prohibitively high computational complexity and memory requirements [167], both of which scale exponentially with the dimensionality of the configuration space [141], hence impractical for complex higher dimensional planning scenarios. Furthermore, they require explicit representation of the obstacle configuration space. These limitations motivated the development of sampling-based path planners whose computational complexity is independent of environmental complexity [143]. Sampling-based methods output either deterministic or random sample sequences. In the former case, resolution tuning is necessary. Here, the focus is on randomized sampling as it requires no parameter tuning, while exhibiting incremental resolution improvement with increased sample density. The main disadvantages of randomized sampling-based approaches are their non-deterministic nature, weak completeness and relatively poor path quality. This work proposes an approach to sampling path planning that improves on these disadvantages, i.e., enabling online planning and replanning and improving on success rate, environmental adaptability and short planning time.

Randomized sampling-based planners randomly sample the configuration space to construct a data structure representative of the free configuration space. These methods can be primarily categorized into roadmap and tree-based methods, Probabilistic Roadmap (PRM) and Rapidly-exploring Random Tree (RRT) respectively [139]. RRT is a variant of Rapidly-exploring Dense Trees (RDTs) with random sampling sequences [138]. A basic PRM randomly samples the configuration space for obstacle free configurations, connects them to build a roadmap during the learning phase, the resulting roadmap from this pre-processing step is saved and queried for paths during the query phase making PRM a multi-query randomized planner. On the other hand, for every query, RRT probes the free configuration space, but does not add random samples to the tree directly. Instead, adds a free configuration between the randomly sampled configuration and a nearest neighbour already in the tree determined by a state transition function and an input that minimizes the distance between the random and neighbouring configuration. Unlike the basic PRM that is intended for holonomic robots [168], basic RRT can incorporate nonholonomic constraints [142].

Performance evaluation of sampling-based path planners is never conclusive, one can hardly find persistent performance patterns. Even when patterns seem to exist, the performance boundaries of different algorithms seem to overlap. This makes it hard to choose one algorithm over another on average cases. It should also be noted that simple planners are preferred to complex ones if their performance is comparable [169], which is in agreement with the law of parsimony. Together, these observations led to the choice of working not with a single planner but multiple concurrent planners stack together to create an ensemble from which the best planner (first to return a feasible path or one with the shortest path) is selected at runtime.

5.1.1 Ensemble Planner

The ensemble is aimed at simulating repetitive planning trials with similar queries and uncontrollable free-configuration map building. Building this ensemble necessitates selecting candidate planner from a set of probabilistically complete planners that solve the feasibility problem as fast as possible, permitting tractable concurrent queries. Optimality did not make the list because optimal variants of RRT and PRM sampling-based path planners are over 30 times slower than their non-optimal variants [146].

From the preliminary planning speed experiments with PRM, RRT, PRM*, RRT*, Bidirectional Transition-based Rapidly-exploring Random Tree (BiT-RRT), LazyPRM, LazyRRT, RRT-Connect, Transition-based Rapidly-exploring Random Tree (T-RRT) and Path-Directed Subdivision Tree (PDST), on a multi-floor planning problem, the bidirectional planners BiT-RRT and RRT-Connect exhibited the shortest mean planning time, with BiT-RRT running the fastest. This is in agreement with [169, 138] who asserted that bidirectional tree expansion greatly improves on planner performance. Techniques like lazy collision checking and kd-tree nearest neighbour search, which improve collision checking speed and nearest neighbour search speed respectively [167], did not produce consistently observable runtime improvement as bidirectional exploration.

RRT-Connect, BiT-RRT and RRT were selected as ensemble planners. This choice was generally based on their implementation simplicity and rapid exploration capability. Furthermore, the former two scored the highest on the planning time scale, while choice of the latter was additionally influenced by the observation that for closer queries, the bidirectional trees may explore a large unnecessary area before they can connect, which degrades their bidirectional advantage [169]. Therefore, included the basic unidirectional RRT to establish an upper bound on planning time for cases where the two bidirectional planners lose their bidirectional advantage. The name ensemble was motivated by the fact that the selected planners are derivatives of the same planner, i.e., RRT. The concurrent ensemble is expected to improve on the success rate and planning time. Furthermore, RRTs are incremental as opposed to batch algorithms, hence single-query, while PRMs are multi-query (build a free space representation once during the learning phase and use it to answer multiple queries with the assumption of a fixed obstacle set [168]), a property which makes them unsuitable for dynamic environments.

The incremental nature of RRT is beneficial for online implementation [146]. Online problems do not require multiple queries since the environment is not fully known a priori [146] or may change over time. Therefore, it is best to work with the incremental online, single query RRTs as ensemble planners. Online and real-time capability of RRTs has already been established in [167], who extended RRT with waypoint caching to enable information reuse, which improved replanning and local minima avoidance, and applied it to a team of ground robots. Despite those advantages, biasing the path search towards a previously cached solution may hinder the possibility of finding a path that is orders of magnitude better than the previously cached path, if the two are not homotopic. On the contrary, the derivation process of the proposed planner opted for information gathering diversification by deploying multiple planners. The selected planners were stack together to create a non-interacting ensemble of path planners.

The idea of multiple sampling-based planners featured in [170], where RRT and modified versions of RRT were applied to quickly generate an initial solution to a query and successively improve on the path quality respectively, as long as time allowed. This method defers from the one presented here in that the latter considers concurrent planning and also focuses primarily on boosting success rate and planning time. The choice of fast path planners allowed time for path smoothing as a post-process. Next, are detailed descriptions of the selected ensemble constituent planners.

RRT: Rapidly-exploring random tree (RRT) [142] is an incremental sampling algorithm that samples the configuration space for free configurations $q \in Q_{free}$ to build a random tree capturing the connectivity of the free configuration space. It is a randomized variant of RDTs. While oblivious to the geometric complexities of the obstacle space, RRTs rely on collision checkers to eliminate invalid configurations. It encodes nonholonomic constraints in a state transition function $\dot{q} = f(q, u)$, where $u \in U$ is a set of possible inputs. The uniqueness of RRT comes from the fact that the new configuration state q_{new} is determined, not randomly selected. Initially, the tree contains only the initial state $q_{start} \in Q_{free}$. Then iteratively, a random configuration $q_{rand} \in Q_{free}$ is selected. The nearest neighbour search then looks up for the closest neighbour to q_{rand} among the tree vertices using a distance metric. Next, an input u that minimizes the distance between q_{rand} and q_{near}, while ensuring a non-colliding configuration is selected. Finally, a new configuration q_{new} determined by applying the selected input u to the state transition function is added to the tree. Added with it is its associated edge and input u. RRT bears interesting properties like probabilistic completeness, minimal, simplistic implementation, relatively fast planning and ease of incorporation of nonholonomic constraints [142], but it is not optimal [146] and allows no control over the path quality.

RRT-Connect: RRT-Connect is a single query randomized planner, which as the name suggests is a variant of RRT. In the original research work that presented RRT, the authors conjectured performance improvement by spawning two trees, one rooted at the start configuration and the other rooted at the goal configuration [142]. This conjecture is the basis for RRT-Connect. RRT-Connect constructs two trees rooted at the start configuration and goal configuration respectively, with a greedy heuristic that tries to connect the trees to each other [145]. This greedy heuristic biases the trees towards each other hence introducing some degree of informedness.

Unlike the original RRT, RRT-Connect assumes no differential constraints. The approach works as follows, two trees are spawned, one rooted at the start configuration and the other at the goal configuration. Then an arbitrary tree T_S is selected in which a nearest neighbour q_{near} to a randomly selected configuration $q_{rand} \in Q_{free}$ is searched. After the search, a sequence of new configurations $\{q_{new,i},\}$, $i \geq 1$ between q_{near} and q_{rand} are continuously sampled at an incremental distance ρ and added to T_S until an obstacle is encountered or q_{rand} is reached. Next, an attempt is made to connect the latest q_{new} to the second tree T_G. If the connection is successful, a path is found, else, the second tree T_G is selected for expansion and the process repeats. The attractive properties of this planner include an exact representation of the goal configuration, requires no pre-processing and is real-time planning capable, but it exhibits weak completeness and allow no control over the path quality.

BiT-RRT: BiT-RRT is derived from T-RRT, which is a cost-space planner derived from an RRT scheme that generates a cost optimal path relative to a cost function. T-RRT biases expansion towards low cost regions in the configuration space with a transition probability derived from mechanical work using an exponential decay mapping. Downhill expansions are accepted with probability one and uphill expansions are accepted with a transition probability that is derived from an exponential decay mapping of mechanical work done to move from current configuration to the new configuration. A parameter T (Temperature), dynamically controls the climbing ability of the expansion step, with higher temperatures enabling climbing and lower temperatures limiting the expansion to lower gradient regions. T also serves to overcome local minima traps. T-RRT aims to improve on both the planning speed and path quality, qualities that are inherited and improved upon by the bidirectional variant of T-RRT called BiT-RRT [171].

BiT-RRT grows two trees rooted at the start and goal configurations respectively. Unlike RRT-Connect, BiT-RRT only attempts to connect the two trees if the connection is of downhill slope and the two vertices are close to within a distance $\{d|d < 10 \times \delta\}$, where δ is the expansion step size. The connect heuristic in BiT-RRT and RRT-Connect improves the running time by a factor of three-to-four in uncluttered environments, but by less in cluttered environments [145]. It has also been observed that in cluttered environments, the two trees may expand into unnecessary regions before they can connect, than a single tree would [171], thereby losing their bidirectional advantage. This is the reason why the basic RRT was included in the ensemble, as it is easy to maintain over a small search area.

5.1.2 Ensemble Planner Implementation and Simulation

The individual planners were implemented in C++ using the Open Motion Planning Library (OMPL) [172] as independent executable processes. The ensemble logic that manipulated the individual planners was implemented in C# and deployed as a script in the Unity software framework. Planning requires user inputs including maximum planning time, path simplification setting, robot mesh, world mesh, start and goal configurations, which are provided through a settings XML file. Upon planning completion, each planner outputs a random sequence of configurations, path generation time and path simplification time into the settings XML file. The complete architecture is as indicated in Figure 5.1.

To evaluate the ensemble performance, two test strategies were conducted, (1) winner-take-all and (2) fixed-time-window. In the former strategy, all planners are run concurrently and the first to find a valid path is selected. Return of a valid path by the fastest planner triggers termination of the other planning processes. In the latter strategy, all planners are run concurrently for a fixed time lapse. Upon time expiration, plans from successful planners are collected, analysed and the shortest of them all is selected for execution. These paths are composed of straight-line segments between waypoints. The best path is suboptimal but has some properties of an optimal path, i.e., piece-wise smoothness and keeps a minimum clearance from obstacles.

Non-optimal sampling-based path planners have path quality problems, the paths are

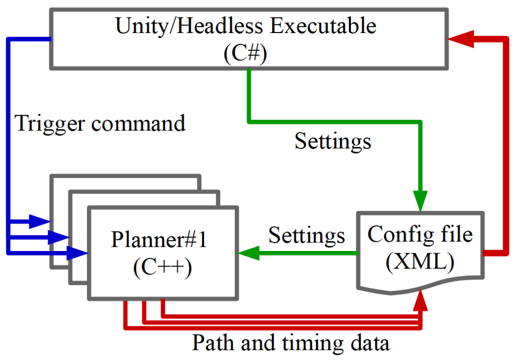


Figure 5.1: Ensemble architecture.

highly irregular. The choice of fast path planners as ensemble planners allows extra time for path smoothing, which is a crucial step towards path quality improvement for randomized sampling path planners. Therefore, in this project, the path planning time is the sum of path generation time and path simplification time. Fig. 5.2a and Fig. 5.2b show two example sampling paths for a start configuration S on the ground floor and goal configuration A on the first floor of an office building. From this, it is visually evident that the path quality is greatly improved by the path simplification post processing step.

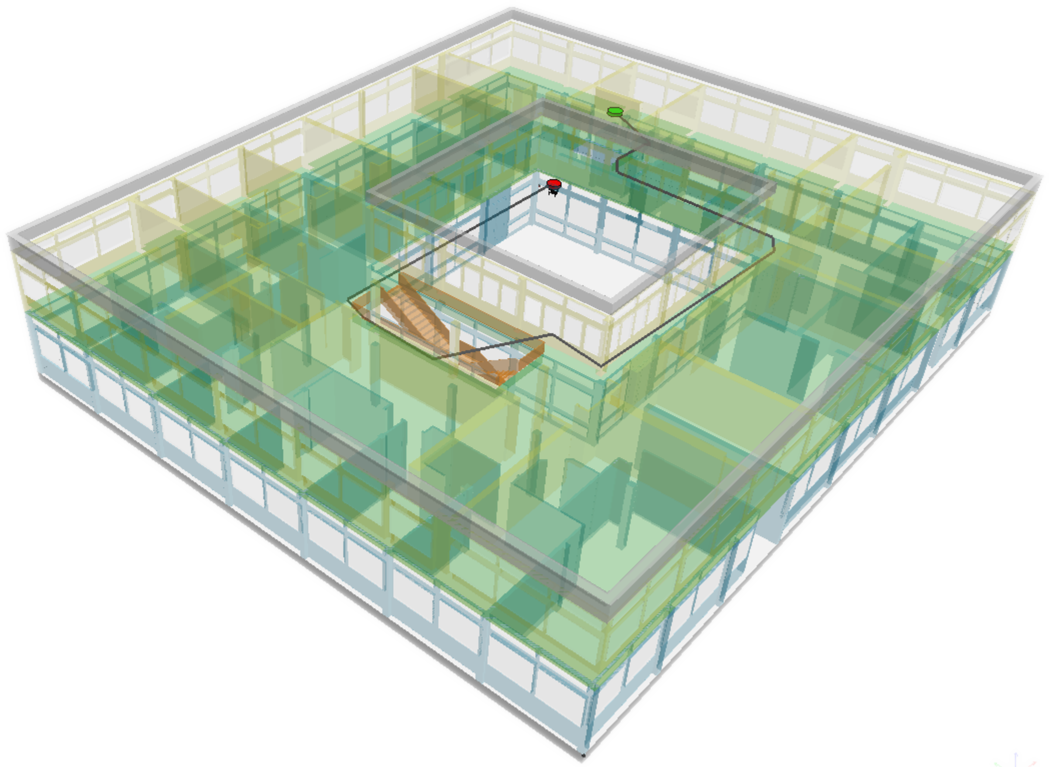
To test a path planner, an environmental representation is necessary. For this case, a two-floor office building, with only two points of entry between the floors is used . The size of the floors is 42.5 m $\times$ 37.20 m $\times$ 4.056 m and 42.5 m $\times$ 37.2 m $\times$ 3.885 m for the ground and first floor respectively. For planning purposes, the environment is represented as a 3D model with 16944 triangle primitives. Six queries $\{(q_S, q_A), (q_S, q_B), (q_S, q_C), (q_S, q_D), (q_S, q_E), (q_S, q_F)\}$ were arbitrarily selected for performance assessment of the ensemble planner. The target points are indicated by cubes $A - F$ and the start point by cube S in Fig. 5.3. Besides point A, all points are located on the ground floor. The coordinate values of all query points are presented in Table 5.1.

Table 5.1: Query coordinate values.

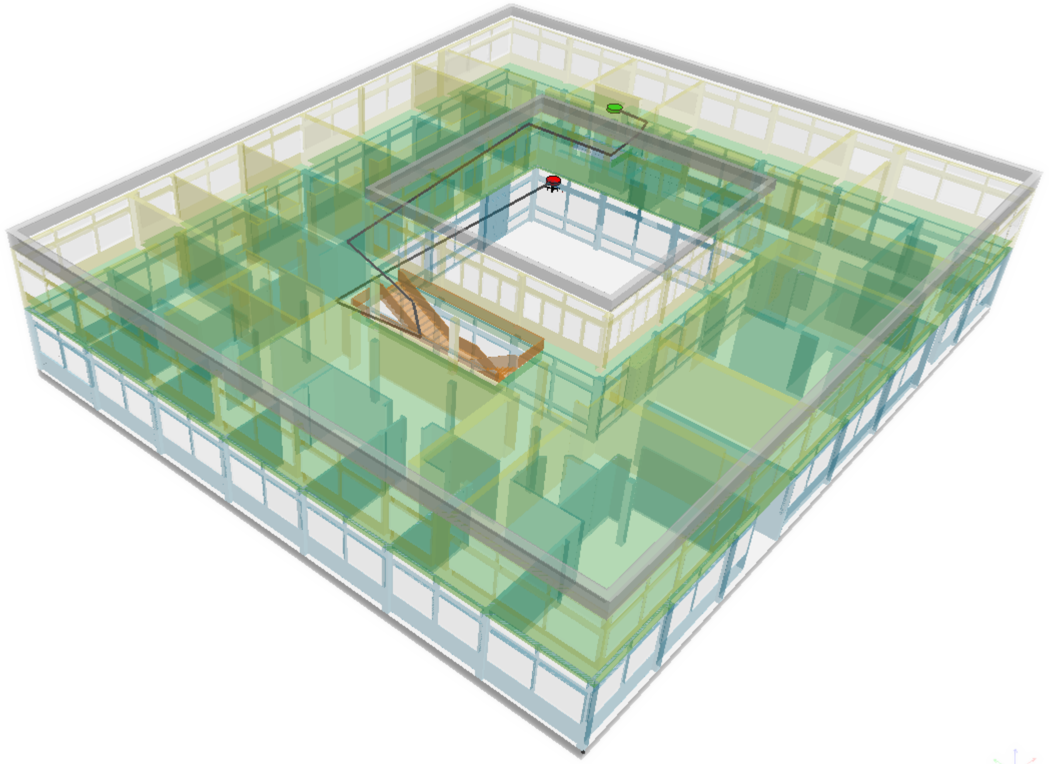
	S	A	B	C	D	E	F
$x(m)$	36.000	38.650	9.000	33.000	33.000	33.000	21.000
$y(m)$	27.800	25.325	27.800	25.000	27.800	2.000	27.800
$z(m)$	0.750	6.000	0.400	1.000	1.000	1.000	1.000

The two test strategies were tested in simulation on a personal computer with 16 GB of physical memory, 64-bit Intel® Core™ i7 processor with 2.60 GHz clock frequency and six cores, and a 6.0 GB Nvidia GeForce GTX 1660 Ti GPU (Graphics Processing Unit). The fixed-time-window strategy was conducted as a success rate test, where each planner attempted to answer each query 400 times within fixed time windows of duration $t \in [0.01, 60]$ seconds. The average success rates for each of the six queries are indicated in Fig. 5.4. In this figure, the query complexity decreases as one moves from left to right then top-down. It is evident that BiT-RRT has an exceptional success rate for all queries, followed by RRT-Connect and trailed by RRT.

Based on this, one might be tempted to choose a single planner e.g. BiT-RRT in this case as opposed to an ensemble, but this figure shows just half the story. Looking at each planner's planning time in Figure 5.5, it is evident that, while BiT-RRT dominates at complex queries, RRT dominates at simpler queries. The dominance of RRT is attributed to the ease with which a single tree is maintained. So, for closer queries in open spaces,



(a)



(b)

Figure 5.2: (a) and (b) are two different paths generated by the ensemble planner for a similar query on two different planning trials. The lower cylinder is the start configuration, while the upper cylinder is the goal configuration.

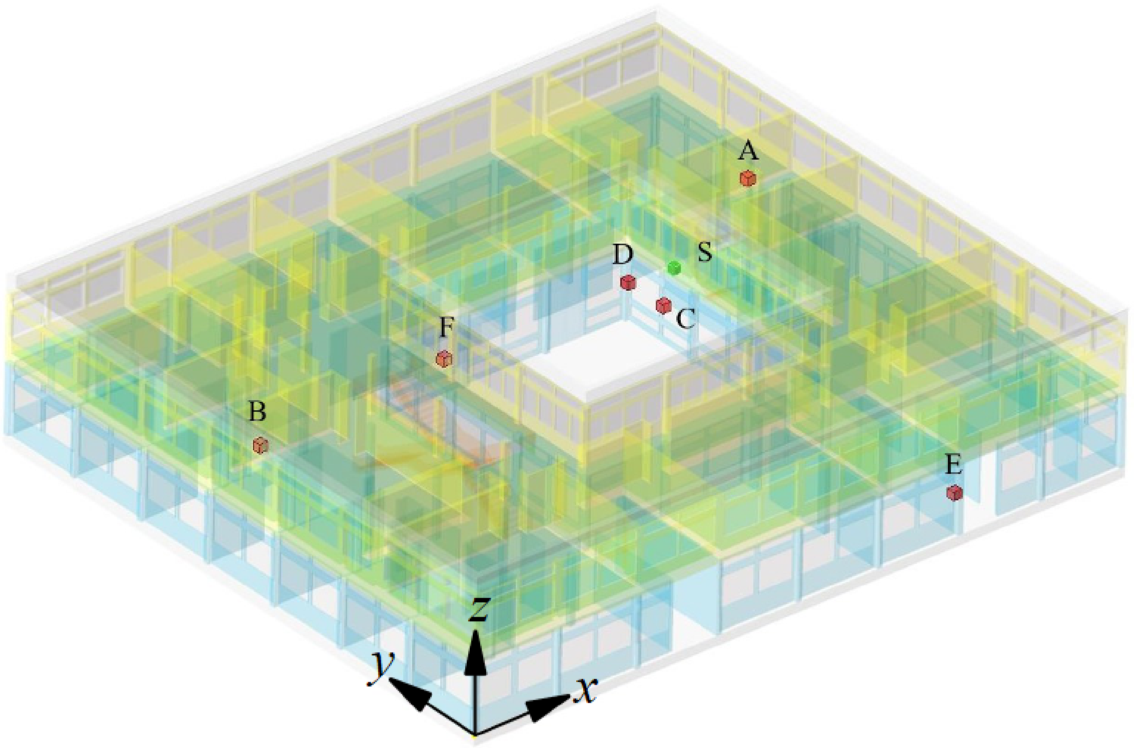


Figure 5.3: Test environment. The first floor contains only target A, while the ground floor contains targets $B - F$ and the start point S.

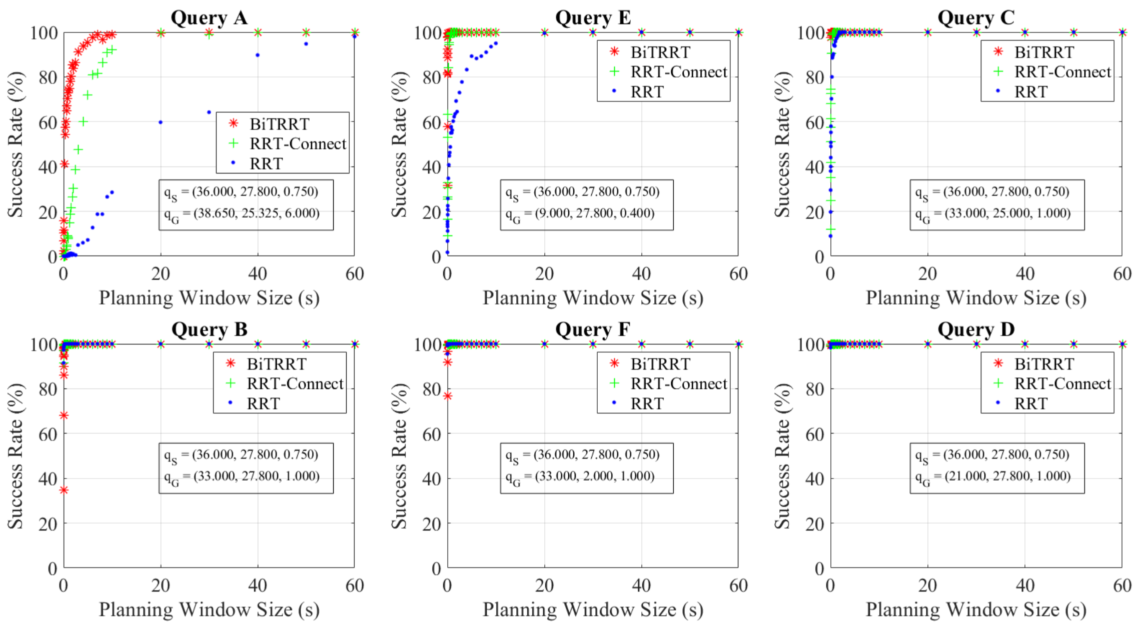


Figure 5.4: Success rate simulation results for the six queries.

the bidirectional planners sporadically lose their bidirectional advantage.

Next, is the analysis of the winner-take-all simulation test on the six queries. Unlike the previous test which involved a fixed time window, here the time is not restricted. The three planners are run concurrently and the first to find a valid solution to a query is returned whilst the other planning processes are terminated. For each query, the ensemble ran 10,000 trials. The resulting contribution from each planner is represented in Figure 5.6, in which a similar trend as for the fixed-time-window strategy emerged. BiT-RRT has a

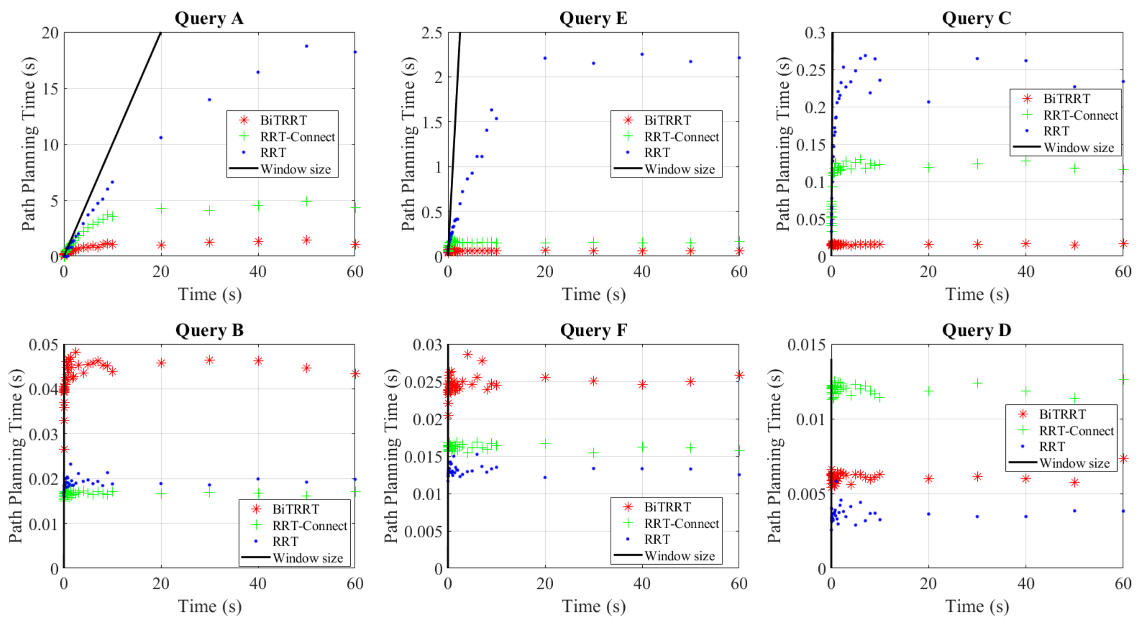


Figure 5.5: Runtime simulation results for the six queries.

probability $p_{BiTRRT}(SUCCESS) > 0.7$ of finding a valid path for complex queries, but this probability drops to $p_{BiTRRT}(SUCCESS) < 0.4$ for easier queries. Although the performance of RRT for complex queries is poor, it consistently exhibited a relatively higher probability of finding a valid solution to easier queries.

With these results, it can be concluded that an ensemble improves not only on the success rate of path planning, but also on the planning time and introduces a degree of adaptability which improves on the overall performance. These benefits are not without associated cost as running an ensemble of planners consumes relatively more computational resources, but the choice of fast randomized sampling-based planners as ensemble member planners bears the benefit of tractability and speed.

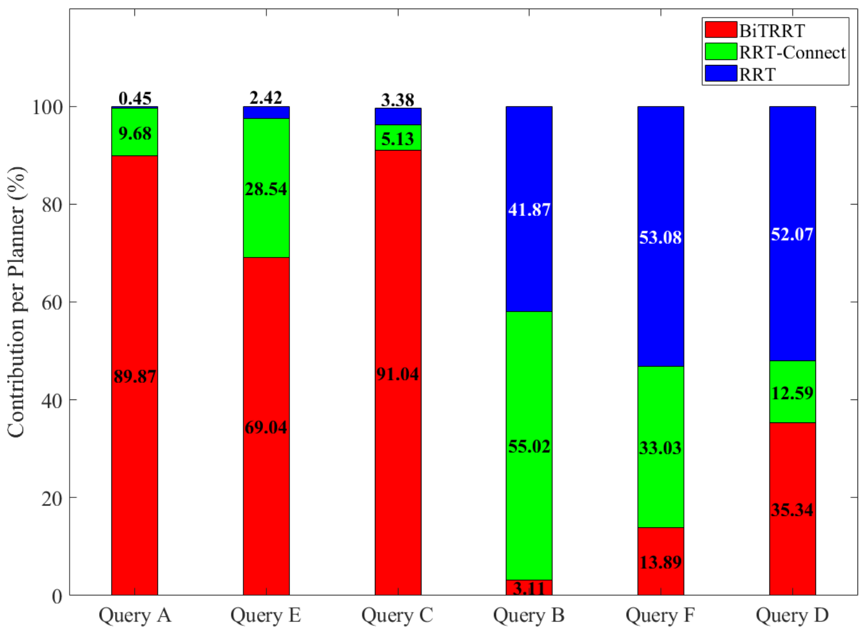


Figure 5.6: Percentage contribution of each planner in the ensemble evaluated over 10,000 trials per query.

Replanning Simulation

To test for the re-planning ability of the proposed ensemble planner, a new set of ten goal configurations was arbitrarily chosen. Unlike, in the planning tests, here the goal configurations are distributed equally among the two floors, i.e., five goal configurations on each floor as indicated in Fig. 5.7.

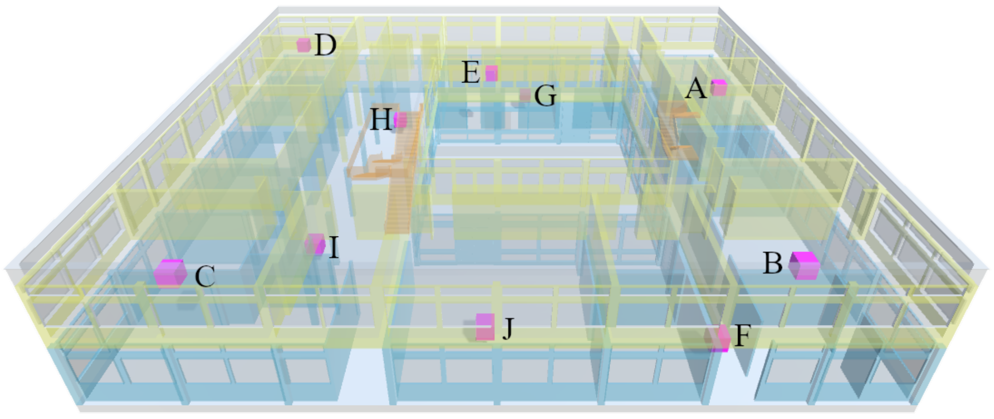


Figure 5.7: Replanning goal configurations. Goal configurations A-E are located on the first floor, while goal configurations F-J are located on the ground floor.

The replanning function was implemented on top of the planning ensemble, with the addition of an iterative replanning function and an external replanning trigger interface. The former iteratively calls the planning function until a valid path is found, while the latter allows for manual triggering of the replanning process. Upon triggering, one of the ten goal configurations A-J is randomly set as the goal configuration and then planning is executed. After incorporating these functions, the replanning module was tested.

Two hundred replanning trials were executed on the same computing machine as used for the planning simulation tests and the replanning times recorded. Again, as mentioned in the previous section, the planning/replanning time is the sum of path search time and path smoothing time. The resulting path replanning time Gaussian distribution is showed in Fig. 5.8. From this distribution, the expected replanning time is 1.34 seconds with a standard deviation of 0.80 seconds , which is sufficient for online replanning.

5.2 Aerial Coverage Path Planning

Coverage Path Planning (CPP) is a type of path planning where the generated path ensures the robot footprint or sensor field-of-view (FoV) covers all open spaces in the Area of Interest (AOI). It differs from start-to-goal path planning where a path from a start point to a goal is sought. CPP algorithms find applications in mobile ground robotics like vacuum cleaning, lawn mowing, security and surveillance, de-icing of airports, and harvesting or seeding, among others, and mobile aerial robotics like crop sensing, geological documentation, urban planning, wetland management, search and rescue, land-use monitoring, mapping and remote sensing among others.

Literature has many elegant solutions to the CPP problem. Unfortunately, most of these solutions are specific to mobile ground robotic applications and do not scale optimally to aerial robotics. This limited generalizability together with advances in aerial robotics and remote sensing have fuelled intense research in the field of Aerial Coverage Path Planning (ACPP), which is the subject of this section. Large-scale coverage industrial applications often-times exceed the single flight coverage capability of modern multirotor

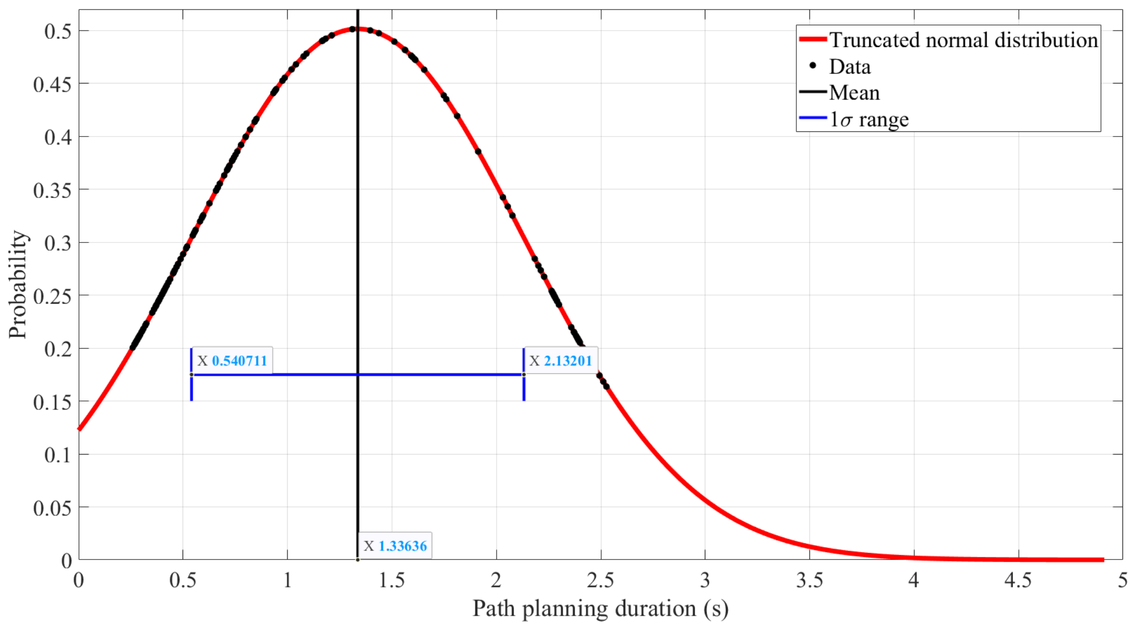


Figure 5.8: Replanning time Gaussian distribution. The mean replanning time is 1.34 s and replanning time standard deviation is 0.80 s.

MAVs. Luckily, dropping prices have enabled acquisition of multiple platforms, whose aggregate capability can easily satisfy most of these large-scale coverage applications. It should be noted that even with a single platform, multiple flights can be systematically conducted in a short period of time, owing to manoeuvrability and low operating costs of these aerial platforms. To harness the cumulative power in numbers, partitioning schemes are necessary for systematically partitioning large areas into manageable portions and assigning them to a fleet of MAVs or flying them with one platform multiple times. Such a planner is key to autonomous robotics because it guarantees coverage, enables proper task planning, impact assessment of operational costs as well as management of hardware and human resources.

When designing coverage path planners, up to four factors can be considered namely, environment, vehicle, coverage actuator/sensor and algorithm. Table 5.2 highlights the properties underlying each of the factors. Literature contains CPP algorithms based on one or a combination thereof. Interested readers can find more details on characterization of CPP methods in [173], a survey article on coverage path planning in robotics.

Table 5.2: Design factors for coverage path planners.

Factor	**Properties**
Environment	Static/dynamic, 2D/3D/2.5D, non-differentiable, size
Vehicle	Aerial/ground/amphibian, unmanned, holonomic/nonholonomic, finite energy storage/infinite energy storage, single robot/multiple robots
Coverage sensor/actuator	Footprint shape, mounting (Gimbal or no gimbal)
Algorithm	Offline/online, completeness, complexity, optimality (with respect to number of turns, path length, time-to-completion, coverage area per distance travelled or energy)

For complete coverage, spiral in Fig.5.9a and boustrophedon in Fig.5.9b are the most commonly used flight path patterns for both fixed-wing and multirotor UAVs. An empirical performance assessment based on three metrics namely, energy, time and distance showed that spiral trajectories were more suitable for fixed wing UAVs, whereas boustrophedon trajectories for multirotor UAVs [174]. Based on this conclusion, boustrophedon path patterns are applied in this study for coverage path planning, since the deployment platforms of interest are multirotor MAVs. But patterns alone do not solve the large-scale problem, where the area of interest exceeds the coverage capability of the deployed vehicles. Therefore, this section introduces a coverage path planner with large-scale coverage capability, with focus on large-scale photogrammetric applications.

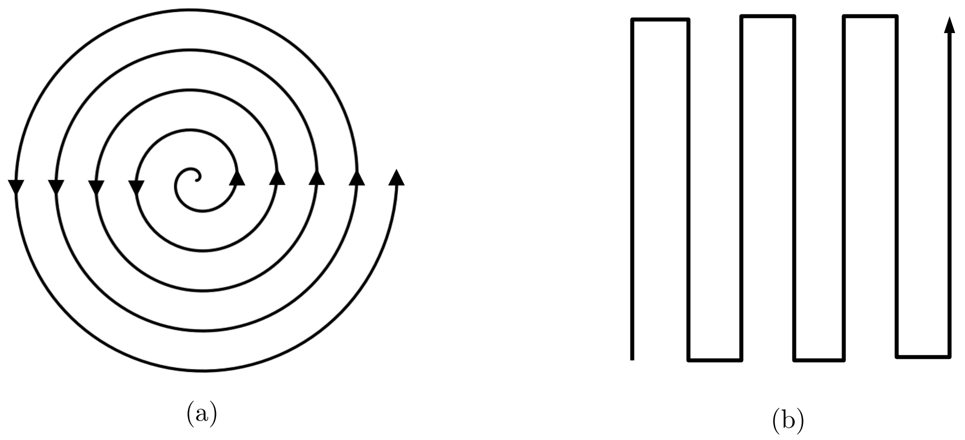


Figure 5.9: (a) Spiral path pattern. (b) Boustrophedon path pattern.

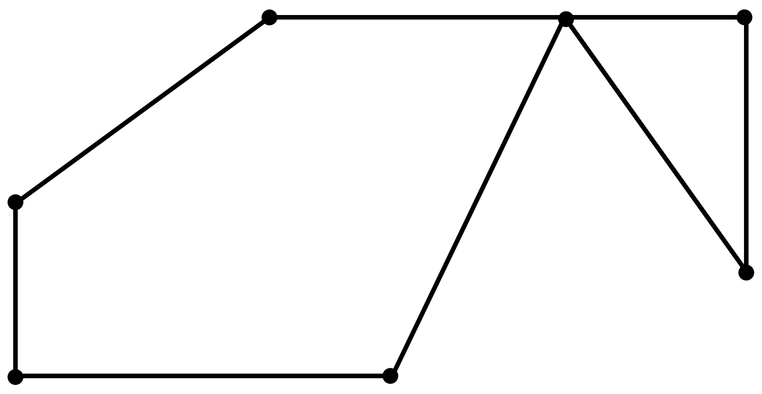
Described herein is a large-scale aerial coverage path planner for multi-rotor type MAVs. The planner is capable of planning coverage paths for a single platform and homogeneous or heterogeneous multi-rotor fleets. The proposed approach analyses the AOI, then non-deterministically partitions it into manageable portions with respect to maximum available endurance and assigns each cell the most suitable MAV (with minimum coverage time). The proposed approach accounts for the home point and MAV endurance to ensure vehicle recovery, i.e., platforms do not travel further than their endurance can support, guarantees complete coverage and resolution, but does not guarantee optimality. Guarantee of coverage is through exact cellular decomposition of AOI, and designing paths that ensure complete coverage of each cell.

The path planning process consists of three steps namely, input area preparation, area partitioning and coverage path planning. Unlike start-to-goal planners, which input a start and goal configuration, coverage planners input a set of vertices defining the area of interest. The resulting input area could be convex, concave or complex in geometry.

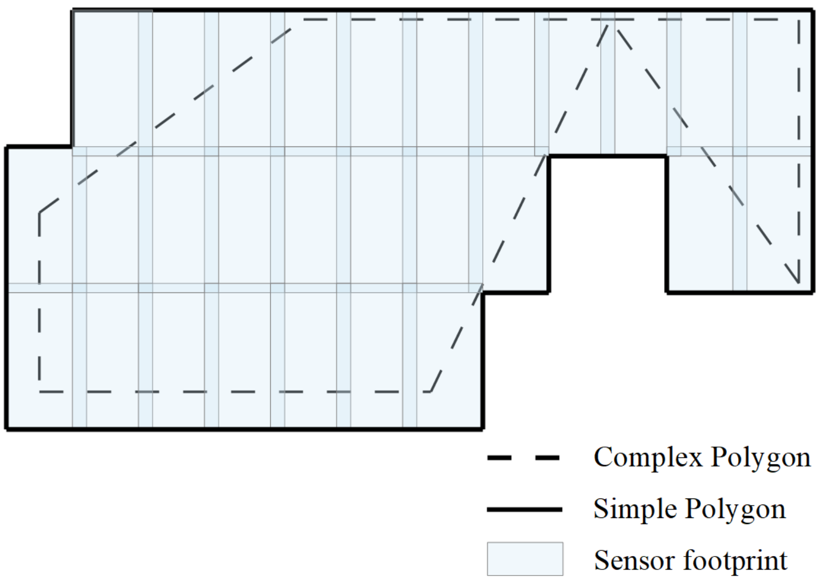
5.2.1 Area Preparation

The planner presented here inputs an unordered set of vertices, which as mentioned before could result in complex or simple polygonal areas. Operating on complex or non-convex input areas adds unnecessary levels of complexity and needs to be avoided whenever possible by transforming such areas into convex areas. Since coverage sensors or actuators have a non-zero footprint, a composition of such footprints is incapable of exact representation of complex input areas. An illustration of this can be seen in Fig. 5.10, where the complex input area in Fig. 5.10a is transformed into a simple polygon by a composition of sensor footprints in Fig. 5.10b. This justifies the transformation of complex input areas into simpler geometries.

To simplify the input geometry, two steps are implemented namely, sorting and convex hull approximation. During sorting, vertices are first transformed into polar coordinates



(a)



(b)

Figure 5.10: (a) Input area of interest as a complex polygon. (b) Complex area of interest automatically transformed into a simple polygon by sensor footprint coverage.

and then sorted in a counter-clockwise manner with ties broken through radii comparison. If the resulting geometry is non-convex, it is simplified further to its equivalent convex hull using Quickhull algorithm [175].

5.2.2 Area Partitioning

For large-scale applications, the area of interest normally exceeds the coverage capability of a single MAV, hence necessitating partitioning of the input area into partitions coverable by a single MAV or multiple MAVs in one or multiple flights.

To ascertain the need for partitioning, the two conditions in Eq. 5.1 must be satisfied.

$$\begin{gathered} T_C > T_E \\ \frac{D_{BB}}{60 \times v_N} > T_E \end{gathered} \tag{5.1}$$

where D_{BB} is the perimeter of an oriented bounding box around the area of interest, v_N is the nominal ground speed of the vehicle, T_E is platform endurance in minutes and T_C is coverage time in minutes.

After ascertaining the necessity for partitioning, all that remains is determining the number of partitions. The number of partitions n_P is given by Eq. 5.2.

$$n_P = \text{ceil}\left(\frac{A_T}{A_N}\right) = \text{ceil}\left(\frac{16 \times A_T}{D_N^2}\right), n_P \in \mathbb{Z} \tag{5.2}$$

where A_T is the area of interest, A_N is the maximum coverable area by the vehicle in time T_E and D_N is its nominal coverage distance, product of nominal speed and endurance.

Since the area is too large to cover from the current take-off point, n_p new take-off points (discarding the take-off point used in the initial test) are generated. These are the sites upon which partitioning is based using Voronoi decomposition, where cells are generated based on Euclidean distance from the sites. The final partitions are the regions of intersection between Voronoi cells and AOI, resulting in an exact cellular decomposition satisfying Eq. 5.3,

$$AOI = \sum_{i=1}^{n_P} A_i \tag{5.3}$$

where A_i is the area of partition i. It should be noted that site placement plays a key role in partitioning. For better results, the sites should be distributed spatially evenly within and/or around the AOI. The recommended locations for sites are regions near vertices of intersection between bounding box and AOI. In Eq. 5.1, If only $T_C > T_E$ is true, no area partitioning is necessary, but path splitting.

5.2.3 Coverage Path Planning

This step operates on the output of the area partitioning step, which is a partitioned area of interest. In so doing, it generates coverage paths for each of the partitions. The problem at hand is that of aerial mapping with photogrammetry, in which the interest is in moving the imaging sensor to specific locations regardless of the overall area geometry. This reduces the path planning problem to visiting a set of waypoints as opposed to planning at the geometric level. This transformed problem is solvable as a graph traversal problem or as in this case, with boustrophedon coverage paths. These patterns consist of equally spaced parallel line segments connected to their neighbours at the front and back ends by perpendicular line segments.

To achieve optimality with respect to number of turns, number of traversal lines and coverage distance, an optimal orientation for the traversal lines need to be determined. This is known as the sweep direction. The number of turns is proportional to number of traversal lines. Therefore, minimizing number of turns minimizes coverage distance and completion time.

Approaches for sweep line direction determination include orientation of the longest edge of input polygon [176, 177], longest edge of an axis aligned minimum area-bounding box, longest edge of an oriented minimum area bounding box and principle direction of variation of convex hull vertices. Aligning traversal lines parallel to the longest edge of an oriented minimum area bounding box is the only method proven to yield optimal number of turns [178]. This assertion has been empirically ascertained by monitoring the variations in coverage path length with respect to path orientation for a sample cell indicated in Figure.5.11. Figure. 5.12 shows the variation in total path length as the sweep line orientation is varied through a 180° rotation. Such characteristic is typical of a diameter function with global minima corresponding to the orientation of the longest edge of an oriented minimum area bounding box, and hence optimal sweep line orientation.

Now that the sweep line direction is known and the fact that the line length is determined by the cell boundary, the remaining question is "How spaced apart should the sweep

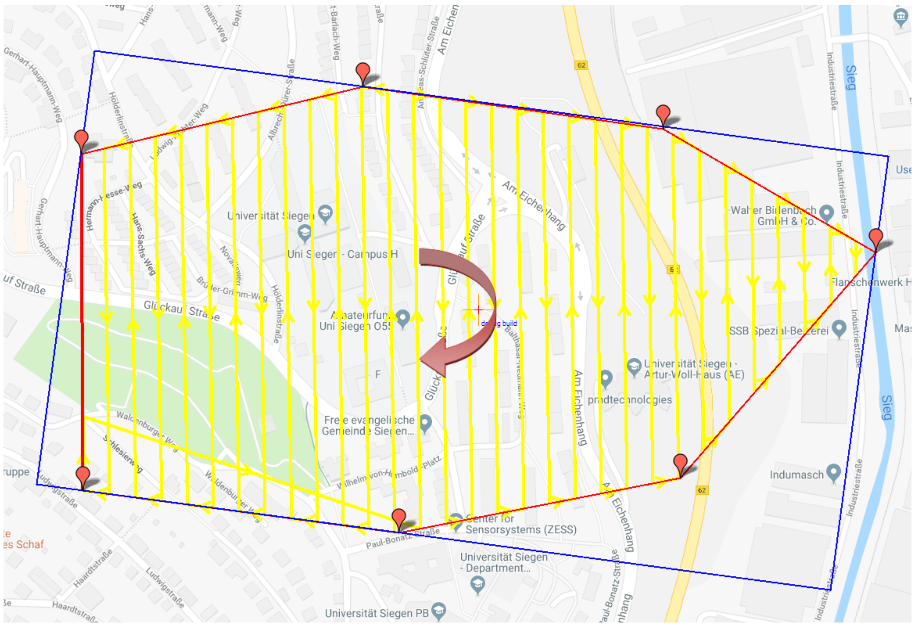


Figure 5.11: Sweep lines covering the AOI. The outer rectangle represents a minimum area oriented bounding box for the associated input area. The vertical pattern is a sample path with a 90° sweep orientation. Angular measurements are based on a left-hand convention.

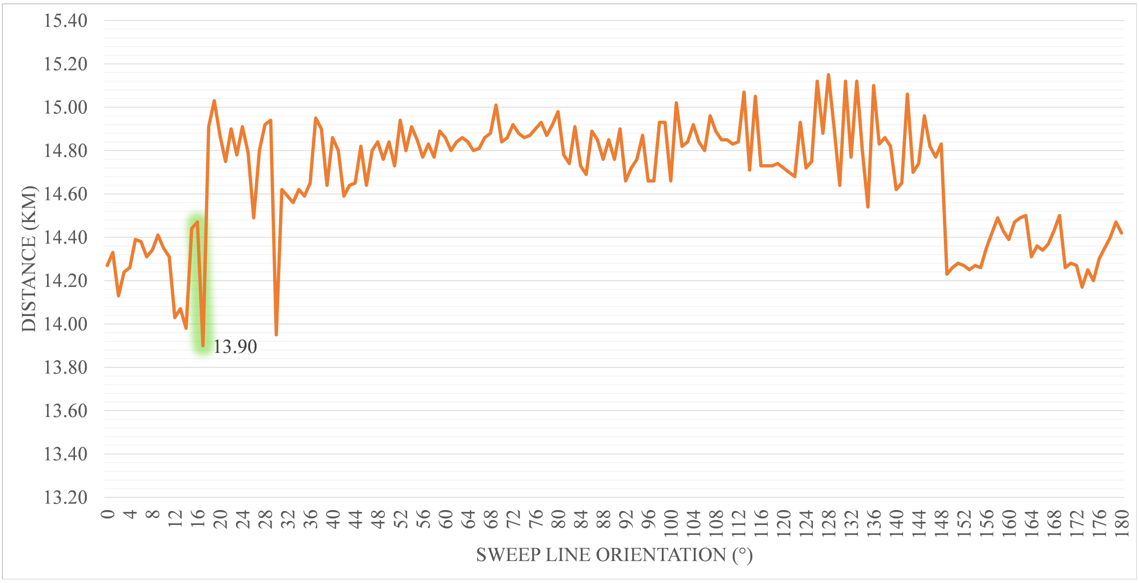


Figure 5.12: Changes in total path length as a function of sweep line orientation. Minimum path length occurs at an angle of 17°, marked by the shaded region, which corresponds to the orientation of the longest edge of the minimum area bounding box in Fig. 5.11.

lines be?” this information is specified by the application, with common examples including spraying and photogrammetry applications, with spray nozzle footprint and ground sample distance as the determinant properties respectively. These properties determine *AGL* and sweep line spacing. Continuing with photogrammetry, this application requires the following specifications:

- Camera specifications: focal length f, pixel pitch p, pixel count $(m \times n)$ pixels and shutter speed T_S.
- MAV specifications: ground nominal speed v_N, endurance T_E.

- Task specifications: desired Ground Sample Distance (GSD), forward overlap f_{OVLP}, side overlap s_{OVLP}, AOI.

From these specifications, the following dependent parameters are calculated:

- Above Ground Level (AGL)

$$AGL = \frac{GSD \times f}{p}$$

- Image size on the ground

$$D_w \times D_h = GSD \cdot m \times GSD \cdot n = GSD \cdot (m \times n)$$

- Side gain

$$s_{gain} = D_w \frac{100 - s_{OVLP}}{100}$$

- Forward gain

$$f_{gain} = D_h \frac{100 - f_{OVLP}}{100}$$

- Number of flight lines (NFL)

$$NFL = ceil\left(\frac{b}{s_{gain}} + 1\right)$$

  where b is the diameter of AOI.

- Number of images (NIM)

$$NIM = NFL + ceil\left(\sum_{i=1}^{NFL} \frac{I_i}{f_{gain}}\right)$$

  where I_i is the length of flight line i.

- Camera trigger rate

$$T_T = \frac{f_{gain}}{v_N}, \text{ subject to } T_T \geq T_S$$

The generated coverage paths for each cell constitute a series of parallel traversal lines oriented towards the optimal sweep direction. The generated coverage path may exceed the nominal endurance of the available aerial platforms, in which case a splitter function automatically divides the path further into manageable path segments cyclically connected to the take-off point. This approach is directly generalizable to multiple homogeneous MAVs. Application to heterogeneous MAVs requires special analysis. To accommodate the variation in platform capabilities, the partitioning decision in Eq. 5.1 is instead based on minimum endurance MAV. This ensures accessibility to the furthest points in the AOI by all MAVs. With this modification, the first partitioning condition in Eq. 5.1 becomes,

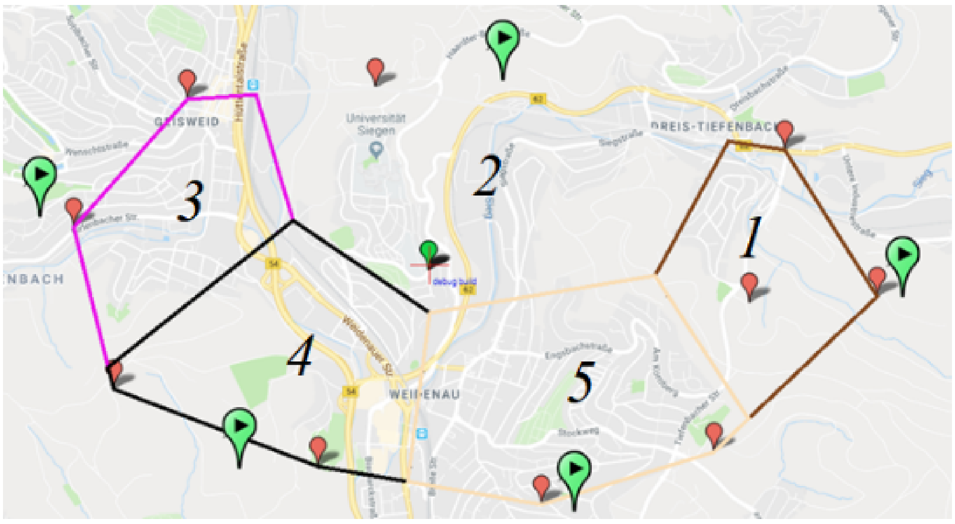
$$\frac{D_{BB}}{60 \times v_{N,E_{min}}} > T_{E_{min}} \tag{5.4}$$

where $v_{N,E_{min}}$ and $T_{E_{min}}$ are the nominal ground speed and endurance of the MAV with the shortest endurance respectively. Then for each platform i, a coverage path is planned on each cell j, resulting in coverage time $T_{C,ij}$ and number of flights $n_{F,j}$. A scheduler then allocates cells to available quadcopters with priority given to high endurance quadcopters and longest coverage time cells.

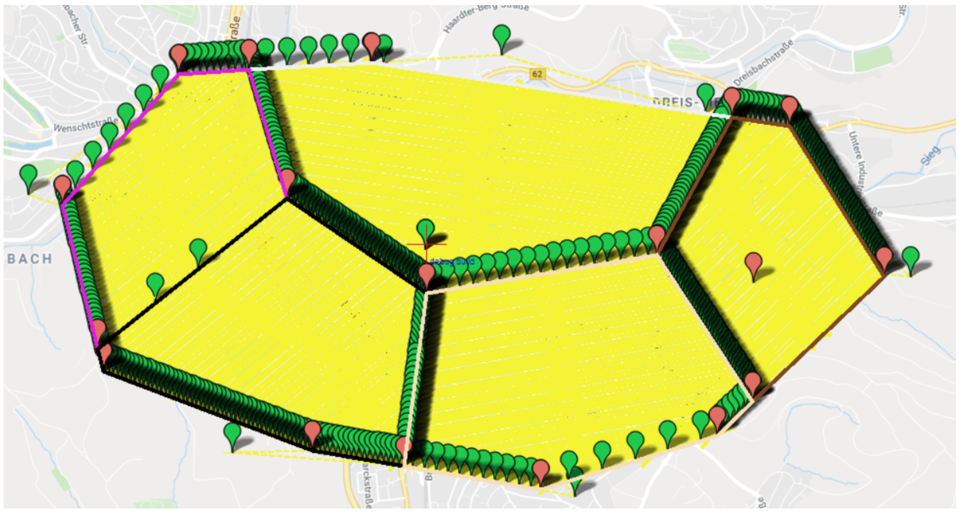
5.2.4 Simulation Results

The large-scale coverage planner presented in this section was incorporated into a C# based Ardupilot Mission Planner software package to take advantage of its readily available open-source functions. This software package also supports Software-In-the-Loop (SIL) testing for ground and aerial vehicles. Testing the proposed partitioning planner employed a DJI Matrice 100 quadcopter with endurance $T_E = 20$ minutes and nominal (minimum camera vibration) speed $v_N = 5$ m/s, and md4-1000 quadcopter with endurance $T_E = 45$ minutes, nominal speed 6 m/s, a gimbal stabilized Sony nex-5 camera with focal length 25 mm, image size of 4595×3056 pixels, $p = 5.07$ µm, sensor size of 23.5 mm $\times$ 15.6 mm, flight altitude of 100 m, giving a ground resolution of 2.03 cm. The overlaps were set to 50% and 60% for forward and side overlaps respectively. On analysing the input AOI, $A_T = 9838175$ m^2, $D_{BB} = 12.6$ km, and $n_P = 5$.

Fig. 5.13a shows the input area and Fig. 5.13b the resultant coverage paths for each cell. The details for each of the cells are made available in Table 5.3.



(a)



(b)

Figure 5.13: (a) AOI partitioned into five cells corresponding to the five home points (Voronoi sites). The biggest pins indicate home positions, while the others delimit the AOI. (b) Complete coverage paths for each of the five cells.

Table 5.3: Results of AOI partitioning and coverage path planning.

Cells	**Cell 1**	**Cell 2**	**Cell 3**	**Cell 4**	**Cell 5**
Area (m^2)	1372337	2895632	1339350	1996098	2235851
Perimeter (m)	4500	5000	3900	4700	4800
Path length (m)	61660	123220	60160	86900	93520
DJI Matrice 100 T_C (min)/n_F	256/13	513/26	250/13	**362/19**	**389/20**
md4-1000 T_C (min)/n_F	**214/5**	**421/10**	**208/5**	301/7	324/8

For the homogeneous case, only DJI Matrice 100 quadcopter was considered. Since the cell coverage paths exceeded the platform endurance, the paths were automatically split further into segments matching the quadcopter's nominal coverage, which was 6000 m and output as waypoint files. SIL simulation tests with the generated path files ascertained the feasibility and deployability of this conceptualized path planning framework. For the heterogeneous case, DJI Matrice 100 and md4-1000 were considered. Each platform planned a coverage path for each cell. The resulting completion times are tabulated in Table 5.3. The cells were then scheduled on the quadcopters with priority given to high coverage quadcopters and longest completion time cells. Cells 2, 1 and 3 were assigned to md4-1000, and cells 4 and 5 to DJI Matrice 100 as per the scheduler in Fig. 5.14. The total completion time and total number of flights are 843 minutes and 20, 751 minutes and 39 for md4-1000 and DJI Matrice 100 respectively. The assumptions upon which the planner is based, which include perfect waypoint tracking, constant endurance and zero environmental influence do not hold in the real world and may lead to performance degradation and gaps in generated photogrammetric maps.

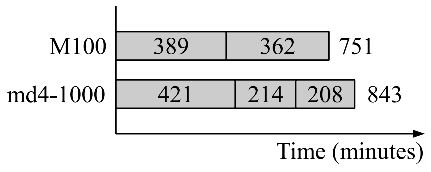


Figure 5.14: Partitions scheduler on md4-1000 and DJI Matrice 100 quadcopters.

5.3 Chapter Summary

This chapter introduces two solutions to two existing problems: (1) a solution to the problem of fast online path planning in higher dimensional and large spaces and (2) a solution to the offline large-scale aerial coverage path planning problem. The former solution is an ensemble of three concurrently executed RRT path planners with two output modes namely winner-take-all and fixed-time-window modes. In the former mode, the three planners are run concurrently and the first to return a valid solution triggers termination of all pending path searching processes. In the latter mode, all planners are run concurrently for a fixed time window. Upon time expiration, all successful planner solutions are checked for the shortest solution, which is then returned. In Sect. 5.2.3, a description of an offline coverage path planner for large-scale aerial mapping is presented. On ascertaining the partitioning necessity, the planner performs an exact cellular decomposition of the area of interest with user specified seeds, then determines the optimal sweep line direction and transversal lines spacing for each cell. And finally uses this information to generate Boustrophedon coverage paths for each of the cells and outputs them as waypoint files.

Chapter 6

Localization

Localization answers the question "Where am I?", which enables a robotic system to estimate its position at all time. For MAVs, this is one of the most important capability as it supports almost all their conceivable applications. A number of localization approaches have been studied in this work, including inertial, radio and vision techniques. This chapter describes the implementation and performance of each of these methods and/or combinations thereof.

6.1 UWB-Aided Inertial Localization

The approach demonstrated here loosely couples inertial measurements with UWB position measurements in a state estimator for vehicle state estimation. The selection process of the state estimator is described in the subsequent paragraphs, but important to note is that this choice depends mainly on the nature of the system model. In this MAV study project, the system model in Eq. 3.6 consisting of three force equations, three navigation equations, three moment equations and three kinematic equations is applied, in which the non-linearities and coupling are evident. This model assumes symmetry in the frontal and sagittal planes of the vehicle, as shown in Figure. 3.2.

For such a non-linear system, the common choices of state estimators include EKF and Unscented Kalman Filter (UKF). For this study, UKF was chosen for its ability to determine the propagated mean to third-order accuracy for Gaussian inputs, and ease of implementation [179].

UKF is a sample-based state estimator that estimates the first two moments of a Gaussian state distribution propagated through a non-linear model. UKF algorithm follows a recursive predict-update cycle. During the predict phase, the estimator deterministically samples $2n + 1$ (n is the state vector dimension, i.e., $\underline{x} \in \mathbb{R}^n$) sigma vectors $\underline{\mathcal{X}}_i \in \mathbb{R}^n$, $i = 1, ..., 2n + 1$ from a state distribution, propagates them through a discrete-time dynamic model $\underline{\mathcal{Y}} = f(\underline{\mathcal{X}})$ derived from Eq. 3.6 and estimates the first two moments as weighted sums using the Unscented Transform (UT). For the system at hand, the state vector $\underline{x} \in \mathbb{R}^{12}$ constitutes of three translational velocities, three angular velocities, orientation angles and 3D position, with mean $\overline{\underline{x}}$ and covariance $\underline{P}_x \in \mathbb{R}^{12\times 12}$. The methods for sigma vectors and weights calculation presented in [179] are adopted, where the sigma vectors are calculated as indicated in Eq. 6.1, the mean weights are calculated as indicated in Eq. 6.2, the covariance weights are calculated as indicated in Eq. 6.3, while the mean and covariance of the prior are estimated by the unscented transformation $\overline{\underline{y}}, \underline{P}_y = UT(\underline{\mathcal{Y}}, \underline{W}^m, \underline{W}^c, \underline{Q})$ expressed in Eq. 6.4.

$$\left.\begin{aligned} &\underline{\mathcal{X}}_0 = \underline{\overline{x}} \\ &\underline{\mathcal{X}}_i = \underline{\overline{x}} + \left(\sqrt{(n+\lambda)\,\underline{P}_x}\right)_i \qquad i = 1,...,n \\ &\underline{\mathcal{X}}_i = \underline{\overline{x}} - \left(\sqrt{(n+\lambda)\,\underline{P}_x}\right)_i \qquad i = n+1,...,2n \end{aligned}\right\} \tag{6.1}$$

$$\left.\begin{aligned} &W_0^m = \frac{\lambda}{n+\lambda} \\ &W_i^m = \frac{1}{2\,(n+\lambda)} \qquad i = 1,...,2n \end{aligned}\right\} \tag{6.2}$$

$$\left.\begin{aligned} &W_0^c = \frac{\lambda}{n+\lambda} + \left(1-\alpha^2+\beta\right) \\ &W_i^c = \frac{1}{2\,(n+\lambda)} \qquad i = 1,...,2n \end{aligned}\right\} \tag{6.3}$$

$$\left.\begin{aligned} &\underline{\overline{y}} = \sum_{i=0}^{2n} W_i^m \underline{\mathcal{Y}}_i \\ &\underline{P}_y = \sum_{i=0}^{2n} W_i^c (\underline{\mathcal{Y}}_i - \underline{\overline{y}})(\underline{\mathcal{Y}}_i - \underline{\overline{y}})^T + \underline{Q} \end{aligned}\right\} \tag{6.4}$$

where $\lambda = \alpha^2\,(n+\kappa) - n$ and $\underline{Q}$ is process noise, which is assumed to be additive white noise in this study. Details on the implications of the λ expression parameters and their recommended values can be found in [179].

During the update phase, the transformed sigma vectors are projected into the measurement space using the measurement function $\underline{\mathcal{Z}} = h(\underline{\mathcal{Y}})$ and applied through the unscented transform $\underline{\overline{z}}, \underline{P}_z = UT(\underline{\mathcal{Z}}, \underline{W}^m, \underline{W}^c, \underline{R})$ expressed in Eq. 6.5 to yield measurement mean and covariance.

$$\left.\begin{aligned} &\underline{\overline{z}} = \sum_{i=0}^{2n} W_i^m \underline{\mathcal{Z}}_i \\ &P_z = \sum_{i=0}^{2n} W_i^c (\underline{\mathcal{Z}}_i - \underline{\overline{z}})(\underline{\mathcal{Z}}_i - \underline{\overline{z}})^T + \underline{R} \end{aligned}\right\} \tag{6.5}$$

where $\underline{R}$ is measurement noise.

Finally, the posterior is estimated by the pair of equations presented in Eq. 6.6.

$$\left.\begin{aligned} &\underline{\overline{x}} = \underline{\overline{y}} + \underline{K}(\underline{z} - \underline{\overline{z}}) \\ &\underline{P}_x = \underline{P}_y - \underline{K}\ \underline{P}_z \underline{K}^T \end{aligned}\right\} \tag{6.6}$$

where $\underline{K} = \underline{P}_{xz}\underline{P}_z^{-1}$ with $\underline{P}_{xz} = \sum_{i=0}^{2n} W_i^c (\underline{\mathcal{Y}}_i - \underline{\overline{y}})(\underline{\mathcal{Z}}_i - \underline{\overline{z}})^T$.

This UKF recursively tracks the evolution of a system state through a predict-update loop, where a prior state is predicted by temporal propagation through the system model using the classic Runge–Kutta ODE solver. Then the prior is updated with IMU attitude and UWB position measurements to yield a state posterior containing twelve states of which six are of localization interest, i.e., three position coordinates and three attitude angles. Then the process is repeated at every sample time. The UWB radio positioning system applied in this study is a Pozyx [1] indoor positioning system.

[1]https://pozyx.io/

6.1.1 Simulation and Testing

This localization framework has been tested in simulation to assess its performance. The simulation setup implemented the hardware simulation models presented in Sect. 7.1 in Unity software framework. To allow for a qualitative performance assessment, a path tracking task was chosen. The path to track is generated using the ensemble path planner of Sect. 5.1.1. Fig. 6.4a shows a sample path generated by the ensemble planner for localization testing.

6.1.2 Hardware Implementation

To operate in an environment with obstacles, mobile robots like MAVs must have the capability of perceiving and avoiding these obstacles, both static and dynamic . For this purpose, the deployed vehicle is equipped with four ultrasonic range-finders. With the platform being of ×-configuration, the range-finders are mounted facing forward, left, backward and right respectively. An additional downward facing range-finder provides AGL measurements. The five sensors and their relative locations are indicated in Figure 6.1. It should be noted that the five stereo-vision modules integrated with the ultrasonic range-finders are not used in this study due to the vulnerabilities associated with stereo-vision. A GNSS-receiver is also available onboard but serves no sensing purpose since the experiment is conducted indoors.

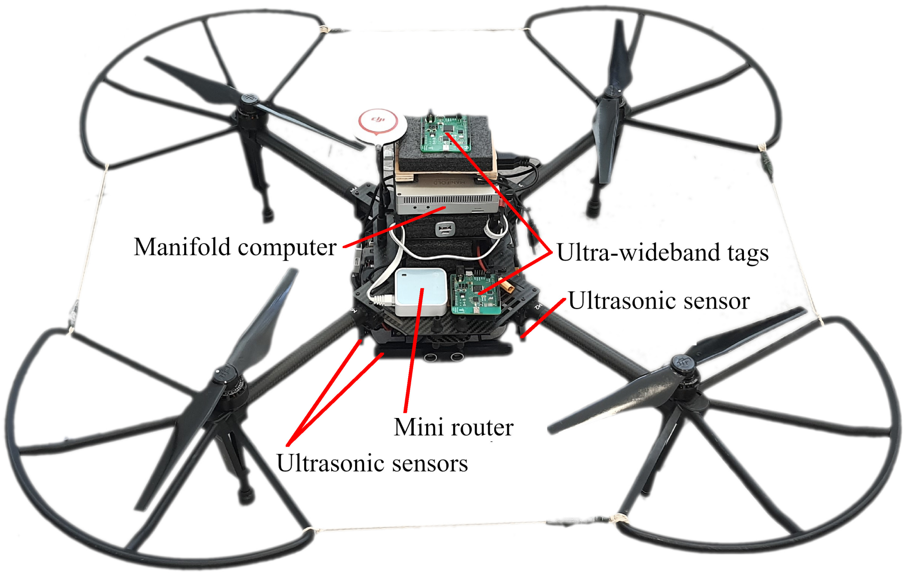


Figure 6.1: Onboard hardware modules. These include a manifold onboard computer, two ultra-wideband tags, a WiFi mini access point and five ultrasonic-stereo vision modules.

In addition to the range-finders, the vehicle is equipped with two ultra-wideband tags, where the second is for redundancy to improve on localization reliability, a DJI-manifold onboard computer and a mini WiFi access point for communication with the ground station. Besides the onboard hardware, the system utilizes off-board hardware modules as well. The off-board modules consist of four ultra-wideband anchors for indoor localization. The complete system architecture is as indicated in Fig. 6.2, where the dashed boarders identify off-board modules and thick borders indicate processing units.

The underlying software for the different modules has been implemented as Robot Operating System (ROS) nodes in C++ and python. The nodes include a Pozyx node that publishes 3D position of onboard tags, guidance node that publishes ultrasonic range-finder depth measurements, path planning node that provide path planning and replan-

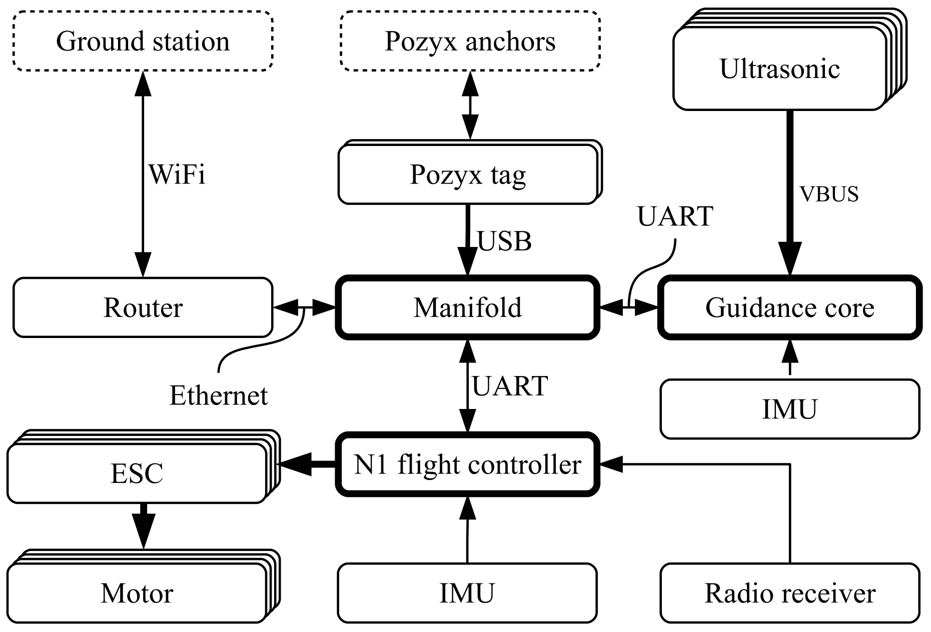


Figure 6.2: System architecture for the ultra-wideband-based-aided inertial localization system.

ning services and position tracking node that takes in the actual vehicle position, sensor measurements and desired position, and generates position errors that it publishes to the vehicle's lower-level controller node for actuation. The position tracking node also manages obstacle avoidance. The control architecture consists of three main layers, behavioural, executive and planning layers as indicated in Figure 6.3. To cope with the limited onboard computational resources, all publishing nodes were limited to a fixed frequency of 20 Hz.

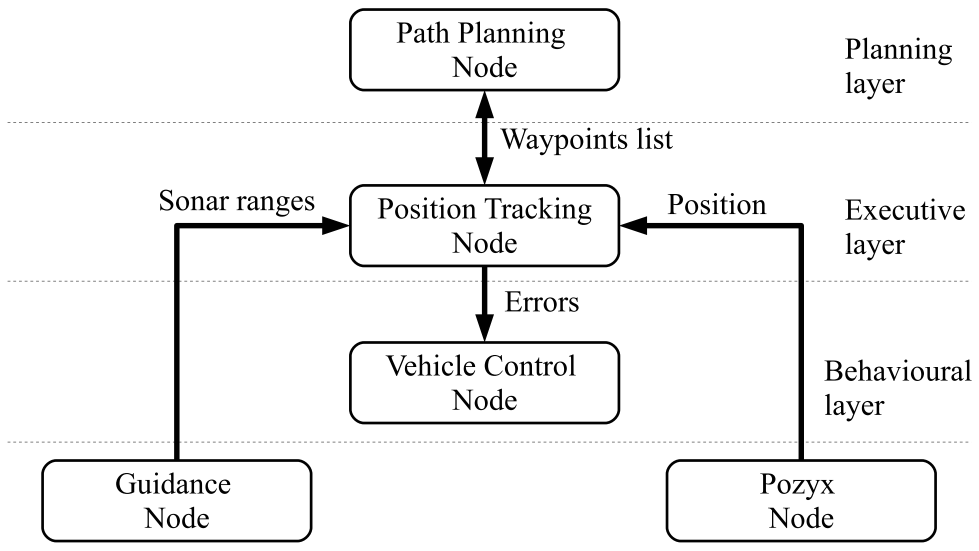
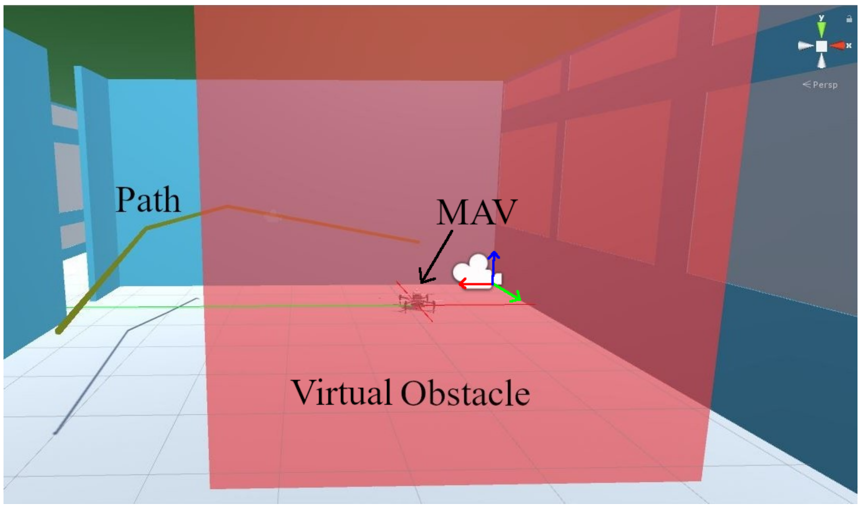


Figure 6.3: ROS control architecture with sensing, behavioural control, executive control and planning layers, bottom to top respectively.

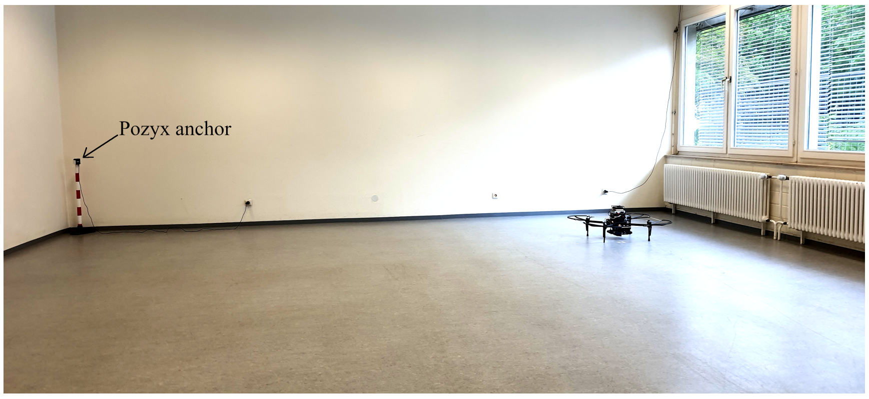
6.1.3 Experimental Results

The proposed framework has been tested in a 10.25 m × 7.65 m × 3.40 m enclosed section of the studied environment to verify the performance of the different navigational components. Virtual obstacles have been introduced in the simulator and paths planned to account for them, then flown by the vehicle. During each test flight, the system lifted-off in the vicinity of the first way point at a known initial heading. After stabilizing at a height of 1 m, tracking of the desired waypoints commences.

In this test, a path was planned as if there was a 4.00 m×3.65 m×3.40 m static obstacle in the enclosure. This kind of hybridization enables low risk testing of the waypoint tracking behaviour of the vehicle. The test path planning environmental setup was as indicated in Fig. 6.4a and Fig. 6.4b for simulation and real tests respectively. Fig. 6.5 shows the fidelity of the actual path to the planned path. The spikes near the take-off point are attributed to ground effects and environmental interference with the ultra-wideband radio positioning system. Generally, the positioning system exhibited non-uniform performance with chattering resulting from onboard tag occlusion and vehicle motion. Overall, despite the common notion of poor quality paths from sampling-based path planners, their fast planning speed provided ample time for path simplification that resulted in a piece-wise linear feasible path executable by a MAV.



(a)



(b)

Figure 6.4: (a) Simulated test environment with a virtual obstacle (red cuboid) and the planned path indicated as a piece-wise linear trajectory. (b) Actual test environment with the vehicle placed near the far-right corner.

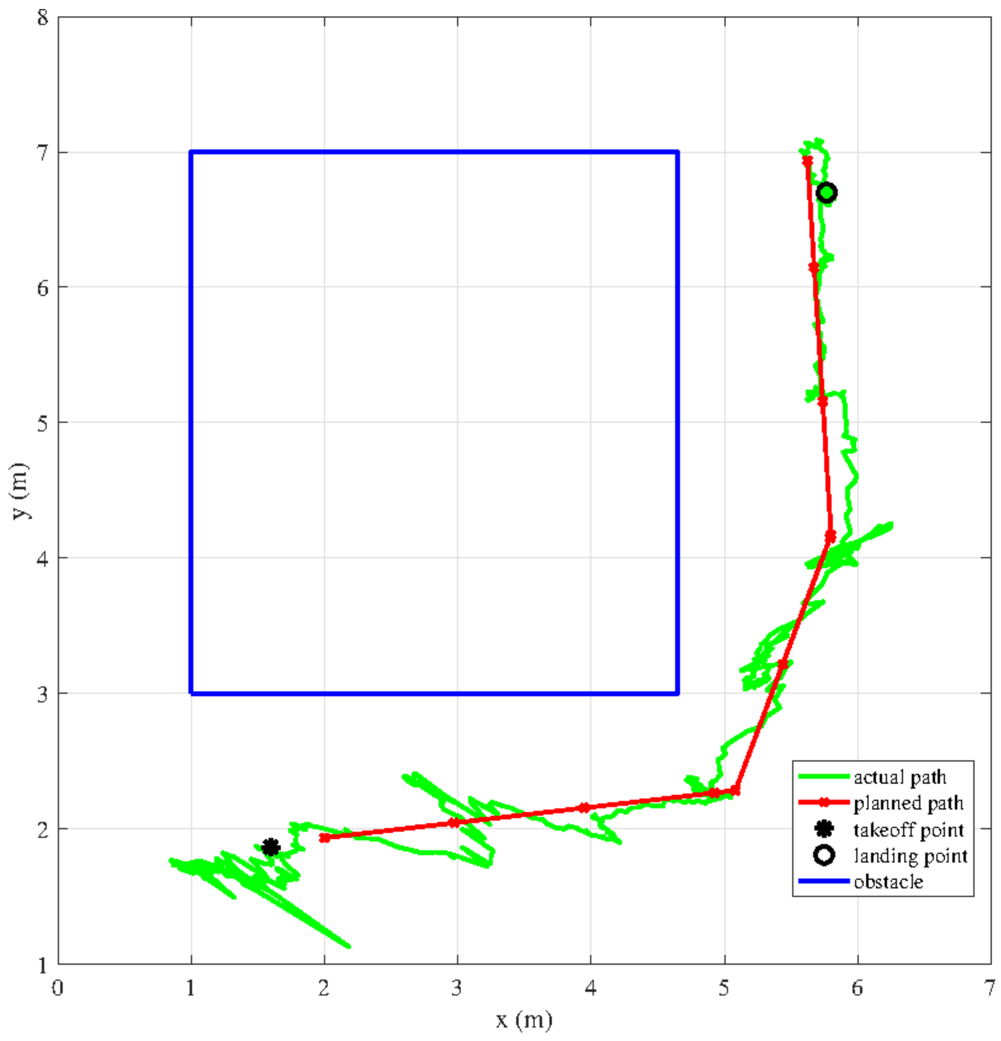


Figure 6.5: Actual path compared to the planned path.

6.2 Visual SLAM

Although the localization approach presented in Sect. 6.1 is GNSS-independent and computationally less demanding during runtime, it requires prior installation of positioning anchors at known locations in the operating environment. This limits its usage to only known and accessible environments. This section describes a class of localization solutions that require no off-board modules, known as SLAM. SLAM can be supported by inertial sensors, LiDAR, RDB-Depth, stereo vision and monocular cameras, but for this work in particular, stereo visual SLAM was implemented to harness its information richness and passive nature. The implementation fused inertial and vision state information, following the assertion stated in Sect. 2.3.1 that inertial-visual integration holds the key to GNSS-independent localization solutions of the future.

Here, visual SLAM is exploited for the purpose of localizing a MAV quadrotor (Matrice 100 in Fig. 6.6) in an office building. In this work stereo visual SLAM is implemented using a front facing stereo camera. The hardware setup is as indicated in Fig. 6.7, where the dashed boarders identify off-board modules and thick boarders indicate processing units. The vehicle has five stereo vision modules, but only the front-facing module is used for visual SLAM due to bandwidth limitations. The camera has been calibrated and camera information stored on the guidance core module showed in Appendix A.

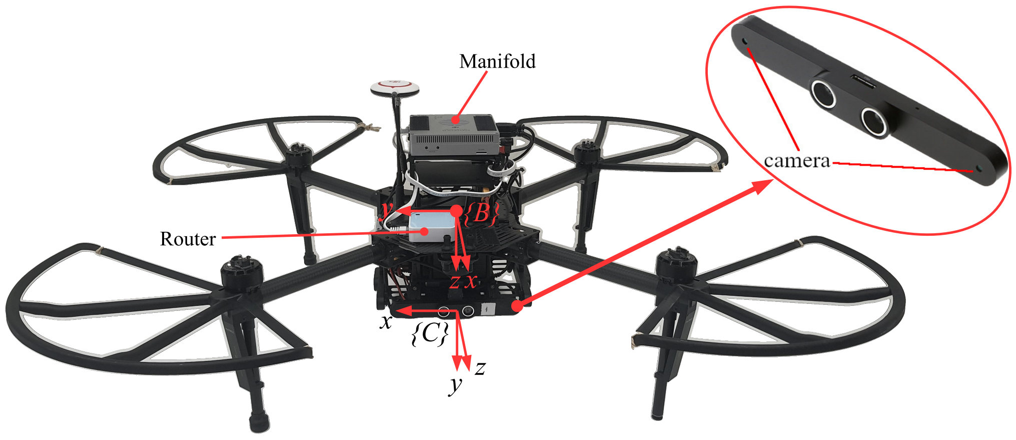


Figure 6.6: Stereo vision module and vehicle coordinate frames. $\{C\}$ is camera frame and $\{B\}$ is body fixed frame. In the upper right is a magnified view of the stereo vision module.

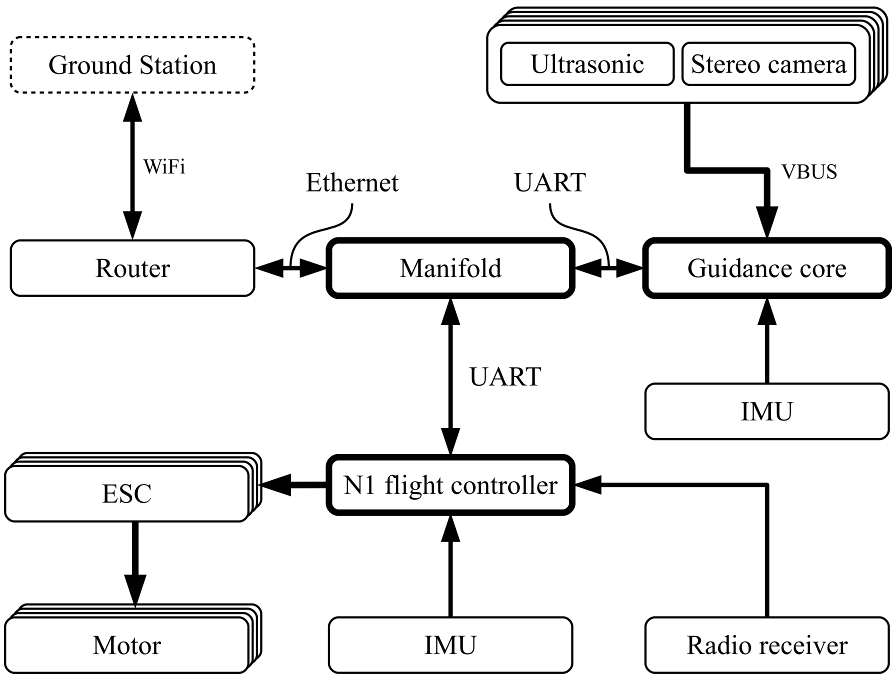


Figure 6.7: Vehicle hardware setup.

The first step in this localization process is environmental perception through the front-facing stereo vision sensor actuated by the guidance core module. The guidance core then applies the stored camera calibration information to correct for image distortions. The main onboard computer (manifold) then polls these image pairs, compresses them to 20% of their original size before transferring them to the ground station for visual localization. The compression process is necessary for bandwidth minimization during wireless transfer to the ground station. On the ground station, the images are decompressed and applied for visual odometric calculations. The complete process is as indicated in Fig. 6.8, where TF is the camera frame to vehicle frame transformation and UFM (Update Feature Map) is a decision checker for when to update the feature map.

Once a new stereo pair of images is received and decompressed on the ground station, GFTT (Good-Features-To-Track) features [180] are extracted and stereo correspondences

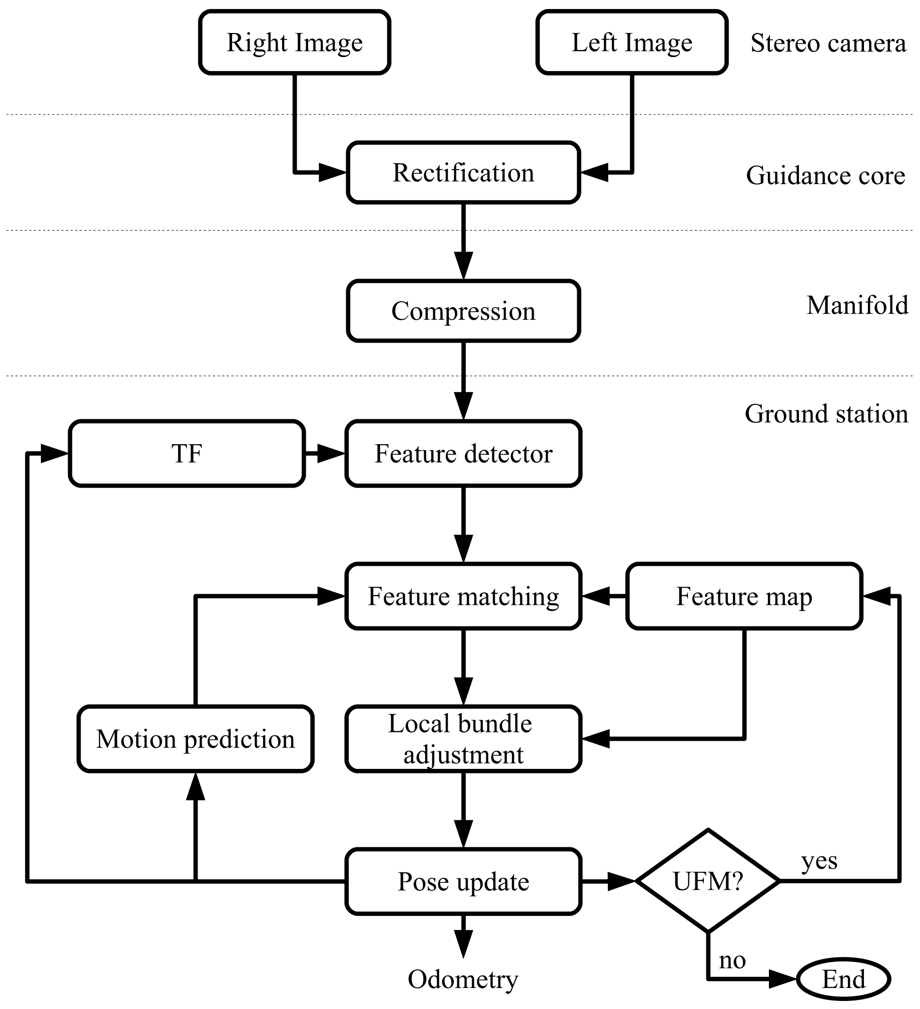


Figure 6.8: Visual odometry process flow. This is an adaptation of the stereo odometry process from [2].

computed from optical flow implemented using the algorithm presented in [181] to establish disparity for each of the detected features in the right and left image pair. Then these features are matched to existing features in the feature map to establish correspondences, and hence transformation matrix. In this step, the features are first represented as Binary Robust Independent Elementary Features (BRIEF) descriptors [182] for comparison using the nearest neighbour distance ratio test. To expedite the search for feature correspondences, a constant velocity model is applied to limit the search space. The perspective-n-point algorithm and Random Sample Consensus (RANSAC) algorithms are run to determine the relative motion as represented by the resulting homogeneous transformation. The relative motion estimate is then refined by applying bundle adjustment on the features in the feature map. Using this refined homogeneous transformation, the vehicle odometry and camera-vehicle transformation are updated. If the number of inlier features obtained during the relative motion estimation step falls below a predefined threshold (in this case 30%), the feature map is updated. To generate the final vehicle pose estimate, visual odometry is fused with IMU measurements in an EKF as indicated in Fig. 6.9.

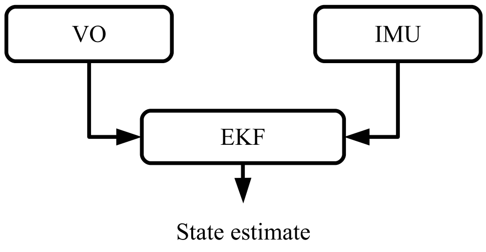
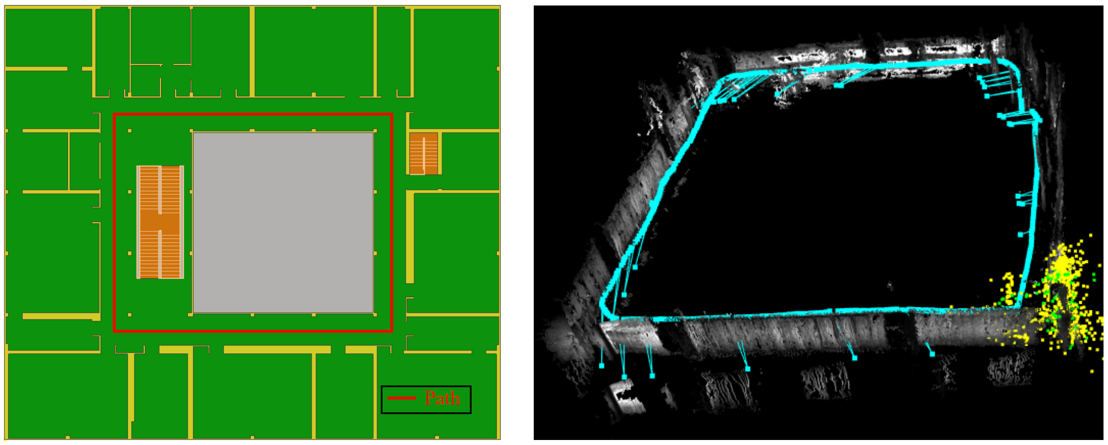


Figure 6.9: Pose estimation scheme, where visual odometric estimates are fused with IMU angular rates in an EKF.

6.2.1 Experimental Results

This localization framework has been implemented in a distributed computing setup, where data acquisition is performed by the onboard manifold computer and localization algorithm performed on the ground station. This is due to the fact that the onboard computer is unable to run the SLAM algorithm for a distance greater than 30 m. The ground station is a 64-bit, 8 GB RAM laptop computer running Ubuntu 14.04 LTS Linux distribution. The onboard manifold is a derivative of Nvidia TK1 system on chip with 192 Compute Unified Device Architecture (CUDA) cores and 2 GB of RAM, running Ubuntu 14.04 LTS distribution. It is physically mounted on top of the vehicle as shown in Fig. 6.6. All localization algorithms are implemented in ROS framework on both computing machines and interconnected via a wireless local area network.

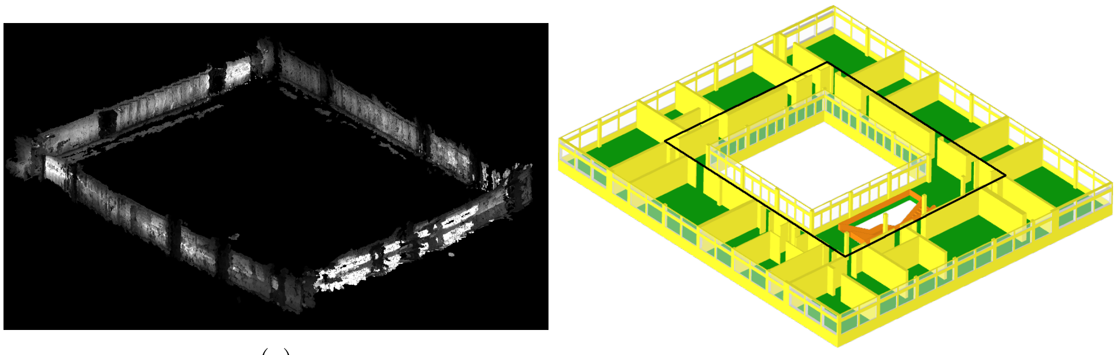
To test the localization capability of the implemented system, an indoor localization task around the hall-way of the environment in Fig. 6.10a is carried out. For safety and reliability reasons, the system was run in a passive SLAM mode, where the actuation commands and vehicle position acted as two independent processes, with actuation conducted by an external operator. Odometric results of a complete flight loop through the hall-way is presented in Fig. 6.10b. The visible spikes are as a result of lost and recovered odometry tracks. Visual SLAM calculated a total distance of 87.72 m instead of the actual 91.2 m representing an absolute error of 3.8%.



(a) (b)

Figure 6.10: (a) Top view of the visual odometry test environment. (b) Visual odometry test results.

The resulting environmental map is as indicated in Fig. 6.11a, put side-by-side with the actual environment for qualitative comparison. The generated map is missing the inner wall information, which is expected since the inner wall is largely transparent. Nevertheless, semantic features like doorways which are key for robot localization in a map are well captured making the map useable for localization.



(a) (b)

Figure 6.11: (a) Generated map of the environment. (b) Actual environment.

6.3 Chapter Summary

This chapter describes and demonstrates the key localization methods studied within this work. The first localization approach utilizes ultra-wideband radio technology to localize the vehicle. Although this approach is found to be simple in terms of implementation and light in terms of computation, it assumes a known and accessible environment and is highly sensitive to metallic materials in and around its measurement volume, and the UWB signals are easily attenuated by obstacles.

As a better alternative, visual SLAM has also been implemented and tested for indoor localization. The SLAM approach fuses stereo visual odometry with IMU measurements for pose estimation. Unlike ultra-wideband localization, visual SLAM assumes no explicit knowledge of the environment, but is computationally demanding and may require regular revisiting of some areas to recapture lost tracks and/or perform loop closure update.

Chapter 7

Modelling and Simulation

All models are wrong but some are useful.

George Box

Simulation offers an alternative to hardware-based validation of both software and hardware performance, but requires a model with reasonable fidelity to work with. This chapter describes an alternative modelling strategy for Micro Aerial Vehicles (MAVs) and their environments that requires no explicit development of mathematical models as in Sect. 3.3, known as geometric modelling. In addition to geometric models, sensor mathematical models and simulation scenes for testing and evaluating the performance of these models are also demonstrated. These together constitute the MAV simulators.

Simulators are an indispensable tool in the design and development process of modern robotic systems. These modern systems are normally complex beyond human imagination requiring the help of tools to conduct systematic analysis and evaluation of the envisioned systems. Simulators find applications in controller designing, algorithm validation, parameter identification, training of human operators, software debugging, generating model training datasets, and systems evaluation and testing to mention but a few.

The simulators presented here are graphical simulators, which substitute the burden of building high fidelity mathematical models with that of building realistic geometric representations of the system and its environment. Numerical simulations are susceptible to approximation errors if not robustly designed, which may lead to numerical instability. Therefore, geometric model simulation provides a much simpler alternative that abstracts away the lower-level numerical implementations. It also facilitates the process of model parameter identification and environmental representation, making it a valuable complement or alternative to hardware simulation, which may not exist, may be dangerous, may be highly complex or may be prone to damage among others.

Geometric model simulation requires a dynamics simulation engine to manipulate the models and extract information necessary for computing the state of the system or for inputting into mathematical models. The dynamics simulation engine also known as physics engine takes care of applying the laws of physics to the simulated bodies and generating physics correct output behaviour. For this purpose, Unity[1] software framework was selected and applied to this study as it is cross-platform, easy to use and supports mathematical model and geometric model hybridisation, which is the most common use case, and one that is exploited in this work. It also supports colliders, ambient lighting control, ray casting, rigid-body dynamics and kinematics, and simulation of joints, conservative forces like gravity and non-conservative forces like friction and drag among others.

[1]https://www.unity.com/

With geometric simulation, identifying the necessary simulation parameters is simplified (since there are relatively fewer parameters and their associated values are normally related to the physical properties of the models) in comparison to parameter identification for mathematical model simulation. This simplicity offers a number of benefits like an easy and rapid modelling process, readily available environmental parameters, automated handling of the physics, also geometries exhibit spatial extents onto which forces and moments can be applied and reactions observed.

Graphical simulators are an assembly of interacting replicas of dynamic systems, sensors, actuators and environment models, with interactions governed by the laws of physics implemented in a software package known as a physics engine. The next section describes the different models, including sensors, vehicle and environment models.

7.1 Components Modelling

Mathematical and geometric models of the different hardware components have been designed and implemented in such a way as to achieve high fidelity with the actual systems.

Inertial Measurement Unit

The Inertial Measurement Unit (IMU) consists of two triaxial sensors namely, rate-gyroscope and accelerometer. The specifications used in designing the IMU simulation model were adopted from MPU-9255 motion tracking device product specification datasheet [183]. The rate gyroscope specifications include a dynamic range of ± 500 °/s and rate noise spectral density of 0.01 $°/\text{s}/\sqrt{\text{Hz}}$ corresponding to an angle random walk of 0.6 $°/\sqrt{\text{h}}$. The rate gyroscope was modelled as a 3D object whose angular velocity is contaminated by three additive noise processes namely, constant bias error $\delta\omega_{BE}$, angle random walk $\delta\omega_{ARW}$ (additive white Gaussian noise) and bias instability $\delta\omega_{BI}$. Therefore, the gyroscope output is given by Eq. 7.1.

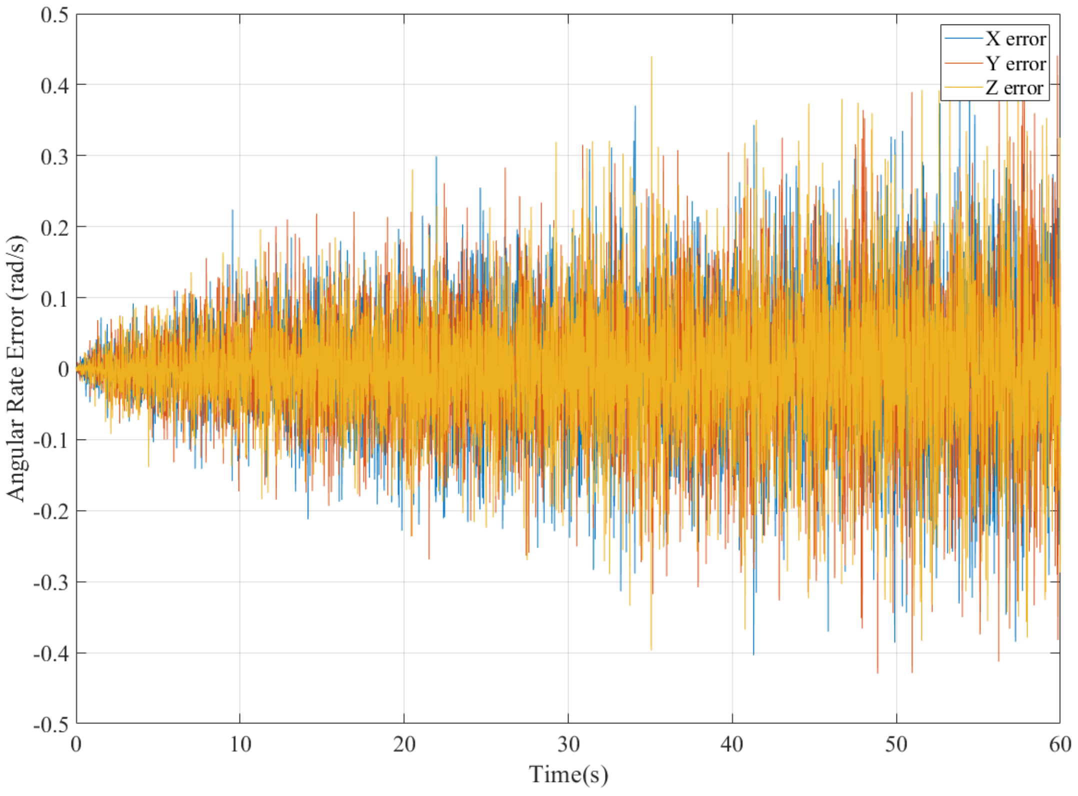
$$\underline{\omega} = \underline{\omega}_{ideal} + \delta\underline{\omega}_{BE} + \delta\underline{\omega}_{ARW} + \delta\underline{\omega}_{BI} \tag{7.1}$$

A sample of the simulated gyroscope stationary output measurements is presented in Fig. 7.1, where Fig. 7.1a indicates angular rate noise process and Fig. 7.1b indicates the resulting angle noise process from integration of the noise process in Fig. 7.1a. These results were achieved with the following settings, a constant bias error $\delta\omega_{BE} = 0$ and an angle random walk 0.6 $°/\sqrt{\text{h}}$.

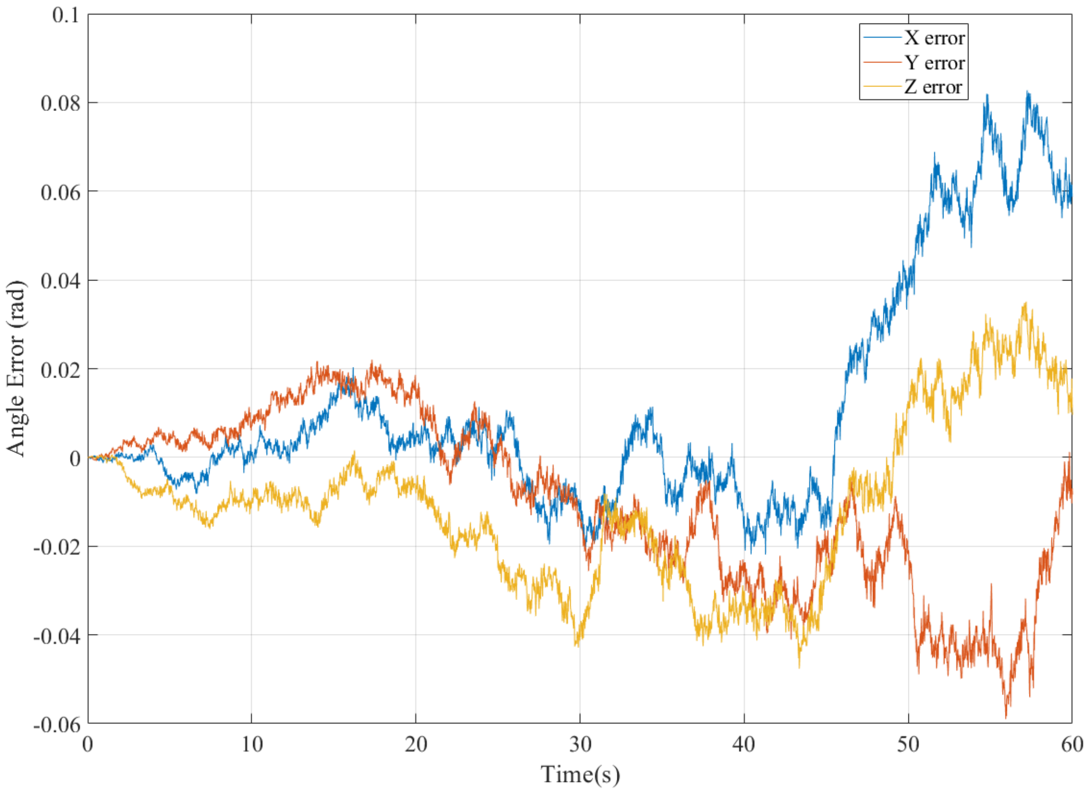
The accelerometer specifications are as follows: a dynamic range of ± 8 g and noise power spectral density of 300 $\mu\text{g}/\sqrt{\text{Hz}}$ corresponding to a velocity random walk of 0.17658 $\text{m/s}/\sqrt{\text{h}}$. The accelerometer was modelled as a 3D object whose linear accelerations were contaminated by three additive noise processes namely, constant bias error δa_{BE}, velocity random walk δa_{VRW} (white Gaussian noise) and bias instability δa_{BI}. Therefore, the accelerometer output is given by Eq. 7.2.

$$\underline{a} = \underline{a}_{ideal} + \delta\underline{a}_{BE} + \delta\underline{a}_{ARW} + \delta\underline{a}_{BI} \tag{7.2}$$

A sample of the simulated accelerometer output measurements is presented in Fig. 7.2, where Fig. 7.2a indicates acceleration noise process and Fig. 7.2b indicates the resulting velocity noise process from integration of the noise process in Fig. 7.2a. These results were achieved with the following settings: a constant bias error $\delta a_{BE} = 0$ and an angle random walk 0.17658 $\text{m/s}/\sqrt{\text{h}}$.

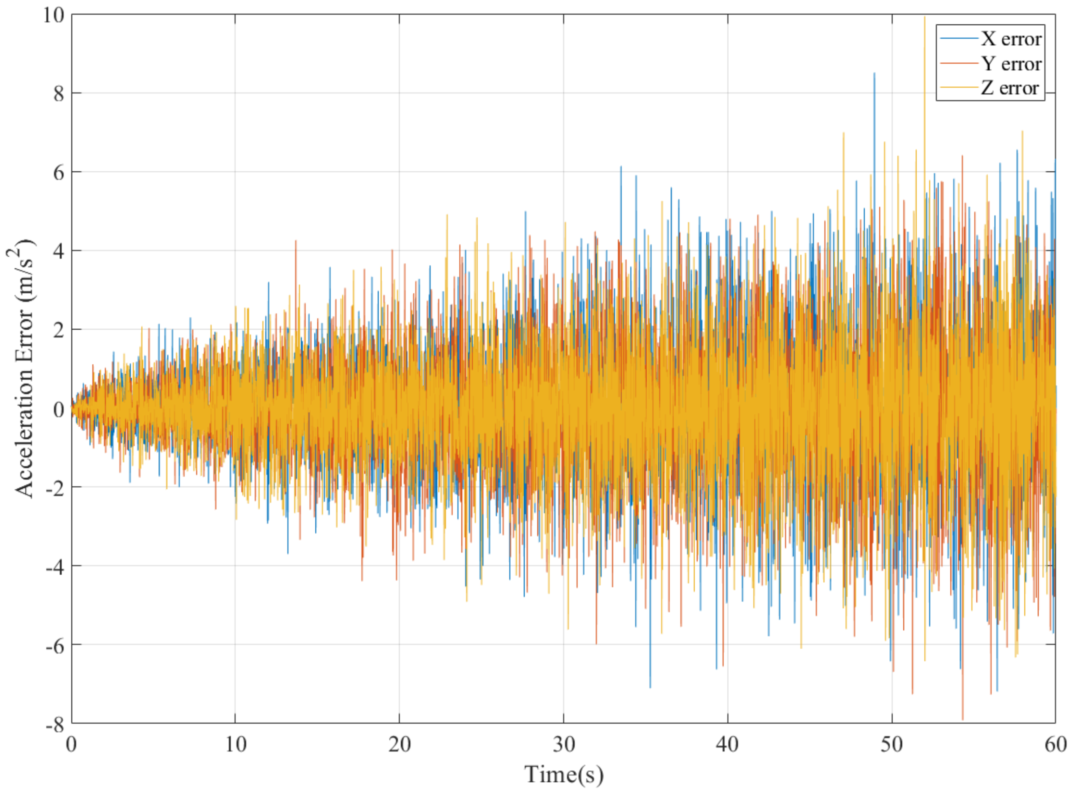


(a)

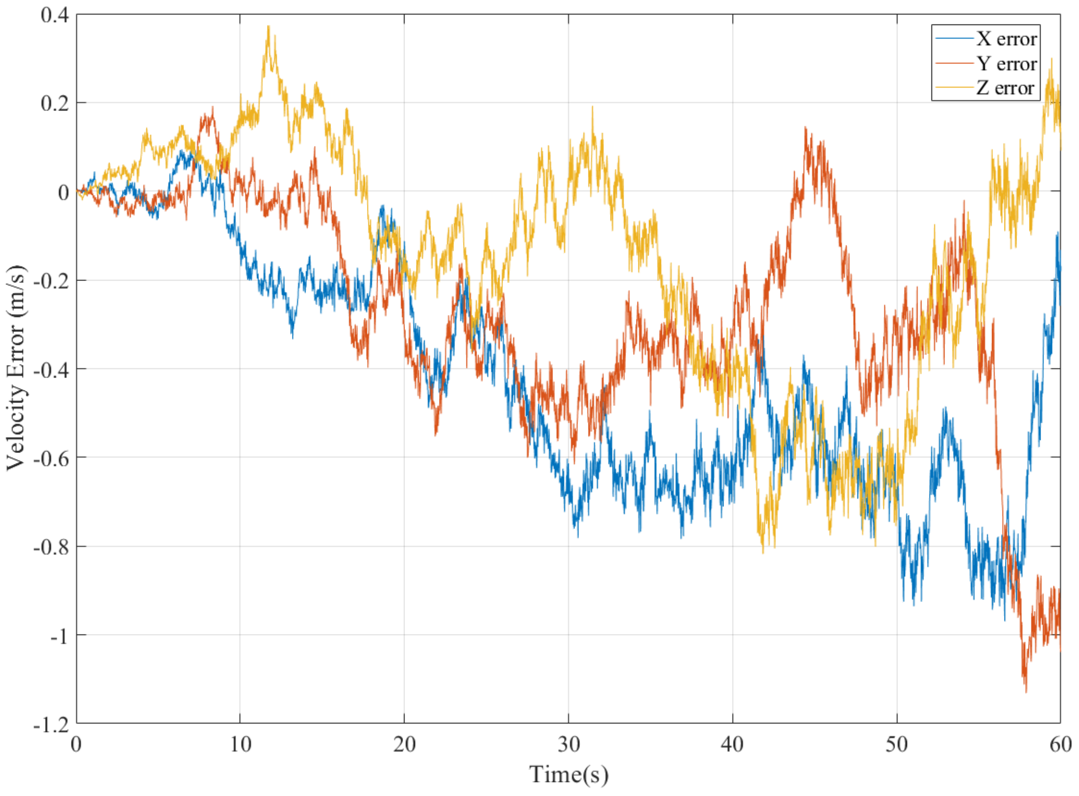


(b)

Figure 7.1: (a) Gyroscope angular rate noise process. (b) Angle noise process.



(a)



(b)

Figure 7.2: (a) Acceleration noise process. (b) Velocity noise process.

Ultrasonic Rangefinder

This sensor serves two purposes on the vehicles namely, obstacle detection and Above Ground Level (AGL) measurement. These 1-DOF sensor modules were modelled to have a dynamic range $z \in [0.03, 6]$ meters and an open angle of 60 °. The measurements are assumed to be contaminated by zero mean additive white Gaussian noise δz parametrized by $\delta z \sim \mathcal{N}(0, \sigma^2)$, where σ was set to 0.01 m. The beam angle pattern of this sensor model is as indicated in Fig. 7.3. These sensor parameters were obtained from SRF08 ultrasonic ranger measurements.

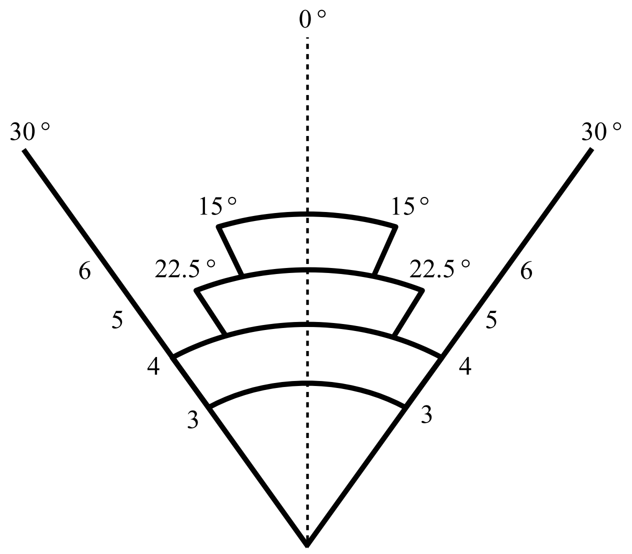


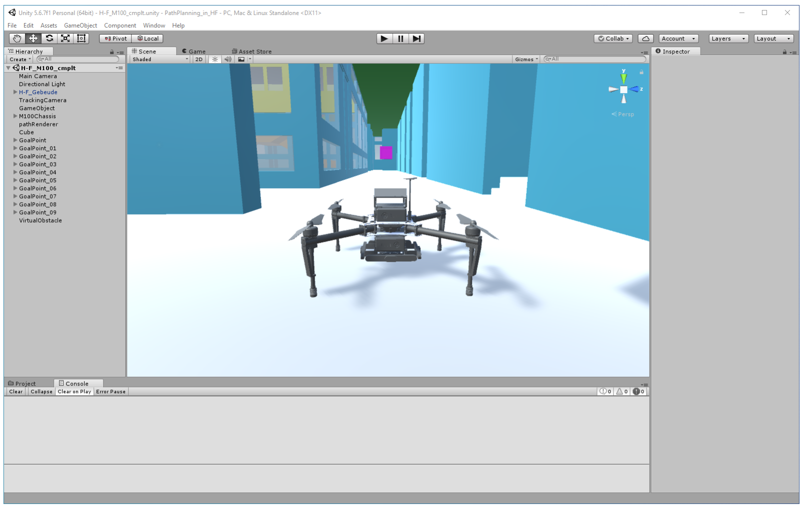
Figure 7.3: Sonar beam angle pattern, modelled after SRF08 ultrasonic ranger.

Ultra-Wideband Sensor Module

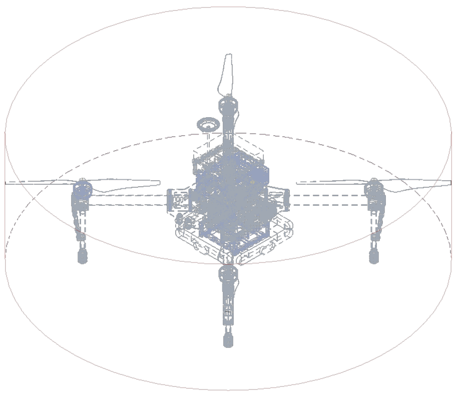
An Ultra-Wideband (UWB) sensor module tracks its own position using Two-Way-Ranging (TWR) protocol, where range is obtained from time-of-flight of a packet. The range measurements are assumed to be contaminated with additive white Gaussian noise $\delta r \sim \mathcal{N}(0, \sigma^2)$. Unlike in practical applications where the position has to be solved for through trilateration or multilateration, the ideal tag position is readily available in the simulators. All that is needed is to contaminate it with appropriate noise to obtained a simulated position measurement. The Pozyx UWB system deployed in this study has a horizontal planar position standard deviation $\sigma = 0.10$ m. Its vertical position is extremely noisy and unreliable due to close vertical positioning of anchors, a problem attributed to the nature of residential buildings, whose vertical size is limited to a few meters, in this case less than 4 m. So, this positioning system is not used for AGL measurement in these tests.

Quadcopter Geometric Models

Two vehicle simulation models are provided, namely a complex vehicle model, i.e., actual airframe representation and a simplified vehicle model, i.e., 3D convex-hull approximation of the actual model. The former is applied for vehicle dynamics simulation and the latter for path planning. Using the complex model for path planning would lead to a slow path search process breaking the desired real-time path planning and replanning capability. The two vehicles deployed in the studies conducted in this work are DJI Matrice 100 presented in Fig. 7.4 and F450 presented in Fig. 7.5, where Fig. 7.4a and Fig. 7.5a show their complex models applied for vehicle dynamics simulation, and Fig. 7.4b and Fig. 7.5b show their simplified convex-hull approximation models applied for path planning. The simplified model for DJI Matrice 100 is a cylindrical convex-hull of size ∅0.996 m × 0.345 m and that of F450 is a cylindrical convex-hull of size ∅0.710 m × 0.165 m. The complex models for Matrice 100 and F450 are composed of 279302 vertices and 1227879 vertices respectively, while the simplified models are composed of 238 vertices and 238 vertices respectively. The differences between the complex and simplified models represent a 99.91% and 99.98% reduction in vertices for Matrice 100 and F450 respectively.



(a)



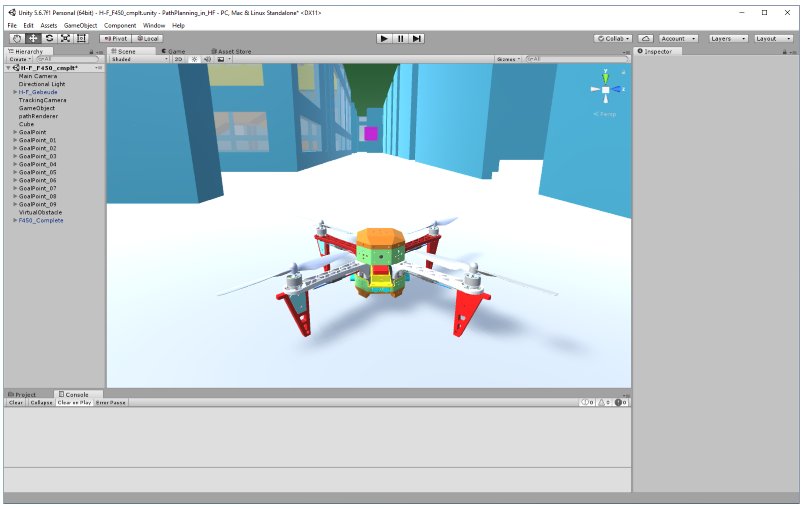
(b)

Figure 7.4: DJI Matrice 100 MAV models. (a) Complex model imported in the test environment in Unity. (b) Simplified cylindrical model.

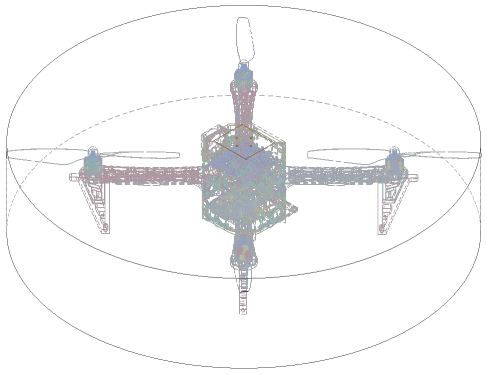
The resulting complex vehicle dynamic models are designed so as to achieve a high fidelity, i.e., the geometric models are actuated by thrusts acting at each of the four propeller-motor joints. These thrusts produce moments about the vehicle centre of mass, therefore, making the dynamic behaviour sensitive to onboard weight distribution as is the case with the actual vehicle. Furthermore, the models experience drag and gravitational effects, phenomena difficult to properly represent in purely mathematical simulators.

Environment

Due to strict European UAV regulations for outdoor flights, an indoor test environment was chosen. The environment is a two-floor building with two points of access between the floors, indicated by the two orange staircases in Fig. 7.6a. It is GNSS-denied, with narrow corridors, but structured. For simulation, the environment is modelled as the collada



(a)



(b)

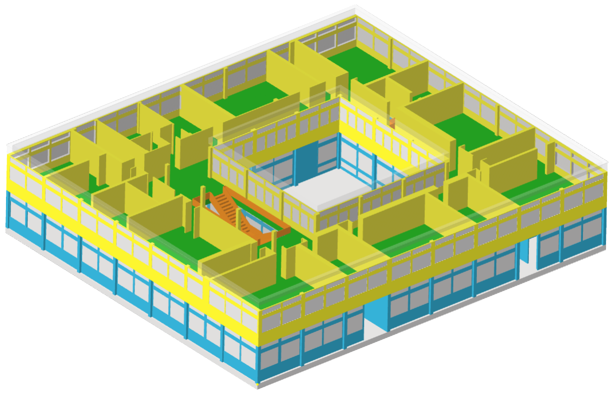
Figure 7.5: F450 MAV models. (a) The complex model imported in the test environment in Unity. (b) Simplified cylindrical model.

dae 3D model in Fig. 7.6a, which inherited properties such as collidability, renderability, gravitation effect and non-conservative force effects upon importation into a Unity scene, presented in Fig. 7.6b.

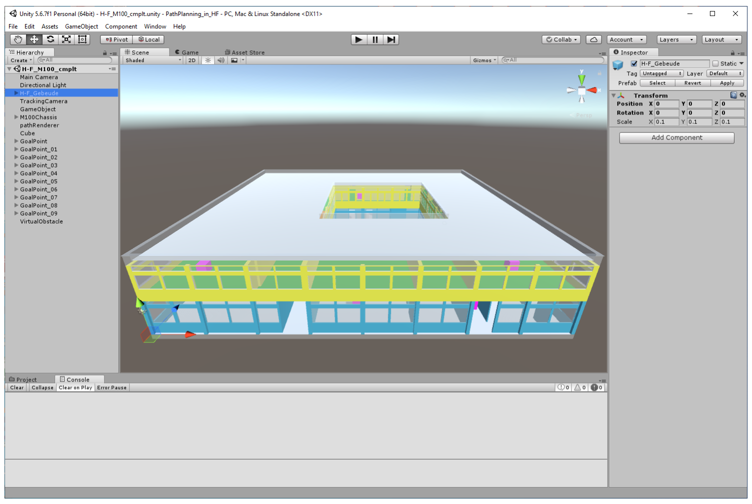
7.2 Chapter Summary

This chapter provides descriptions of model components applied in MAV simulations, including sensors, vehicle and environment models. To simulate sensing, a sonar rangefinder and an IMU have been modelled with an additive white Gaussian noise process, and constant bias, random walk and bias instability noise processes respectively. As an alternative to pure mathematical models that suffer from numerical instability, geometric models have been used for vehicle and environment simulations. The geometric models render well with the physics engines powering graphical simulators like Unity[2] and Unreal[3]. For studies performed during this research, the Unity software framework was chosen for its ease of use, but with similar performance as Unreal. Controlling these models requires additional control logic which has been developed in C# programming language using Unity scripting API (Application Development Interface). The developed hardware, environment and

[2]https://www.unity.com/
[3]https://www.unrealengine.com/



(a)



(b)

Figure 7.6: Test environment model. (a) Collada 3D model of the building without a rooftop. In blue is the ground floor and in yellow is the first flow. (b) Building model imported into a Unity scene.

algorithms have been integrated into various MAV simulator scenes, one of which is the path tracking scene presented in Fig. 7.7.

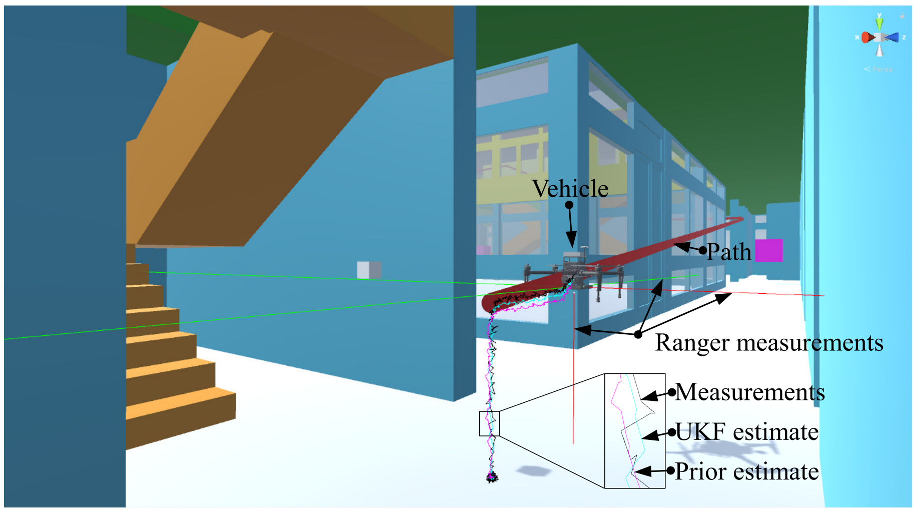


Figure 7.7: Dynamic simulation of DJI Matrice 100 in Unity.

Chapter 8

Conclusion

The question of how to build complete GNSS-independent UAV autonomous navigation solutions has been tackled from a bottom-up approach, where navigation competences, namely perception, localization, motion control, cognition and obstacle avoidance are analysed independently and improved upon whenever deemed necessary. From the preliminary literature review, it was determined that UAV stability controllers and structural design reached full technology maturity and hence not covered in this study.

The proposed online path planning ensemble with three concurrently executed randomized sampling-based path planners, namely RRT, RRT-Connect and BiT-RRT demonstrated a fast, but variable planning time. When tested in a two-floor building environment for arbitrary random goal queries, it demonstrated a mean path planning time (sum of path searching and path simplification time) of 1.28 s with a standard deviation of 1.70 s, which is fast enough for online path planning and replanning. Despite small planning times, the ensemble is generally not complete, a property inherited from the constituent planners, but the environmental adaptability resulting from concurrent planning increases the planning success rate as it simulates three simultaneous trials for each query.

For large-scale path plans that exceed the endurance of the vehicle, an offline large-scale aerial coverage path planning algorithm that uses Voronoi exact cellular decomposition to partition large areas into manageable partitions, and Boustrophedon coverage patterns to generate coverage paths for partition has been developed, and its feasibility successfully tested in a SIL simulator. The planner is applicable to tasks with a single MAV, heterogeneous and homogeneous fleets of MAVs. Although, close to perfect waypoint tracking has been assumed, which is practically difficult to achieve.

Two complete GNSS-independent navigation frameworks have been demonstrated. The first one is a radio navigation solution based on UWB-aided inertial localization, a randomised sampling-based global path planning ensemble and ultrasonic range-finders for obstacle detection. Path tracking was accomplished through a custom higher-level linear controller in a sequential loop with an off-the-shelf lower-level stability controller. UWB provides direct 2D-position updates to compensate for inertial drift, but it is affected by presence of conductive materials and requires environmental installation of anchors, which may not be feasible under certain circumstances like localization in a partially collapsed building. These challenges have been partially addressed in the second navigation framework, which implemented a SLAM algorithm. It localizes through fusion of stereo visual odometry and inertial odometry in an EKF. The resulting localizer gave an error of 3.8% over a distance of 91.2 m (flight in a looped hallway). Unlike UWB-based localization, visual SLAM assumes no explicit knowledge of the environment, but is computationally demanding, may require regular revisiting of some areas to recapture lost track and in this case it exhibited a prohibitively low update rate of 10 Hz.

By observing that autonomous robots ultimately replace human skilled workers, a mapping between human job characteristics and robot task characteristics gave a set of four robot task characteristics upon which autonomy metric functions have been derived, for assessing LoA and DoA. These two autonomy measures augment each other, where LoA provides a global categorical discrete measure and DoA provides a continuous local measure capable of telling the performance differences among systems at a similar level of autonomy. The robot task characteristics include capabilities, trust factor, performance capacity and environmental complexity. Capabilities define LoA, capabilities, trust factor and performance capacity define DoA, while environmental complexity provides contextual information. The output of this proposed autonomy framework is a three-part autonomy designation, i.e., degree of autonomy at a particular level of autonomy in an environment of a specific complexity. The framework has been tested successfully on a simulated MAV with hypothetical capabilities. Also to show that the proposed metrics are relevant and easily obtainable, a demonstration of their extraction from the competition rules of the DARPA subT challenge is reported. In the process, missing reference information was identified as well. Besides being easily obtainable, these metric functions output continuous values in range $[0, \infty)$. Hence satisfy the three desired characteristics of autonomy metrics namely easily measurable, broad enough to capture autonomy evolution and with good output resolution. Despite all the positive aspects of these metrics, the framework assumes availability of environmental information, which may be unknown is some applications. Nevertheless, autonomy can still be determined in a non-contextual sense.

A technology readiness assessment of GNSS-independent navigation solutions indicated that the current solutions are of TRL-6, interpreted as "successful high-fidelity prototype demonstrations in relevant environments". Prototype testing in the actual environments, final product production and deployment in the actual environments, representing levels 7, 8 and 9 respectively are yet to be achieved.

High fidelity MAV graphical simulations in form of Unity software framework scenes simulating the different test scenarios have been developed. Each scene features a thrust actuated MAV, sensors and the test environment models and supporting scripts to simulate the different navigation competences. The physics engine integrated in the Unity software framework enable seamless integration of appropriate physics behaviour in the environment, vehicle and sensor models, an insurmountable task in exclusively mathematical simulators. An interfacing of this simulator in a real flight enabled demonstration of safe experimentation using augmented reality, in which virtual obstacles are introduced, accounted for and avoided during actual flights without the risk of collision.

To summarise, this research has contributed in the following subfields:

- A fast global randomized sampling-based path planning ensemble.
- A large-scale coverage path planner with an integrated task scheduler and exact cellular decomposition. This planner is presented in the following article: N. Gyagenda, A. K. Nasir, H. Roth, and V. Zhmud, "Coverage path planning for large-scale aerial mapping," in Annual Conference Towards Autonomous Robotic Systems. Springer, 2019, pp. 251–262.
- A high resolution autonomy evaluation framework, consisting of an eleven autonomy-level chart and a novel DoA mathematical model. Part of this framework is presented in the following article: N. Gyagenda, O. Gamal, and R. Hubert, "A non-contextual method for determining the degree of autonomy to develop in a mobile robot," IFAC-PapersOnLine, vol. 50, no. 2, pp. 271–276, 2017.
- Expanded the existing UAV classification schemes to include the nano class.
- Developed high fidelity quadcopter indoor graphical simulators with state estimation, sensing, obstacle avoidance and path planning functionality.

- Established the technology readiness level of the existing GNSS-independent navigation solutions to highlight the achievement of over a decade of research effort.
- An overview of the current state of GNSS-independent UAV autonomous navigation.

8.1 Future Works

Looking at the expansion plans of GNSS systems both on a global and local scale, with the aim of supporting autonomous applications, hybrid navigation solutions combining GNSS and GNSS-independent navigation techniques provide a reasonable future research direction. These would require development of adaptive state estimators incorporating integrity monitoring models and adaptive uncertainty models to enable stable transition between the different modes of operation. This adaptability would ensure system reliability during transition phases, which might be triggered as a result of GNSS interference, changing lighting conditions, magnetic interference, solar interference and ultraviolet radiation to mention but a few. Therefore, developing navigation systems capable of adapting to deteriorating perception with use of other sensing modalities would result in more robust navigation solutions.

Navigation is a core functionality of autonomous UAVs. Its failure may result in costly property damage and loss of lives, which emphasize the need for high reliability in these systems. For this purpose, integrity monitors have been developed and integrated into navigation systems. But the current integrity monitors are lacking in a number of aspects, namely monitor a single fault yet the complexity of navigation systems suggests a likelihood of multiple faults, inexistent requirements of navigation performance, need for advanced fault elimination techniques not limited to only one or two fragile system aspects, need for predictive fault detection as opposed to actual fault detection. Addressing these pending issues in low computational complexity integrity monitoring-UAV navigation software architectures would greatly improve on the overall performance of autonomous navigation systems of the future.

Another avenue worth exploring is augmenting perception with event cameras and signal-of-opportunity sensors. These sensors offer conditional perception, which may improve localization robustness resulting in better navigation performance. With advanced sensors also comes the need for advanced algorithms. Active vision, machine learning techniques especially physics-informed neural networks and other adaptive AI techniques are expected to produce comparatively better solutions for addressing sensor error modelling and semantic scene understanding. Finally, 5G communication technology standard can be applied to achieve high bandwidth data exchange enabling distributed onboard-off-board data-driven navigation techniques.

Last but not least, robotic systems share their environments with other dynamic agents like humans and robots. Safe operation in such environments necessitates accounting for the constraints imposed by those other agents, but the autonomy evaluation framework presented in Sect. 4.1 does not account for these dynamic obstacles. Therefore, there is need to expand this framework to account for such obstacles in the operating environments.

Appendix A

Hardware Specifications

The main hardware used in the projects presented in this work include DJI Matrice 100 quadcopter, Manifold minicomputer, Guidance system, Pozyx positioning system and F450 quadcopter. The important specifications for each are given in the subsequent sections.

DJI Matrice 100 Quadcopter

This is an x-configuration quadcopter developed by SZ DJI Technology Co., Ltd. Its key specifications include those listed in Table A.1.

Table A.1: DJI Matrice 100 specifications.

Property	**Value**
Diagonal wheelbase (motor to motor)	650 mm
Maximum diagonal wheelbase	996 mm
Maximum pitch and roll angular velocity	300 °/s
Maximum yaw angular velocity	150 °/s
Maximum pitch and roll angle	30 °
Maximum climb speed	5 m/s
Maximum descend speed	4 m/s
Maximum horizontal speed	22 m/s
Maximum propeller thrust	2.100 kg/rotor
Maximum take-off weight	3.600 kg
ESC signal frequency	30 Hz - 450 Hz
Battery capacity	4500 mAh

This vehicle has a two-level control architecture with a customizable higher level controller, but inaccessible lower-level controller. The lower-level controller runs on a dedicated DJI N1 flight controller hardware, which houses a tri-axial IMU as well. The accessibility to the higher-level controller is what enabled its usage in this work, where a higher-level controller is implemented on a general purpose embedded-Linux onboard computer known as Manifold developed by DJI Technology. This computer accesses sensor values, runs the higher-level control algorithms and generates set-point values for the lower-level controller to track. The specification for the manifold onboard computer are listed in Table A.2.

Table A.2: Manifold specifications.

Property	**Value**
RAM	2 GB DDR3L
CPU	Quad-core, 4-Plus-1 ARM Cortex-A15 MPcore processor
GPU	NVIDIA Kepler “GK20a” GPU with 192 SM3.2 CUDA cores
Storage	16 GB
Network	10/100/1000 BASE-T Ethernet
USB	USB 3.0 Type-A x2, USB 2.0 Type-A x2, Micro-B USB connector
I/O	Mini-HDMI connector, UART port (3.3 V) x2, I/O expansion headers (26 pins), Half mini-PCIe expansion slot
Input voltage	14 V-26 V
Power consumption	5 W-15 W
Weight	197 g
Dimensions	110 mm × 110 mm × 26 mm

The other hardware module used in this work is the Guidance system. The Guidance system constitutes five ultrasonic-stereo vision modules arranged in the four cardinal directions and bottom around the Guidance core as indicated in Fig.A.1, which is an embedded processing unit that manages and controls the five sensor modules, as well as interfacing with external hardware seeking to utilize these sensor readings. Table A.3 shows the main specifications for the Guidance system, which is developed by DJI Technology.

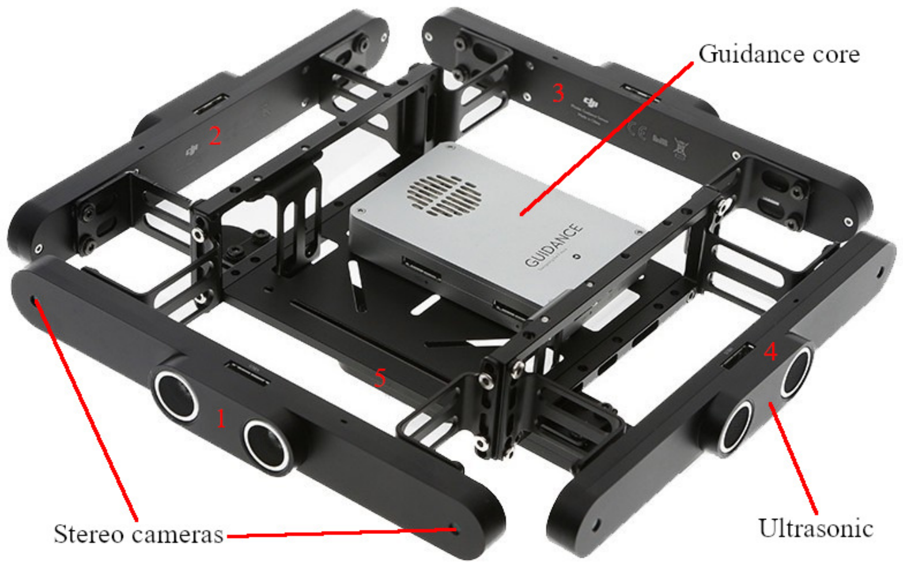


Figure A.1: Guidance system with the five sensor modules marked 1 through 5.

For indoor positioning in this works, one of the implemented solutions is that based on Pozyx[1] UWB positioning system. The system constitutes of anchor modules marked as A, B, C and D in Fig. A.2 and tags marked as P and Q in Fig. A.2. For positioning, at

[1]https://www.pozyx.io

Table A.3: Guidance system specifications.

Property	**Value**
Velocity measurement range	0-16 m/s
Velocity measurement accuracy	0.04 m/s
Ranging accuracy	0.05 m
Range	0.20 m-20 m
Weight	0.28 kg

least one tag is rigidly attached to the platform whose position is desired and the anchors placed at known locations around the operating environment to support localization of the tag by multilateration. Pozyx system specifications are indicated in Table A.4.

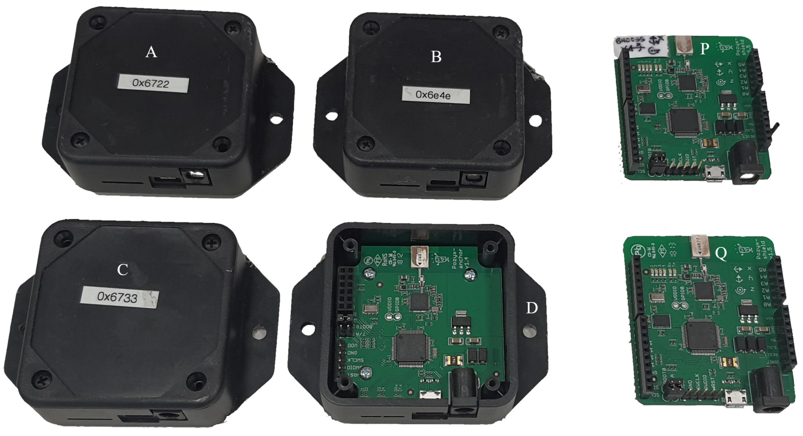


Figure A.2: Pozyx positioning system hardware. A-D are anchors and P-Q are tags.

Table A.4: Pozyx system specifications.

Property	**Value**
Power	5V via Micro USB, 4.5 V - 12 V via DC jack
Weight	0.012 kg
Dimension	0.06 m $\times$ 0.053 m $\times$ 0.026 m
Interface	I2C, micro USB
Frequency range	3.5 GHz-6.5 GHz
Bitrate	Up to 6.8 Mbps
Tri-axial gyroscope	yes
Tri-axial accelerometer	yes
Tri-axial compass	yes
Pressure sensor	yes

F450 Quadcopter

This is also an ×-configuration quadcopter, whose chassis and propulsion system are developed by SZ DJI Technology Co., Ltd. Unlike Matrice 100, the F450 quadcopter is fitted with a custom made STM32 Nucleo-64-F303RE [184] board-based flight management system. This autopilot is supported by five ultrasonic rangefinders for obstacle detection and AGL measurement, a 10DOF IMU and a Pozyx positioning tag. Fig. A.3 shows the Pozyx tag mounted onto the STM32 Nucleo-64 board's Arduino interface to provide position information. The key specifications of this vehicle are listed in Table A.5.

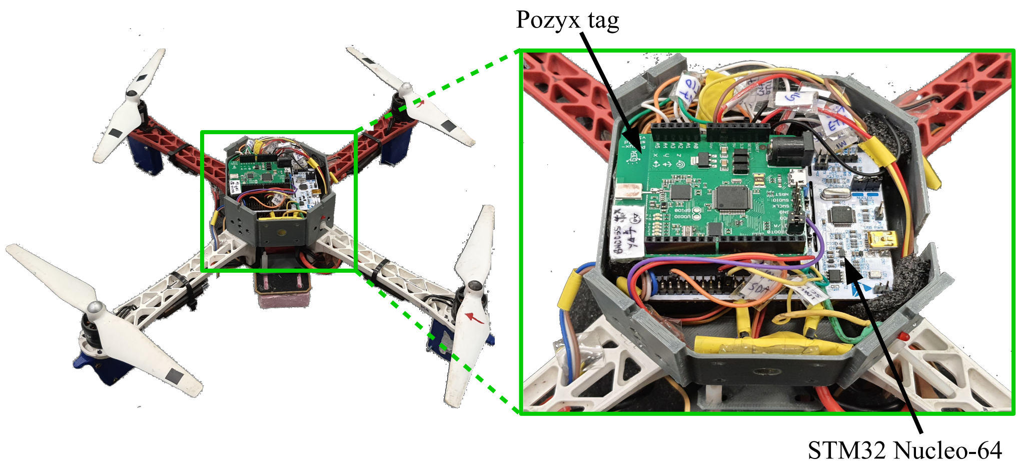


Figure A.3: Pozyx positioning tag mounted onto STM32 Nucleo-64 board.

Table A.5: F450 quadcopter specifications.

Property	**Value**
Diagonal wheelbase (motor to motor)	450 mm
Maximum diagonal wheelbase	710 mm
Maximum pitch and roll angular velocity	180 °/s
Maximum yaw angular velocity	90 °/s
Maximum pitch and roll angle	15 °
Maximum propeller thrust	0.850 kg/rotor
Maximum take-off weight	1.600 kg
ESC signal frequency	30 Hz - 450 Hz
Battery capacity	2200 mAh

Appendix B

Autonomy Evaluation Case Studies

To evaluate the applicability of the non-contextual part of the proposed autonomy assessment framework, it has been applied to the six unmanned aircraft system case studies presented in [6]. For the interest of completeness, the six vehicles are described in Table B.1. In Table B.2, are the classification results from ten existing level of autonomy frameworks as presented in [6]. The table was adapted and augmented with the classification results of applying the proposed level of autonomy chart in Table 4.6 to classification of the six unmanned aircraft systems of Table B.1.

Table B.1: Unmanned aircraft case studies from [6].

Vehicle	**Description**
UAS A	Unsupervised mission execution, operator-machine collaborative mission planning.
UAS B	Continuous operator input during execution, operator mission planning.
UAS C	Machine execution, continued operator supervision, operator mission planning and replanning, deterministic behaviour, operator mission planning.
UAS D	Machine mission planning and replanning, machine mission execution, operator supervised behaviour, non-deterministic behaviour.
UAS E	Machine mission planning and replanning, machine mission execution, unsupervised behaviour, immutable operator set goals and constraints.
UAS F	Machine mission planning and replanning, machine mission execution, unsupervised behaviour, machine may change goals and constraints during mission execution.

Table B.2: Level of autonomy evaluation with existing frameworks and the proposed framework. With the exception of the last row, the rest of the table is adapted from [6]. In the table, the autonomy levels range from I-XI, which correspond to levels 0-10.

Autonomy Scale	**UAS A**	**UAS B**	**UAS C**	**UAS D**	**UAS E**	**UAS F**
Sheridan	I	I	I	VIII	IX	X
Riley	XII	I	VII, VIII	IX	XI	XII
Billings	VII	I, II	III, IV, V	V, VI	VII	VII
Endsley	III	I, II	II, III	NC	IX, X	X
Clough	II	I	III	NC	NC	X, XI
Taylor	-	I	NC	IV, V, VI	-	-
Huang	NC	I	NC	X	XI	-
Galster	VIII	I	V	VI, VII	VIII	VIII
Kendoul	II	I	NC	NC	NC	XI
USDoD	-	I	III	III, IV	-	-
Proposed one	V	I	IX	VI	XI	XI

Bibliography

[1] M. Mazur, A. Wisniewski, J. McMillan, *et al.*, “Clarity from above: Pwc global report on the commercial applications of drone technology,” *Warsaw: Drone Powered Solutions, PriceWater house Coopers*, 2016.

[2] M. Labbé and F. Michaud, “Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[3] Y. H. Gao, D. Zhao, and Y. B. Li, “Uav sensor fault diagnosis technology: A survey,” in *Applied Mechanics and Materials*, vol. 220, pp. 1833–1837, Trans Tech Publ, 2012.

[4] G. Heredia, A. Ollero, R. Mahtani, M. Béjar, V. Remuß, and M. Musial, “Detection of sensor faults in autonomous helicopters,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2229–2234, IEEE, 2005.

[5] C. W. Lytle, “Job evaluation methods.,” 1946.

[6] R. Clothier, B. Williams, and T. Perez, “A review of the concept of autonomy in the context of the safety regulation of civil unmanned aircraft systems,” in *Proceedings of the Australian System Safety Conference 2013 (ASSC 2013)[Conferences in Research and Practice in Information Technology (CRPIT), Conferences in Research and Practice in Information Technology]*, pp. 15–27, Australian Computer Society Inc., 2014.

[7] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, “A compilation of uav applications for precision agriculture,” *Computer Networks*, vol. 172, p. 107148, 2020.

[8] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges,” *Ieee Access*, vol. 7, pp. 48572–48634, 2019.

[9] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots.* MIT press, 2011.

[10] G. Schmidt, “Gps based navigation systems in difficult environments,” *Gyroscopy and Navigation*, vol. 10, no. 2, pp. 41–53, 2019.

[11] F. Vanegas, K. J. Gaston, J. Roberts, and F. Gonzalez, “A framework for uav navigation and exploration in gps-denied environments,” in *2019 IEEE Aerospace Conference*, pp. 1–6, IEEE, 2019.

[12] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, and A. Savvaris, “Lidar-inertial integration for uav localization and mapping in complex environments,” in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 649–656, IEEE, 2016.

[13] S. M. Sánchez-Naranjo, N. G. Ferrara, M. J. Paśnikowski, J. Raasakka, E. Shytermeja, R. Ramos-Pollán, F. A. G. Osorio, D. Martínez, E.-S. Lohan, J. Nurmi, *et al.*, "Gnss vulnerabilities," in *Multi-Technology Positioning*, pp. 55–77, Springer, 2017.

[14] M. Cuntz, A. Konovaltsev, A. Dreher, and M. Meurer, "Jamming and spoofing in gps/gnss based applications and services–threats and countermeasures," in *Future Security Research Conference*, pp. 196–199, Springer, 2012.

[15] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, "A survey and analysis of the gnss spoofing threat and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–31, 2016.

[16] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[17] D. Chen and G. X. Gao, "Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation," *Navigation*, vol. 66, no. 1, pp. 151–168, 2019.

[18] G. Zhang and L.-T. Hsu, "Intelligent gnss/ins integrated navigation system for a commercial uav flight control system," *Aerospace science and technology*, vol. 80, pp. 368–380, 2018.

[19] J. Duo and L. Zhao, "Uav autonomous navigation system for gnss invalidation," in *2017 36th Chinese Control Conference (CCC)*, pp. 5777–5782, IEEE, 2017.

[20] M. Majidi, A. Erfanian, and H. Khaloozadeh, "A new approach to estimate true position of unmanned aerial vehicles in an ins/gps integration system in gps spoofing attack conditions," *International Journal of Automation and Computing*, vol. 15, no. 6, pp. 747–760, 2018.

[21] F. Andert, N. Ammann, S. Krause, S. Lorenz, D. Bratanov, and L. Mejias, "Optical-aided aircraft navigation using decoupled visual slam with range sensor augmentation," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 547–565, 2017.

[22] S. Zahran, A. Moussa, N. El-Sheimy, and A. B. Sesay, "Hybrid machine learning vdm for uavs in gnss-denied environment," *Navigation: Journal of The Institute of Navigation*, vol. 65, no. 3, pp. 477–492, 2018.

[23] S. international, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE*, 2018.

[24] E. Jackson, O. Williams, and K. Buchan, "Achieving robot autonomy," in *Defence Research Establishment Suffield, Proceedings of the 3 rd Conference on Military Robotic Applications p 242-248(SEE N 94-30275 08-37)*, 1991.

[25] B. T. Clough, "Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway," tech. rep., Air Force Research Lab Wright-Patterson AFB OH, 2002.

[26] A. Lampe and R. Chatila, "Performance measure for the evaluation of mobile robot autonomy," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 4057–4062, IEEE, 2006.

[27] C. C. Insaurralde and D. M. Lane, "Autonomy-assessment criteria for underwater vehicles," in *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pp. 1–8, IEEE, 2012.

[28] F. Kendoul, "Towards a unified framework for uas autonomy and technology readiness assessment (atra)," in *Autonomous Control Systems and Vehicles*, pp. 55–71, Springer, 2013.

[29] M. Elbanhawi, A. Mohamed, R. Clothier, J. Palmer, M. Simic, and S. Watkins, "Enabling technologies for autonomous mav operations," *Progress in Aerospace Sciences*, vol. 91, pp. 27–52, 2017.

[30] H.-M. Huang, E. Messina, and J. Albus, "Autonomy levels for unmanned systems (alfus) framework, volume ii."

[31] A. Doboli, D. Curiac, D. Pescaru, S. Doboli, W. Tang, C. Volosencu, M. Gilberti, O. Banias, and C. Istin, "Cities of the future: Employing wireless sensor networks for efficient decision making in complex environments," *arXiv preprint arXiv:1808.01169*, 2018.

[32] T. B. Curtin, D. M. Crimmins, J. Curcio, M. Benjamin, and C. Roper, "Autonomous underwater vehicles: trends and transformations," *Marine Technology Society Journal*, vol. 39, no. 3, pp. 65–75, 2005.

[33] Y. Bi, M. Lan, J. Li, K. Zhang, H. Qin, S. Lai, and B. M. Chen, "Robust autonomous flight and mission management for mavs in gps-denied environments," in *2017 11th Asian Control Conference (ASCC)*, pp. 67–72, IEEE, 2017.

[34] J. Vautherin, S. Rutishauser, K. Schneider-Zapp, H. F. Choi, V. Chovancova, A. Glass, and C. Strecha, "Photogrammetric accuracy and modeling of rolling shutter cameras.," *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 3, no. 3, 2016.

[35] A. R. Kuroswiski, N. M. F. de Oliveira, and É. H. Shiguemori, "Autonomous long-range navigation in gnss-denied environment with low-cost uav platform," in *2018 Annual IEEE International Systems Conference (SysCon)*, pp. 1–6, IEEE, 2018.

[36] R. Mebarki, V. Lippiello, and B. Siciliano, "Nonlinear visual control of unmanned aerial vehicles in gps-denied environments," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1004–1017, 2015.

[37] J. Unicomb, L. Dantanarayana, J. Arukgoda, R. Ranasinghe, G. Dissanayake, and T. Furukawa, "Distance function based 6dof localization for unmanned aerial vehicles in gps denied environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5292–5297, IEEE, 2017.

[38] A. R. Vetrella, A. Savvaris, G. Fasano, and D. Accardo, "Rgb-d camera-based quadrotor navigation in gps-denied and low light environments using known 3d markers," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 185–192, IEEE, 2015.

[39] H. Goforth and S. Lucey, "Gps-denied uav localization using pre-existing satellite imagery," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2974–2980, IEEE, 2019.

[40] M. Shan, F. Wang, F. Lin, Z. Gao, Y. Z. Tang, and B. M. Chen, "Google map aided visual navigation for uavs in gps-denied environment," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 114–119, IEEE, 2015.

[41] F. Vanegas Alvarez and F. Gonzalez, "Uncertainty based online planning for uav target finding in cluttered and gps-denied environments," in *Proceedings of the 2016 IEEE Aerospace Conference:*, pp. 706–714, IEEE, 2016.

[42] F. de Babo Martins, L. F. Teixeira, and R. Nóbrega, "Visual-inertial based autonomous navigation," in *Robot 2015: Second Iberian Robotics Conference*, pp. 561–572, Springer, 2016.

[43] F. Andert, J. Dittrich, S. Batzdorfer, M. Becker, U. Bestmann, and P. Hecker, "A flight state estimator that combines stereo-vision, ins, and satellite pseudo-ranges," in *Advances in Aerospace Guidance, Navigation and Control*, pp. 277–296, Springer, 2013.

[44] S. J. Dumble and P. W. Gibbens, "Airborne vision-aided navigation using road intersection features," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 185–204, 2015.

[45] S. Wang and T. Hu, "Ros-gazebo supported platform for tag-in-loop indoor localization of quadrocopter"," in *Intelligent Autonomous Systems 14* (W. Chen, K. Hosoda, E. Menegatti, M. Shimizu, and H. Wang, eds.), (Cham), pp. 185–197, Springer International Publishing, 2017.

[46] V. Walter, T. Novák, and M. Saska, "Self-localization of unmanned aerial vehicles based on optical flow in onboard camera images," in *Modelling and Simulation for Autonomous Systems* (J. Mazal, ed.), (Cham), pp. 106–132, Springer International Publishing, 2018.

[47] S. J. Dumble and P. W. Gibbens, "Efficient terrain-aided visual horizon based attitude estimation and localization," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 205–221, 2015.

[48] T. Nguyen, G. K. Mann, R. G. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 217–240, 2016.

[49] X. Liu, X. Tao, Y. Duan, and N. Ge, "Visual information assisted uav positioning using priori remote-sensing information," *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 14461–14480, 2018.

[50] X. W. Leong and H. Hesse, "Vision-based navigation for control of micro aerial vehicles," in *IRC-SET 2018*, pp. 413–427, Springer, 2019.

[51] F. Kendoul, K. Nonami, I. Fantoni, and R. Lozano, "An adaptive vision-based autopilot for mini flying machines guidance, navigation and control," *Autonomous robots*, vol. 27, no. 3, p. 165, 2009.

[52] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza, "Low computational-complexity algorithms for vision-aided inertial navigation of micro aerial vehicles," *Robotics and Autonomous Systems*, vol. 69, pp. 80–97, 2015.

[53] T. T. Mac, C. Copot, R. De Keyser, and C. M. Ionescu, "The development of an autonomous navigation system with optimal control of an uav in partly unknown indoor environment," *Mechatronics*, vol. 49, pp. 187–196, 2018.

[54] P. Lutz, M. G. Müller, M. Maier, S. Stoneman, T. Tomić, I. von Bargen, M. J. Schuster, F. Steidle, A. Wedler, W. Stürzl, *et al.*, "Ardea—an mav with skills for

future planetary missions," *Journal of Field Robotics*, vol. 37, no. 4, pp. 515–551, 2020.

[55] Y. Tang, Y. Hu, J. Cui, F. Liao, M. Lao, F. Lin, and R. S. Teo, "Vision-aided multi-uav autonomous flocking in gps-denied environment," *IEEE Transactions on industrial electronics*, vol. 66, no. 1, pp. 616–626, 2018.

[56] C. Wang, T. Wang, J. Liang, Y. Chen, Y. Zhang, and C. Wang, "Monocular visual slam for small uavs in gps-denied environments," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 896–901, IEEE, 2012.

[57] X. Zhang, B. Xian, B. Zhao, and Y. Zhang, "Autonomous flight control of a nano quadrotor helicopter in a gps-denied environment using on-board vision," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6392–6403, 2015.

[58] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, 2019.

[59] F. Valenti, D. Giaquinto, L. Musto, A. Zinelli, M. Bertozzi, and A. Broggi, "Enabling computer vision-based autonomous navigation for unmanned aerial vehicles in cluttered gps-denied environments," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3886–3891, IEEE, 2018.

[60] J. Chudoba, M. Kulich, M. Saska, T. Báča, and L. Přeučil, "Exploration and mapping technique suited for visual-features based localization of mavs," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 351–369, 2016.

[61] C.-L. Wang, T.-M. Wang, J.-H. Liang, Y.-C. Zhang, and Y. Zhou, "Bearing-only visual slam for small unmanned aerial vehicles in gps-denied environments," *International Journal of Automation and Computing*, vol. 10, no. 5, pp. 387–396, 2013.

[62] R. C. Leishman, T. W. McLain, and R. W. Beard, "Relative navigation approach for vision-based aerial gps-denied navigation," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 97–111, 2014.

[63] J. Marzat, S. Bertrand, A. Eudes, M. Sanfourche, and J. Moras, "Reactive mpc for autonomous mav navigation in indoor cluttered environments: Flight experiments," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15996–16002, 2017.

[64] S. Yang, S. A. Scherer, X. Yi, and A. Zell, "Multi-camera visual slam for autonomous navigation of micro aerial vehicles," *Robotics and Autonomous Systems*, vol. 93, pp. 116–134, 2017.

[65] Y. Li, Y. Wang, and D. Wang, "Multiple rgb-d sensor-based 3-d reconstruction and localization of indoor environment for mini mav," *Computers & Electrical Engineering*, vol. 70, pp. 509–524, 2018.

[66] Q. Li, D.-C. Li, Q.-f. Wu, L.-w. Tang, Y. Huo, Y.-x. Zhang, and N. Cheng, "Autonomous navigation and environment modeling for mavs in 3-d enclosed industrial environments," *Computers in Industry*, vol. 64, no. 9, pp. 1161–1177, 2013.

[67] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, "Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *Journal of field robotics*, vol. 30, no. 3, pp. 415–438, 2013.

[68] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[69] K. Schmid, P. Lutz, T. Tomić, E. Mair, and H. Hirschmüller, "Autonomous vision-based micro air vehicle for indoor and outdoor navigation," *Journal of Field Robotics*, vol. 31, no. 4, pp. 537–570, 2014.

[70] F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero, "An architecture for robust uav navigation in gps-denied areas," *Journal of Field Robotics*, vol. 35, no. 1, pp. 121–145, 2018.

[71] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, vol. 37, no. 4, pp. 642–666, 2020.

[72] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, 2018.

[73] Z. Hou, J. Qi, and M. Wang, "Fusing optical flow and inertial data for uav motion estimation in gps-denied environment," in *2019 Chinese Control Conference (CCC)*, pp. 7791–7796, IEEE, 2019.

[74] R. Mebarki and V. Lippiello, "Image moments-based velocity estimation of uavs in gps denied environments," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pp. 1–6, IEEE, 2014.

[75] R. Mebarki, J. Cacace, and V. Lippiello, "Velocity estimation of an uav using visual and imu data in a gps-denied environment," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–6, IEEE, 2013.

[76] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *2009 International Conference on Advanced Robotics*, pp. 1–6, IEEE, 2009.

[77] J. Garcia-Pulido, G. Pajares, S. Dormido, and J. M. de la Cruz, "Recognition of a landing platform for unmanned aerial vehicles by using computer vision-based techniques," *Expert Systems with Applications*, vol. 76, pp. 152–165, 2017.

[78] J. Zhao, X. Ma, Q. Fu, C. Hu, and S. Yue, "An lgmd based competitive collision avoidance strategy for uav," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 80–91, Springer, 2019.

[79] A. Shabayek, C. Demonceaux, O. Morel, and D. Fofi, "Vision based uav attitude estimation: Progress and insights," *Journal of Intelligent & Robotic Systems*, vol. 65, pp. 295–308, 01 2012.

[80] Y. Gui, P. Guo, H. Zhang, Z. Lei, X. Zhou, J. Du, and Q. Yu, "Airborne vision-based navigation method for uav accuracy landing using infrared lamps," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 197–218, 2013.

[81] S. Dotenco, F. Gallwitz, and E. Angelopoulou, "Autonomous approach and landing for a low-cost quadrotor using monocular cameras," in *European Conference on Computer Vision*, pp. 209–222, Springer, 2014.

[82] M. K. Al-Sharman, B. J. Emran, M. A. Jaradat, H. Najjaran, R. Al-Husari, and Y. Zweiri, "Precision landing using an adaptive fuzzy multi-sensor data fusion architecture," *Applied soft computing*, vol. 69, pp. 149–164, 2018.

[83] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *Ifac-Papersonline*, vol. 50, no. 1, pp. 10488–10494, 2017.

[84] M. Bhargavapuri, A. K. Shastry, H. Sinha, S. R. Sahoo, and M. Kothari, "Vision-based autonomous tracking and landing of a fully-actuated rotorcraft," *Control Engineering Practice*, vol. 89, pp. 113–129, 2019.

[85] F. Liao, S. Lai, Y. Hu, J. Cui, J. L. Wang, R. Teo, and F. Lin, "3d motion planning for uavs in gps-denied unknown forest environment," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 246–251, IEEE, 2016.

[86] C. Shang, L. Cheng, Q. Yu, X. Wang, R. Peng, Y. Chen, H. Wu, and Q. Zhu, "Micro aerial vehicle autonomous flight control in tunnel environment," in *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, pp. 93–98, IEEE, 2017.

[87] T. Pavlenko, M. Schütz, M. Vossiek, T. Walter, and S. Montenegro, "Wireless local positioning system for controlled uav landing in gnss-denied environment," in *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pp. 171–175, IEEE, 2019.

[88] D. Scaramuzza and Z. Zhang, "Visual-inertial odometry of aerial robots," *arXiv preprint arXiv:1906.03289*, 2019.

[89] H. Qin, Y. Bi, K. Z. Ang, K. Wang, J. Li, M. Lan, M. Shan, and F. Lin, "A stereo and rotating laser framework for uav navigation in gps denied environment," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6061–6066, IEEE, 2016.

[90] D. Eynard, P. Vasseur, C. Demonceaux, and V. Frémont, "Uav altitude estimation by mixed stereoscopic vision," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 646–651, IEEE, 2010.

[91] W. Liu, D. Zou, D. Sartori, L. Pei, and W. Yu, "An image-guided autonomous navigation system for multi-rotor uavs," in *China Satellite Navigation Conference*, pp. 513–526, Springer, 2019.

[92] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 601–627, 2019.

[93] J. Madeiras, C. Cardeira, and P. Oliveira, "Vision-aided complementary filter for attitude and position estimation: Design, analysis and experimental validation," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 388–393, 2019.

[94] R. Murphy, "An introduction to ai robotics (intelligent robotics and autonomous agents)," *A Bradford Book*, 2000.

[95] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* MIT Press, 2005.

[96] X. Wan, J. Liu, H. Yan, and G. L. Morgan, “Illumination-invariant image matching for autonomous uav localisation based on optical sensing,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 119, pp. 198–213, 2016.

[97] J. Li-Chee-Ming and C. Armenakis, “Uav navigation system using line-based sensor pose estimation,” *Geo-spatial information science*, vol. 21, no. 1, pp. 2–11, 2018.

[98] A. Haque, A. Elsaharti, T. Elderini, M. A. Elsaharty, and J. Neubert, “Uav autonomous localization using macro-features matching with a cad model,” *Sensors*, vol. 20, no. 3, p. 743, 2020.

[99] A. L. Majdik, D. Verda, Y. Albers-Schoenberg, and D. Scaramuzza, “Air-ground matching: Appearance-based gps-denied urban localization of micro aerial vehicles,” *Journal of Field Robotics*, vol. 32, no. 7, pp. 1015–1039, 2015.

[100] P. DeFranco, J. D. Mackie, M. Morin, and K. F. Warnick, “Bio-inspired electromagnetic orientation for uavs in a gps-denied environment using mimo channel sounding,” *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 10, pp. 5250–5259, 2014.

[101] F. Vanegas and F. Gonzalez, “Enabling uav navigation with sensor and environmental uncertainty in cluttered and gps-denied environments,” *Sensors*, vol. 16, no. 5, p. 666, 2016.

[102] F. Causa, A. R. Vetrella, G. Fasano, and D. Accardo, “Multi-uav formation geometries for cooperative navigation in gnss-challenging environments,” in *2018 IEEE/ION position, location and navigation symposium (PLANS)*, pp. 775–785, IEEE, 2018.

[103] S. M. Siam and H. Zhang, “Fast-seqslam: A fast appearance based place recognition algorithm,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5702–5708, IEEE, 2017.

[104] E. López, R. Barea, A. Gómez, Á. Saltos, L. M. Bergasa, E. J. Molinos, and A. Nemra, “Indoor slam for micro aerial vehicles using visual and laser sensor fusion,” in *Robot 2015: Second Iberian Robotics Conference*, pp. 531–542, Springer, 2016.

[105] M. Nieuwenhuisen, D. Droeschel, M. Beul, and S. Behnke, “Autonomous navigation for micro aerial vehicles in complex gnss-denied environments,” *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 199–216, 2016.

[106] A. Bachrach, S. Prentice, R. He, and N. Roy, “Range–robust autonomous navigation in gps-denied environments,” *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

[107] E. T. Dill and M. Uijt de Haag, “3d multi-copter navigation and mapping using gps, inertial, and lidar,” *Navigation: Journal of The Institute of Navigation*, vol. 63, no. 2, pp. 205–220, 2016.

[108] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, *et al.*, “Fast, autonomous flight in gps-denied and cluttered environments,” *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[109] S. S. Mansouri, C. Kanellakis, D. Kominiak, and G. Nikolakopoulos, "Deploying mavs for autonomous navigation in dark underground mine environments," *Robotics and Autonomous Systems*, vol. 126, p. 103472, 2020.

[110] E. Hong and J. Lim, "Visual-inertial odometry with robust initialization and online scale estimation," *Sensors*, vol. 18, no. 12, p. 4287, 2018.

[111] C. Hui, C. Yousheng, and W. W. Shing, "Trajectory tracking and formation flight of autonomous uavs in gps-denied environments using onboard sensing," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, pp. 2639–2645, IEEE, 2014.

[112] K. Gryte, T. H. Bryne, S. M. Albrektsen, and T. A. Johansen, "Field test results of gnss-denied inertial navigation aided by phased-array radio systems for uavs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1398–1406, IEEE, 2019.

[113] S. Zahran, A. Moussa, and N. El-Sheimy, "Enhanced uav navigation in gnss denied environment using repeated dynamics pattern recognition," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 1135–1142, IEEE, 2018.

[114] M. Alnuaimi and M. G. Perhinschi, "Alternative approaches for uav dead reckoning based on the immunity paradigm," *Aerospace Science and Technology*, vol. 98, p. 105742, 2020.

[115] J. Tiemann, F. Schweikowski, and C. Wietfeld, "Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments," in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, IEEE, 2015.

[116] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia, "Experimental analysis of radar odometry by commercial ultralight radar sensor for miniaturized uas," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 3-4, pp. 485–503, 2018.

[117] A. Benini, A. Mancini, and S. Longhi, "An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 461–476, 2013.

[118] Q. Wang, M. Fu, Z. Deng, and H. Ma, "A comparison of loosely-coupled mode and tightly-coupled mode for ins/vms," in *2012 American Control Conference (ACC)*, pp. 6346–6351, IEEE, 2012.

[119] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, pp. 834–849, Springer, 2014.

[120] N. Yang, R. Wang, and D. Cremers, "Feature-based or direct: An evaluation of monocular visual odometry," *arXiv preprint arXiv:1705.04300*, pp. 1–12, 2017.

[121] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[122] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, "A survey on vision-based uav navigation," *Geo-spatial information science*, vol. 21, no. 1, pp. 21–32, 2018.

[123] R. Szeliski, *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

[124] H. Yu, F. Zhang, and P. Huang, "Cslam and gps based navigation for multi-uav cooperative transportation system," in *International Conference on Intelligent Robotics and Applications*, pp. 315–326, Springer, 2019.

[125] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Intuitive 3d maps for mav terrain exploration and obstacle avoidance," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 473–493, 2011.

[126] A. Al-Kaff, D. Martin, F. Garcia, A. de la Escalera, and J. M. Armingol, "Survey of computer vision algorithms and applications for unmanned aerial vehicles," *Expert Systems with Applications*, vol. 92, pp. 447–463, 2018.

[127] M. Warren, M. Paton, K. MacTavish, A. P. Schoellig, and T. D. Barfoot, "Towards visual teach and repeat for gps-denied flight of a fixed-wing uav," in *Field and Service Robotics*, pp. 481–498, Springer, 2018.

[128] H. D. Escobar-Alvarez, N. Johnson, T. Hebble, K. Klingebiel, S. A. Quintero, J. Regenstein, and N. A. Browning, "R-advance: Rapid adaptive prediction for vision-based autonomous navigation, control, and evasion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 91–100, 2018.

[129] H. Xie, K. H. Low, and Z. He, "Adaptive visual servoing of unmanned aerial vehicles in gps-denied environments," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2554–2563, 2017.

[130] A. Volkova and P. W. Gibbens, "More robust features for adaptive visual navigation of uavs in mixed environments," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 1-2, pp. 171–187, 2018.

[131] Y. Zhang, T. Wang, Z. Cai, Y. Wang, and Z. You, "The use of optical flow for uav motion estimation in indoor environment," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 785–790, IEEE, 2016.

[132] K. Li, C. Wang, S. Huang, G. Liang, X. Wu, and Y. Liao, "Self-positioning for uav indoor navigation based on 3d laser scanner, uwb and ins," in *2016 IEEE International Conference on Information and Automation (ICIA)*, pp. 498–503, IEEE, 2016.

[133] T.-M. Nguyen, T. H. Nguyen, M. Cao, Z. Qiu, and L. Xie, "Integrated uwb-vision approach for autonomous docking of uavs in gps-denied environments," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9603–9609, IEEE, 2019.

[134] V. O. Sivaneri and J. N. Gross, "Ugv-to-uav cooperative ranging for robust navigation in gnss-challenged environments," *Aerospace Science and Technology*, vol. 71, pp. 245–255, 2017.

[135] S. Wei, G. Dan, and H. Chen, "Altitude data fusion utilising differential measurement and complementary filter," *IET Science, Measurement & Technology*, vol. 10, no. 8, pp. 874–879, 2016.

[136] H.-M. Huang, K. Pavek, J. Albus, and E. Messina, "Autonomy levels for unmanned systems (alfus) framework: An update," in *Unmanned Ground Vehicle Technology VII*, vol. 5804, pp. 439–448, International Society for Optics and Photonics, 2005.

[137] R. P. Padhy, F. Xia, S. K. Choudhury, P. K. Sa, and S. Bakshi, "Monocular vision aided autonomous uav navigation in indoor corridor environments," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 96–108, 2018.

[138] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[139] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control.* Cambridge University Press, 2017.

[140] N. Correll, *Introduction to autonomous robots.* 2016.

[141] B. Siciliano and O. Khatib, *Springer handbook of robotics.* Springer, 2016.

[142] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning,"

[143] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," in *Motion and operation planning of robotic systems*, pp. 3–27, Springer, 2015.

[144] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, S. Thrun, and R. C. Arkin, *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[145] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2, pp. 995–1001, IEEE, 2000.

[146] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[147] S. A. Cambone, K. Krieg, P. Pace, and W. Linton, "Unmanned aircraft systems roadmap 2005-2030," *Office of the Secretary of Defense*, vol. 8, pp. 4–15, 2005.

[148] X. Qi, D. Theilliol, J. Qi, Y. Zhang, J. Han, D. Song, L. Wang, and Y. Xia, "Fault diagnosis and fault tolerant control methods for manned and unmanned helicopters: a literature review," in *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pp. 132–139, IEEE, 2013.

[149] P. Aboutalebi, A. Abbaspour, P. Forouzannezhad, and A. Sargolzaei, "A novel sensor fault detection in an unmanned quadrotor based on adaptive neural observer," *Journal of intelligent & robotic systems*, vol. 90, no. 3, pp. 473–484, 2018.

[150] R. Sun, Q. Cheng, G. Wang, and W. Y. Ochieng, "A novel online data-driven algorithm for detecting uav navigation sensor faults," *Sensors*, vol. 17, no. 10, p. 2243, 2017.

[151] E. Khalastchi, M. Kalech, G. A. Kaminka, and R. Lin, "Online data-driven anomaly detection in autonomous robots," *Knowledge and Information Systems*, vol. 43, no. 3, pp. 657–688, 2015.

[152] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 5266–5271, IEEE, 2015.

[153] C. Li and S. L. Waslander, “Visual measurement integrity monitoring for uav localization,” in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 22–29, IEEE, 2019.

[154] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[155] J. C. Mankins, “Technology readiness assessments: A retrospective,” *Acta Astronautica*, vol. 65, no. 9-10, pp. 1216–1223, 2009.

[156] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles.* Springer Science & Business Media, 2010.

[157] Y. Li, M. Scanavino, E. Capello, F. Dabbene, G. Guglieri, and A. Vilardi, “A novel distributed architecture for uav indoor navigation,” *Transportation research procedia*, vol. 35, pp. 13–22, 2018.

[158] K. Dalamagkidis, “Classification of uavs,” *Handbook of unmanned aerial vehicles*, pp. 83–91, 2015.

[159] P. van Blyenburgh, “Uav systems: global review,” in *Conference, Amsterdam, The Netherlands*, 2006.

[160] K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl, *On integrating unmanned aircraft systems into the national airspace system: issues, challenges, operational restrictions, certification, and recommendations*, vol. 54. springer science & Business Media, 2011.

[161] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned rotorcraft systems.* Springer Science & Business Media, 2011.

[162] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems.* John Wiley & Sons, 2015.

[163] E. U. A. S. Agency, “Easy access rules for unmanned aircraft systems (regulations (eu) 2019/947 and (eu) 2019/945),” 2021.

[164] P. J. Durst and M. W. Gray, “Levels of autonomy and autonomous system performance assessment for intelligent unmanned systems,” 2014.

[165] W. R. Ashby, *An introduction to cybernetics.* Chapman & Hall Ltd, 1961.

[166] “Darpa subterranean challenge: Competition rules final event.”

[167] J. Bruce and M. M. Veloso, “Real-time randomized path planning for robot navigation,” in *Robot Soccer World Cup*, pp. 288–295, Springer, 2002.

[168] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[169] I. A. Şucan and L. E. Kavraki, “A sampling-based tree planner for systems with complex dynamics,” *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2011.

[170] D. Ferguson and A. Stentz, “Anytime rrts,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5369–5375, IEEE, 2006.

[171] D. Devaurs, T. Siméon, and J. Cortés, “Enhancing the transition-based rrt to deal with complex cost spaces,” in *2013 IEEE International Conference on Robotics and Automation*, pp. 4120–4125, IEEE, 2013.

[172] I. A. Şucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012. https://ompl.kavrakilab.org.

[173] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[174] A. Mora, S. Vemprala, A. Carrio, and S. Saripalli, “Flight performance assessment of land surveying trajectories for multiple uav platforms,” in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pp. 1–7, IEEE, 2015.

[175] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[176] C. Di Franco and G. Buttazzo, “Coverage path planning for uavs photogrammetry with energy and resolution constraints,” *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3, pp. 445–462, 2016.

[177] C. Di Franco and G. Buttazzo, “Energy-aware coverage path planning of uavs,” in *2015 IEEE international conference on autonomous robot systems and competitions*, pp. 111–117, IEEE, 2015.

[178] W. H. Huang, “Optimal line-sweep-based decompositions for coverage algorithms,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 1, pp. 27–32, IEEE, 2001.

[179] E. A. Wan and R. Van Der, “The unscented kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pp. 153–158, Ieee, 2000.

[180] J. Shi and C. Tomasi, “Good features to track (tech. rep.),” *Ithaca, NY, USA*, 1993.

[181] B. D. Lucas, T. Kanade, *et al.*, “An iterative image registration technique with an application to stereo vision,” Vancouver, British Columbia, 1981.

[182] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*, pp. 778–792, Springer, 2010.

[183] “Mpu-9255 product specification.” https://www.waveshare.com/w/upload/0/01/PS-MPU-9255.pdf, 9 2014. Revision 1.0.

[184] N. STMicroelectronics, “Stm32 nucleo-64 boards,” 2018.