Concept and Design of a Cooperative Robotic Assistant Surgery System

Vom Fachbereich Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

von

M.Sc. Raúl Armando Castillo Cruces

1. Gutachter: Prof. Dr.-Ing. Hubert Roth
2. Gutachter: Prof. Dr.-Ing. Heinz Wörn

Vorsitzender: Prof. Dr. Ing. Otmar Loffeld

Tag der mündlichen Prüfung: 28. Juli, 2008

Para mis padres

ACKNOWLEDGEMENT

Firstly, and most importantly, I want to thank God for introducing me to all the various people I have encountered and who have contributed in many different way to make my life so enjoyable during these last few years. I want to thank my family, from the bottom of my heart, for giving me unconditional support in order for me to achieve my goals, even with the big distance between us. They have shown me that, no matter what, my family will always be together. This work would have not been possible without the cooperative support from Mexico, through the National Counsel of Science and Technology (CONACYT), and Germany, through the German Academic Exchange Service (DAAD). I wish to thank all the people at the University of Siegen and the Center for Sensor Systems (ZESS), in particular Prof. Otmar Loffeld, Dr. Stefan Knedlik, Renate Szabó, to all IPP members and staff of the Institute. All of whom created a very pleasant and supportive work environment and greatly helped in my professional development. Thank you very much to Prof. Hubert Roth, the director of the Institute of Control Engineer (RTS) at the University of Siegen, and Prof. Heinz Wörn, the head of the Institute for Process Control and Robotics at the University of Karlsruhe, for supervising me and giving me valuable advice right up to the competition of my work. Special thanks must be made to Dr. Jürgen Wahrburg, the head of the modiCAS project, a wonderful person who guided and helped me so many different ways in order to achieve my objectives. To my colleges Peter Knappe, Sebastian Pieck, Hans-Christian Schneider, Marc Schlimbach, Stephanie Sahm, Domink Scarpin, Steffen Heuel, Jens Bernshausen and Hamza Hassan Shah, a big thank you, it was a pleasure working with you all. I would also like to thank Katharina Selle, John O'Byrne and Petra Nachtigäller for all the time they took to review my texts. In particular, I want to thank Nina Claasen for giving me so much of her time and encouragement during the whole dissertation. And last but not least, I want to express all my gratefulness to all the good friends, Anja Reuter, Daniel Pfeiffer, Jürgen Sorg, Kerstin Bohr, Ricardo Bautista, Hannah Müllners, Michaela Thiel, Meike Schmitz, Horacio Martinez, Manuela Franke, Claudia Leonardi, Javier Kafie, Daniel Hunziker, Eduardo Azcué, Volkmar Böttger and Julia Neuser, among a lot more who made me feel at home in Germany.

CONTENTS

ACKNOWLEDGEMENT i

CONTENTS ii

LIST OF TABLES vi

LIST OF FIGURES vii

ABBREVIATIONS xi

NOMENCLATURES xv

ABSTRACT xvii

1. Introduction 1

2. Motivation to design a cooperative robotic assistant surgery system 3

    2.1 The modiCAS project 3

        2.1.1 Combination of navigation and robotics 4

        2.1.2 Human-robot interaction 5

    2.2 System demands 6

        2.2.1 Controller software architecture 7

        2.2.2 Human-robot interaction 8

    2.3 Technical challenges 9

        2.3.1 Controller redesign 9

        2.3.2 Robot singularities 10

3. State of the art 12

    3.1 Surgical robotic systems 12

        3.1.1 Autonomous system 14

        3.1.2 Cooperative systems 18

        3.1.3 Teleoperative systems 23

    3.2 Virtual fixtures 28

    3.3 Singularity robustness 30

4. Design of a controller framework 33

4.1 System architecture 33
4.2 Command interface 34
4.2.1 TCP/IP message protocol 36
4.2.2 Command library 37
4.2.3 Command sender module 40
4.2.4 Command parser module 41
4.2.5 Data transmitter module 43
4.2.6 Data receiver module 44
4.3 Target computer 45
4.3.1 State machine 47
4.3.2 Task manager 49
4.3.3 Task-loops 50
4.3.4 State manager 52
4.4 Hardware interface selector 53
4.5 Target functions 54
4.5.1 Joint velocity controller 54
4.5.2 Cartesian velocity controller 55
4.5.3 Joint position controller 56
4.5.4 Cartesian position controller 57
4.5.5 Guidance virtual fixture controller 58
4.5.6 Patient tracking controller 59
4.6 Host computer 61
4.6.1 GUI producer 61
4.6.2 External producer 62
4.6.3 Event consumer 62
4.6.4 GUI handler 62
4.7 Modular distribution 63

4.8 Execution and task modules 65
5. Human-robot cooperation 67
5.1 Hands-on interface 67
5.2 Virtual fixtures description 69
5.3 Admittance controller 72
5.4 Deviation error 74
5.5 Boundary conditions 76
5.6 Manual error compensation 77
5.7 Autonomous error compensation 78
5.8 Virtual fixtures classes 79
5.8.1 Reference target 79
5.8.2 Move along an axis 80
5.8.3 Rotate around an axis 81
5.8.4 Move along a plane 82
5.8.5 Rotate around two axes 83
5.8.6 Extension to volume 84
5.8.7 Reference trajectory 87
5.9 Experimental evaluation 89
5.9.1 Manual error compensation 89
5.9.2 Manual compensation vs. autonomous compensation 92
5.9.3 Moving along a trajectory 98
5.10 Discussion 101
6. Singularity robustness 103
6.1 PA10 kinematics 103
6.1.1 Forward position kinematics 104
6.1.2 Forward velocity kinematics 107
6.1.3 Singularities of PA10-6C 109

6.2 Differential kinematics inversion ... 113

6.3 Damped least squares approach ... 113

6.3.1 Singular value decomposition of the damped least squares ... 114

6.3.2 The damping factor ... 115

6.4 Adjoint Jacobian approach ... 116

6.4.1 Relationship between adjoint Jacobian and the null vector ... 117

6.4.2 Velocity relations at singularity ... 119

6.4.3 Selection of scalar variable *b* ... 120

6.4.4 Selecting sign variable σ ... 122

6.4.5 Wrist singularity ... 122

6.5 Experimental results ... 124

6.5.1 Passing through singularity ... 125

6.5.2 Escaping singularity along singular direction ... 129

6.5.3 Passing in the neighborhood of singularity ... 135

6.6 Discussion ... 139

7. Conclusions ... 140

References ... 142

A Quaternion ... 154

B Spline functions ... 156

C Quadratic minimization of a cubic spline function ... 159

D Singular value decomposition ... 161

LIST OF TABLES

Table 3.1. Complementary capabilities of human and surgical robots............................ 12
Table 3.2. Clinical application areas and representative robotic developments.............. 14
Table 4.1. List of commands for general purposes functions.. 38
Table 5.1. Mean value of position and orientation error of translational VF 94
Table 5.2. Mean value of position and orientation error of rotational VF........................ 97
Table 5.3. Mean error while following a reference trajectory... 98
Table 6.1. Maximum and mean error when passing through wrist singularity 127
Table 6.2. Maximum and mean error when passing in the neighborhood of wrist singularity 135

LIST OF FIGURES

Figure 2.1. Components of the navigated robotic system for surgical assistance 3
Figure 2.2. modiCAS system in the operating room 6
Figure 3.1. Autonomous System: The robot executes the procedure while surgeon observes/supervises the operation [94] 15
Figure 3.2. (a)ORTHODOC planning workstation, (b) ROBODOC milling implant cavity for hip replacement surgery (courtesy of ISS, USA) 16
Figure 3.3. CASPAR system in knee operation (courtesy of ortoMaquet) 17
Figure 3.4. CyberKnife robotic radiosurgery system (courtesy of Accuray, USA) 18
Figure 3.5. Neuromate courtesy of Integrated Surgical Systems 19
Figure 3.6. Cooperative: robot and surgeon remains jointly in control [94] 20
Figure 3.7. ACROBOT, special purpose Hands-On robot for knee surgery 20
Figure 3.8. JHU Steady Hand robot for microsurgery.. 21
Figure 3.9. Free-wheel mechanism of PADyC 22
Figure 3.10. Telesurgery system: Surgeon controls the robot in real-time through the haptic interface [94] 23
Figure 3.11. (a) daVinci telesurgery system, (b) Endoscopic EndoWrist^{TM} Instrument (courtesy of Intuitive Surgical) 24
Figure 3.12. Zeus Telesurgery system from Computer Motion Inc.: (a) Console unit, (b) Zeus robot arms 25
Figure 3.13. Setup of Zeus system at Lindbergh operation 2001 [102] 26
Figure 3.14. The telesurgical workstation for laparoscopy at Berkley [90] 26
Figure 3.15. The telerobotic system for mircrosurgery at KAIST 27
Figure 4.1. Command-based architecture of modiCAS framework 35
Figure 4.2. Example of MDL of commands with two instances of the same command. 37
Figure 4.3. Transmission Packet Format 37
Figure 4.4. Command internal structure 39
Figure 4.5 Interaction between command and command sender module 40
Figure 4.6. Command parser module: A PTP(q) command is received, but it is only processed if the current state is comprises the command 42
Figure 4.7. Data distribution inside the data transmitter module 43

Figure 4.8. Data receiver loop at a Host application 44
Figure 4.9. Task distribution among the different threads in RT-Target application...... 46
Figure 4.10. State machine of RT target... 48
Figure 4.11 - Task loop flowchart 50
Figure 4.12. Flow diagram of Joint Velocity Controller (JVEL) 55
Figure 4.13. Flow diagram of Cartesian Velocity Controller (CVEL)........................... 56
Figure 4.14. Flow diagram of Point to Point joint position controller (PTP)................. 57
Figure 4.15. Flow diagram of linear trajectory Cartesian position controller (LIN)....... 58
Figure 4.16. Flow diagram of Guidance Virtual Fixture controller (GVF).................... 59
Figure 4.17. Coordinate systems used for tracking controller.. 60
Figure 4.18. Flow diagram of patient tracking controller (TRK) 60
Figure 4.19. Host application.. 63
Figure 4.20. Graphical User Interface of the client running at host computer 65
Figure 5.1. Handle system with rapid tool-exchange mechanism. 67
Figure 5.2 Flowchart of High Priority Loop for Force Torque Sensor data acquisition and processing (HPL-FTS) 68
Figure 5.3. Projection onto the subspaces of preferred directions *U* and of non-preferred directions *V* 70
Figure 5.4. Control loop for cooperative robot system... 71
Figure 5.5. System reference frames 73
Figure 5.6. Reference target.. 79
Figure 5.7. Move along a line 81
Figure 5.8. Rotate around one axis 82
Figure 5.9. Extension to plane 83
Figure 5.10. Rotate around two axes 84
Figure 5.11. Virtual cylinder 85
Figure 5.12 Virtual Cone 87
Figure 5.13 Closest point of spline curve to p_{TCP} and its tangent vector. 88
Figure 5.14. Influence of gain k_d on manual error compensation of end-effector position along Z-axis while moving it along Y-axis w.r.t. world coordinates. 90

Figure 5.15. Influence of gain k_d on manual error compensation of end-effector orientation while moving it along Y-axis w.r.t. world coordinates. 91
Figure 5.16. Influence of gain k_d on the error norm of manual compensation while moving the end-effector along Y-axis w.r.t. world coordinates. 91
Figure 5.17. End-effector position while moving along the Y-axis w.r.t. world coordinates. 92
Figure 5.18 End-effector orientation while moving along the Y-axis w.r.t. world coordinates. 93
Figure 5.19. Error profile of manual and autonomous error compensation while moving along the Y-axis w.r.t. world coordinates. 93
Figure 5.20. Experimental setup for rotational case 95
Figure 5.21. Tool tip position in 3D while pivot rotation above Z axis 96
Figure 5.22. Tool tip position in 3D while pivot rotation above Z axis 96
Figure 5.23. End-effector error profile while pivot rotation above Z axis 97
Figure 5.24. End-effector position while following a sinusoidal trajectory 99
Figure 5.25. End-effector error profile while following a sinusoidal trajectory 99
Figure 5.26. End-effector position while following a circular trajectory 100
Figure 5.27. End-effector error profile while following a circular trajectory 100
Figure 6.1. (a) PA10-6C robot arm (courtesy of Mitsubishi Heavy Industries), (b) Kinematic description of PA10-6C based on Featherstone [15] 103
Figure 6.2. Kinematics of first three joints of the 321 manipulator [15] 104
Figure 6.3.Singular configurations: (a) arm-extended, (b) wrist-extended, (c) wrist-above-shoulder [15] 110
Figure 6.4. Singular direction and its orthogonal plane at wrist singularity.................. 112
Figure 6.5. End-effector movement along (a) direction orthogonal to singular direction, and (b) singular direction 125
Figure 6.6. End-effector position while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame 126
Figure 6.7. End-effector orientation while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame 126

Figure 6.8. End-effector error profile while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame 127
Figure 6.9. Position of joints θ_4, θ_5 and θ_6 while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame 128
Figure 6.10. Velocity of θ_4, θ_5 and θ_6 while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame 128
Figure 6.11. End-effector position while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 130
Figure 6.12. End-effector orientation while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 131
Figure 6.13. End-effector error profile while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 131
Figure 6.14. Position of robot joints while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 133
Figure 6.15. Velocity of joints θ_4, θ_5 and θ_6 while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 134
Figure 6.16. Position of joints θ_4, θ_5 and θ_6 while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame 134
Figure 6.17. End-effector position while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame 136
Figure 6.18. End-effector orientation while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame 136
Figure 6.19. End-effector error profile while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame 137
Figure 6.20. Position of joints θ_4, θ_5 and θ_6 while passing near singularity with virtual fixture along Y-axis w.r.t. the base frame 138
Figure 6.21. Velocity of joints θ_4, θ_5 and θ_6 while passing near singularity with virtual fixture along Y-axis w.r.t. the base frame 138
Figure B.1. Segment wise parametric curve………………………………………...156
Figure C.1. Closest point of spline curve to p0 and its tangent vector………………...159

ABBREVIATIONS

API	Application Programming Interface: is a source code interface that an operating system, library or service provides to support requests made by a program.
C/C++	C is a high-level programming language, while C++ adds object-oriented features to C.
CAS	Computer Assisted Surgery used to help a surgeon in defining and executing an optimal surgical strategy based on a variety of multimodal data inputs.
CORBA	Common Object Request Broker Architecture: is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together
CMD-DONE	Command Done: Acknowledgment transmitted from target computer to host computer after an incoming command is successfully processed.
CMD-FAIL	Command Failed: Acknowledgment transmitted from target computer to host computer each time an incoming command fails.
CPM	Command Parser Module: One of four modules used in the command interface within the modiCAS system designed for efficient host-target communication. The CPM receives and process incoming commands at the target computer.
CPU	Central Processing Unit: also called processor, is the computing part of a computer.
CSM	Command Sender Module: One of four modules used in the command interface within the modiCAS system designed for efficient host-target communication. The CSM transmits commands to the target computer.
CT	Computed Tomography is a medical imaging method employing tomography. This procedure uses x-ray cross-sectional images of the body to generate a three-dimensional image.
CVEL	Cartesian velocity controller: is a control strategy that accepts velocity commands in the Cartesian space to drive the robot arm. It is implemented on the target computer and can be activated by the host computer with the CVEL command.
CVT	Continuously Variable Transmission: Special purpose variable transmission applied on passive cooperative robots called COBOTS.

DLS	Damped Least Squares: a singularity robust method used to prevents the robot joint velocities from becoming excessively high near singular configurations by using a damping factor to control the norm of the joint velocity vector.
DOF	Degrees of Freedom: the set of independent displacements and/or rotations that specify completely the displaced position and orientation of a body or system.
DRM	Data Receiver Module: One of four modules used in the command interface within the modiCAS system designed for efficient host-target communication. The DRM receives incoming data at the host computer.
DTM	Data Transmitter Module: One of four modules used in the command interface within the modiCAS system designed for efficient host-target communication. The DTM sends data from target computer to the host computer.
FT Sensor	Force Torque Sensor mounted at the end-effector of the robot arm.
FVK	Forward Velocity Kinematics: Transformation of joint velocities into linear and angular velocities in the Cartesian space.
GUI	Graphical User Interface: user interface that allows the user to interact with a computer. It provides graphical icons, visual indicators or special graphical elements called widgets.
GVF	Guidance Virtual Fixture: is a control strategy to drive the robot's end-effector directly with the hands, where virtual constraints are applied to delimit the working space. It is implemented on the target computer and can be activated by the host computer with the GVF command.
HPL-FTS	High Priority Loop running at the target computer for data acquisition of the Force Torque Sensor.
HPL-NAV	High Priority Loop running at the target computer for the interaction with the Navigation system.
JVEL	Joint Velocity controller: is a control strategy that accepts velocity commands in the joint space to drive the robot arm. It is implemented on the target computer and can be activated by the host computer with the JVEL command.
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench: is a development environment for graphical programming language developed by National Instruments.
LIN	Linear Trajectory controller: is a control strategy to follows linear

	trajectories in the Cartesian space. It is implemented on the target computer and can be activated by the host computer with the LIN command.
MDL	Meta Data List: A list of commands, each of which is associated with an identifier (ID). The list is used within the communication protocol in order to increase efficiency during data transmission between computers.
modiCAS	modular interactive Computer Assisted Surgery: A modular concept developed at the University of Siegen to provide an integral solution for different surgical problems by the combination of a navigation system and a robot arm with hands-on capabilities.
MIS	Minimal Invasive Surgery: a surgical technique that makes small incisions on the body through which special instrumentation is used to accomplish the operation. This technique significantly reduce trauma to the body compared to traditional incisions.
MRI	Magnetic Resonance Imaging used in medical imaging to visualize the structure and function of the body. It provides detailed images of the body in any plane.
OR	Operating Room
PA10-6C	Portable General Purpose Intelligent Arm with 6 degrees of freedom manufactured by Mitsubishi Heavy Industries, LDT.
PTP	Point to Point controller: is a control strategy to reach a target position in the joint space following a synchronized velocity trapezoid profile for each joint. It is implemented on the target computer and can be activated by the host computer with the PTP command.
QNX	A POSIX compliant UNIX like real time operating system.
Qt	A cross-platform application development framework used for the development of graphical user interfaces.
RB	Rigid Body: is a reference body equipped with marks that can be detected by a navigation system.
RT	Real-Time: is a level of computer responsiveness that the user senses as sufficiently immediate. High determinism is a characteristic of real-time systems and guarantees that the calculations and operations occur in time and on time.

TCL-ROB	Time Critical Loop running at the target computer containing the main controller to drive the robot arm.
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol/Internet Protocol: set of communications protocols that implements the protocol stack on which the Internet and most commercial networks run.
TSK-DONE	Task Done: Acknowledgment transmitter from target computer to host computer each time a requested task is completed.
XML	Extensible Markup Language: general purpose markup language that allows its users to define their own tags. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet.
SC-Inverse	Singularity Consistent method solves the problem of singularities based on the null space.
VF	Virtual Fixtures are software-generated force and position signals applied to human operators via robotic devices. These help humans perform robot-assisted manipulation tasks by limiting movement into restricted regions and/or influencing movement along desired paths.
w.r.t.	With respect to
ZESS	Zentrum für Sensorsysteme is the german name of the interdisciplinary research Center for Sensor Systems at the University of Siegen.

NOMENCLATURES

General

$\boldsymbol{\theta}$	Joint positions vector
$\dot{\boldsymbol{\theta}}$	Joint velocities vector
${}^{B}\boldsymbol{T}_{A}$	(4×4) homogeneous Transformation matrix from frame {A}to frame {B}
${}^{B}\boldsymbol{R}_{A}$	(3×3) Rotation matrix from frame {A}to frame {B}
${}^{B}\mathbf{p}_{A}$	Position vector of A with respect to frame B
$\mathbf{p}(s)$	Parametric function
$\dot{\mathbf{p}}$	Cartesian linear velocity vector
$\boldsymbol{\omega}$	Cartesian angular velocity vector
$\dot{\mathbf{x}}$	Cartesian linear and angular velocity vector of the form $[\dot{\mathbf{p}}^{T} \quad \boldsymbol{\omega}^{T}]^{T}$
$\mathbf{f}$	Forces vector
$\boldsymbol{\tau}$	Moments vector
$\boldsymbol{\gamma}$	Forces and moments vector of the form $[\mathbf{f}^{T} \quad \boldsymbol{\tau}^{T}]^{T}$
$\boldsymbol{J}$	Jacobian matrix
$\boldsymbol{J}^{\#}$	Pseudoinverse of Jacobian matrix

Virtual Fixtures and Admittance Controller

U	Subspace of preferred directions along which the robot's end-effector is allowed to move
V	Subspace of non-preferred directions along which the robot's end-effector cannot move
$\boldsymbol{S}^{l}$	Subset of $\Re^{3}$ comprising the linear independent set of vectors that span U for position
$\boldsymbol{S}^{\alpha}$	Subset of $\Re^{3}$ comprising the linear independent set of vectors that span U for orientation
P_{U}	Projection operator onto subspace U of preferred directions
P_{V}	Projection operator onto subspace V of non-preferred directions
c	Compliance coefficient for admittance controller

$\boldsymbol{C}$	Diagonal compliance matrix for admittance controller
$\mathbf{e}$	Deviation error vector
ϕ	Quaternion

Damped Least Squares approach

$\boldsymbol{J}^*$	Singularity-Robust (SR) inverse of the Jacobian matrix
w	Manipulability measure of Jacobian matrix
λ	Damping factor
$\boldsymbol{\Sigma}$	Diagonal matrix with singular values of a matrix resulting from singular value decomposition

Adjoint Jacobian approach

$adj\boldsymbol{J}$	Adjoint of Jacobian matrix
$\det\boldsymbol{J}$	Determinant of Jacobian matrix
f_i	Factorized det$\boldsymbol{J}$ corresponding to one singularity of the robot, $i = (1,..,k)$ with k equal maximum number of possible singularities
v	End-effector velocity magnitude
$\mathbf{u}$	End-effector motion direction unit vector
b	Scalar variable of adjoint Jacobian expression
σ	Sign variable of adjoint Jacobian expression
$\boldsymbol{H}$	Column augmented Jacobian
$\mathbf{n}_H$	Vector existing in the null space of matrix $\boldsymbol{H}$

ABSTRACT

Im Rahmen des *modular interactive computer assisted surgery* Projekts (modiCAS) wurde eine Lösung zur Kombination eines Navigationssystems und eines manuell steuerbaren Roboterarms zur Unterstützung verschiedenster chirurgischer Eingriffe entwickelt [70]. Dieses Robotersystem kann als intelligentes Werkzeug betrachtet werden, das die Fähigkeiten eines Chirurgen auf assistierende und kooperative Weise ergänzt. Das System soll den Chirurgen während Operationen unterstützen, keinesfalls jedoch ersetzen. Das Ziel dieser Arbeit liegt darin, die Interaktion zwischen Chirurg und Roboter bei manueller Führung zu erweitern und einfach und sicher zu gestalten.

Der erste Teil dieser Arbeit beschreibt die Architektur eines *software-frameworks*, das die Bedienung des modiCAS-Systems in verschiedenen chirurgischen Bereichen zulässt. Hierbei ermöglicht eine modulare Struktur, in Verbindung mit einer strategischen Verteilung der einzelnen Module, ausreichende Flexibilität um eine Hardware Plattform an verschiedene Anwendungen anzupassen.

Im zweiten Teil wird, im Hinblick auf die kooperativen Fähigkeiten des modiCAS Systems, das Vermeiden von Fehlsteuerungen mittels virtueller Beschränkungen (*virtual fixtures*) durch den *Admittance Controller* behandelt. Diese dienen dazu die Bewegungsmöglichkeiten des Roboterarms bei haptischer Führung so zu begrenzen, dass er sich nur in vordefinierte und erlaubte Richtungen bewegen kann. Die Kombination von gewünschten Richtungen erlaubt die Konstruktion verschiedener virtueller Einschränkungen, die eindimensional (z.B. Linie), zweidimensional (z.B. Oberfläche) oder dreidimensional (z.B. Kegel oder Zylinder) sein können. Außerdem sind komplexe Kurven generierbar, die von dem Roboterarm bei manueller Führung exakt durchlaufen werden.

Die Methoden *Damped Least Squares* und *Adjoint Jacobian* vermeiden hohe Geschwindigkeiten, die während der haptischen Führung des Roboterarms bei singulären Konfigurationen auftauchen können. Eben dies darf während einer kooperativen Operation mittels virtueller Begrenzungen nicht geschehen.

Die in dieser Arbeit vorgestellten Methoden liefern einen Beitrag für eine sichere und präzise Steuerung von manuell gesteuerten Roboterarmen. Sie erhöhen die Funktionalitäten zur Assistenz und das Integrationsniveau von Roboterarmen bei

chirurgischen Einsätzen. Auf diese Weise wird die Interaktion zwischen Chirurg und Roboter intuitiver und leichter in der Handhabung. Die Möglichkeit zur Festlegung virtueller Begrenzungen verbessert die Sicherheit in der Anwendung erheblich, da der Chirurg den Roboterarm nicht unbeabsichtigt in leicht verletzbare Bereiche führen kann.

1. Introduction

The modular interactive computer-assisted surgery (modiCAS) project, settled in the Center for Sensor System (ZESS) at the University of Siegen, in Germany, is engaged to develop an integral solution for different surgical problems by the combination of a navigation system and a robot arm with hands-on capabilities [70]. The robotic system may be thought of as a smart surgical tool that extends surgeon's ability to treat patients, giving him/her surgical assistant by working in cooperative fashion. However, a natural and seamless integration of robotic systems in the operating room is still a big challenge in robotic surgery. The interaction between surgeon and robotic system is a very important issue. Autonomous systems have lost acceptance in the surgical community because the surgeon wants to be in charge of the operation rather than acting only as an observer. In such autonomous procedures, human experience, intuition, reacting capability in front of unexpected situations are lost. Furthermore, assistance intends to improve the performance of the surgeon rather than delimitating or obstructing him/her. An alternative solution is to provide a cooperative system where benefits of both can be combined. In this context, the surgeon gain complete control over the operation by grabbing the tool mounted on the robot and commanding it with his/her own hands. But the fact that a robot is to be used in clinical applications and in direct contact with human beings, imposes some additional requirements in comparison with the well established robotics technology applied in the industry. The most obvious is safety. On the one hand, the surgeon must keep control of the surgical operation. On the other hand, the surgical robots must assure a correctly usage by the surgeon in order to guarantee patient safeness. Therefore, surgeon's freedom of action has to be partially limited so that forbidden regions become unattainable to prevent accidental injuries. For these reasons, a seamless and safety integration of the system within the operating room is considered a paramount issue for a successful assistance and represents an important requirement within the modiCAS project.

The first contribution of this work is a proposal of software framework architecture for the modiCAS system able to support medical interventions in several surgical disciplines. A modular structure, together with a strategic distribution of the modules, provides flexibility for the adaptation of a common basic hardware platform to different applications.

Secondly, concerning the cooperative capabilities of the modiCAS system, the issue of mishandling is avoided with the introduction of virtual constraints, here called virtual fixtures, which help guiding the tool within certain predefined permitted directions. The combination of these preferred directions permits different kinds of virtual constraints, which may be one dimensional (lines), two dimensional (planes), three dimensions (tubes, cones, etc.) or even more complex trajectories (by means of parametric functions).

A particularly issue to be considered when thinking about cooperative manipulation of a robotic arm is the presence of singular configurations. In the neighborhood and at singular configurations an exact solution of the robot inverse kinematic becomes ill conditioned. Consequently, unfeasible joint velocities may be produced which yields into acute behavior of the robotic system. During virtual constrained cooperative operation, high velocities and position deviations are unacceptable. Therefore, another important objective of this work is to assure that such cooperative guidance is robust and accurate even in the presence of such singular robot configurations.

In chapter 2, the modiCAS system is introduced, together with the system demands that motivate the contribution of this work. Chapter 3 presents the state of the art in computer assisted surgery systems, virtual constraints and singularity robustness. The proposed modular software framework is detailed explained in chapter 4. Chapter 5 introduces the concept of virtual fixtures together with the corresponding admittance controller used to apply them to the robotic system. Two types of controllers are compared; each of them differing from the other in the way the deviation error is handled: manual error compensation and the proposed autonomous error compensation. The former relies on the input forces applied by the user to compensate possible deviation error, while the latter delegates this job to the robotic system. The technical challenge of dealing with singularities is treated in chapter 6. Two approaches are analyzed and compared: the Damped-Least-Squares and the adjoint Jacobian approaches. Finally, chapter 7 presents the conclusions.

2. Motivation to design a cooperative robotic assistant surgery system

2.1 The modiCAS project

The basic concept of the modiCAS project is to integrate navigation system and robotic arm into one system that appears a single unit, combining the specific advantages of each other. A hands-on interface mounted at the robot end-effector provides highly interactive operation. The system setup consists of an optical 3D "Polaris" digitizing system (from NDI Inc., Canada), the PA10-6C robot arm (from Mitsubishi Heavy Industries, Ltd., Japan), a light weight (35 kg) robotic arm with 6 degrees-of-freedom (DOF), and a mini45 force-torque sensor (from ATI Industrial Automation, USA), which is integrated in the surgical tool mechanism. Figure 2.1 shows the different components of the navigated robotic system.

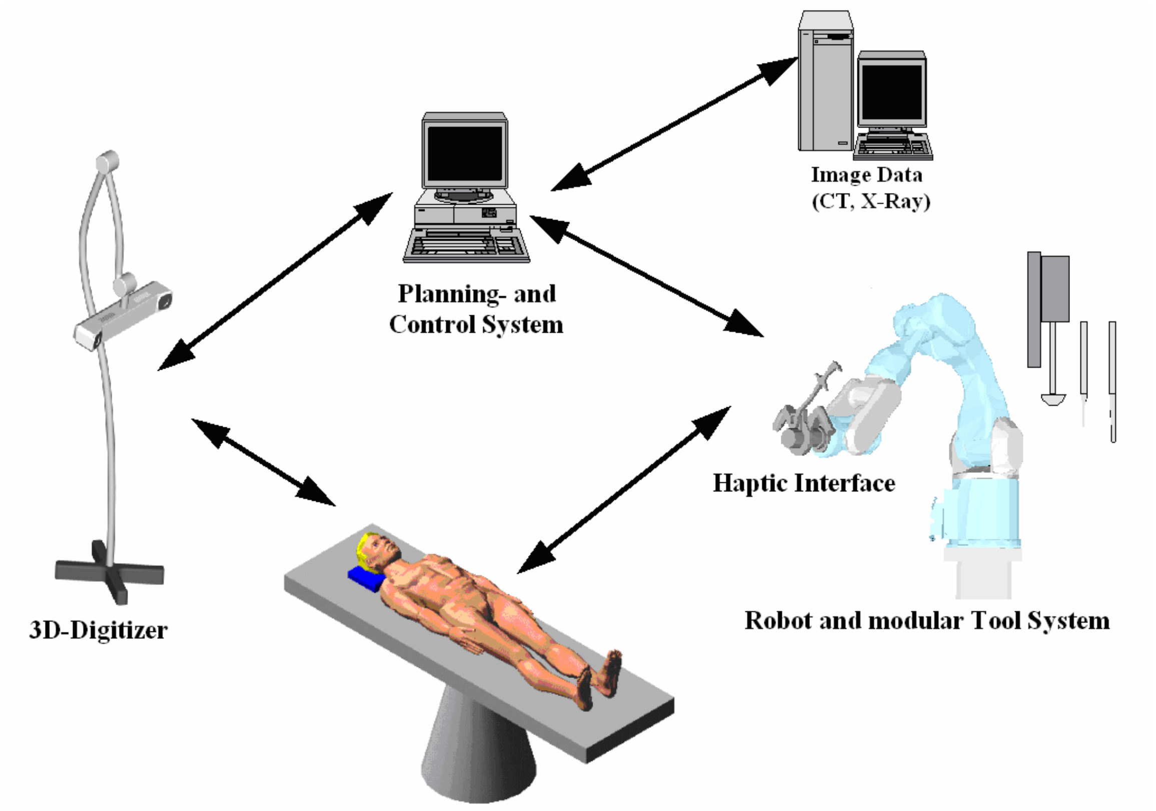


Figure 2.1. Components of the navigated robotic system for surgical assistance

The control system consists of two computers: On the one hand, an embedded target computer, which runs a real-time operating system, where all the fundamental functionalities are implemented, and on the other hand, a host computer, with Microsoft Windows operating system, where the graphical-user-interface (GUI) and the application-oriented tasks are running. Both computers communicate through a dedicated Ethernet connection.

The modiCAS software framework is divided in two main parts: The planning-software and the controller-software. The former is used for preoperative planning, registration and intra-operative visualization. This software provides all important planning functionalities through a virtual toolbox and supports the common standards for the different images modalities (X-Ray, CT, and MRI). The controller software is in charge of the management and usage of the physical components of the system. Although, these two parts work together to provide a complete solution, the development of this work concerns only the controller-software. The reader interested in further details related to the planning software is encouraged to consult [43].

2.1.1 Combination of navigation and robotics

The basic concept of modiCAS system is to integrate a navigation system and robotic arm into one system that appears as a single unit, combining the specific advantages of navigation and surgical robotics [70]. Patient registration is performed by using only the navigation system, while the robot arm positions the surgical instrument during intervention. Thus no unintentional deviations caused for example by tremor or slipping can occur. Furthermore, the surgeon does not have to permanently change his eyes from the operating area to the computer screen where he/she has to monitor the instrument position, and he /she can fully concentrate on the operating area.

The tool adapter is equipped with a rigid body (RB), which can be detected by the navigation system. During system initialization, a setup procedure is carried out to align the coordinate systems of the robot arm and the navigation system. As a result, all movements can be specified and executed with reference to the coordinate system of the navigation system. This also provides redundant measurement of the surgical tool position by two completely independent systems, (a) the navigation system detecting the

RB element, and (b) the built-in encoders of the robot joints. This is an important feature to meet the high safety requirements applicable to surgical robotics.

A special feature of the robotic arm is its ability to automatically track potential movements of the patient in real-time, eliminating the need for rigid fixation of the anatomic structure to be operated. Instead, a RB is attached to the patient operable structure by a suitable fixation mechanism. After registration of the patient anatomy, the operating area is well known by the system and can be tracked by the robot. If any patient movement is detected during surgical intervention, the controller generates corresponding motion commands that move the robot arm to follow the patient, keeping the surgical instrument always in the pre-planned position and orientation with respect to the patient anatomy.

Although the patient tracking capability of the system is of great relevance and represents a key feature of the modiCAS project, a deep insight in this topic is actually out of the scope of this work, which mostly concentrates on the controller framework design and the cooperative capabilities of the system. More detailed information related to the tracking mode can be found in [70].

2.1.2 Human-robot interaction

One important goal of the system is transparency i.e., the ability to move the tool freely and dexterously. Therefore, the robot arm is equipped with a hands-on interface consisting of a 6 DOF force-torque-sensor (FTS) mounted at the robot's end-effector. External applied forces to the tool can be detected by the system. The surgeon can thus freely move the end-effector through the desired operating region just by grabbing the handle mounted on the end-effector and guiding the arm towards the target area. This hands-on capability integrates the robot seamlessly in the operating procedure, because there is no need to use any input-device like mouse, touch screen or keyboard to command the robot. An additional feature of the hands-on interface is that any force/torque externally applied to the surgical tool can be monitored. Such information can be used to enhance safety during surgical procedures. At the beginning of this work, the hands-on interface was already available within the system, but non mechanism was available to constrain the working area.

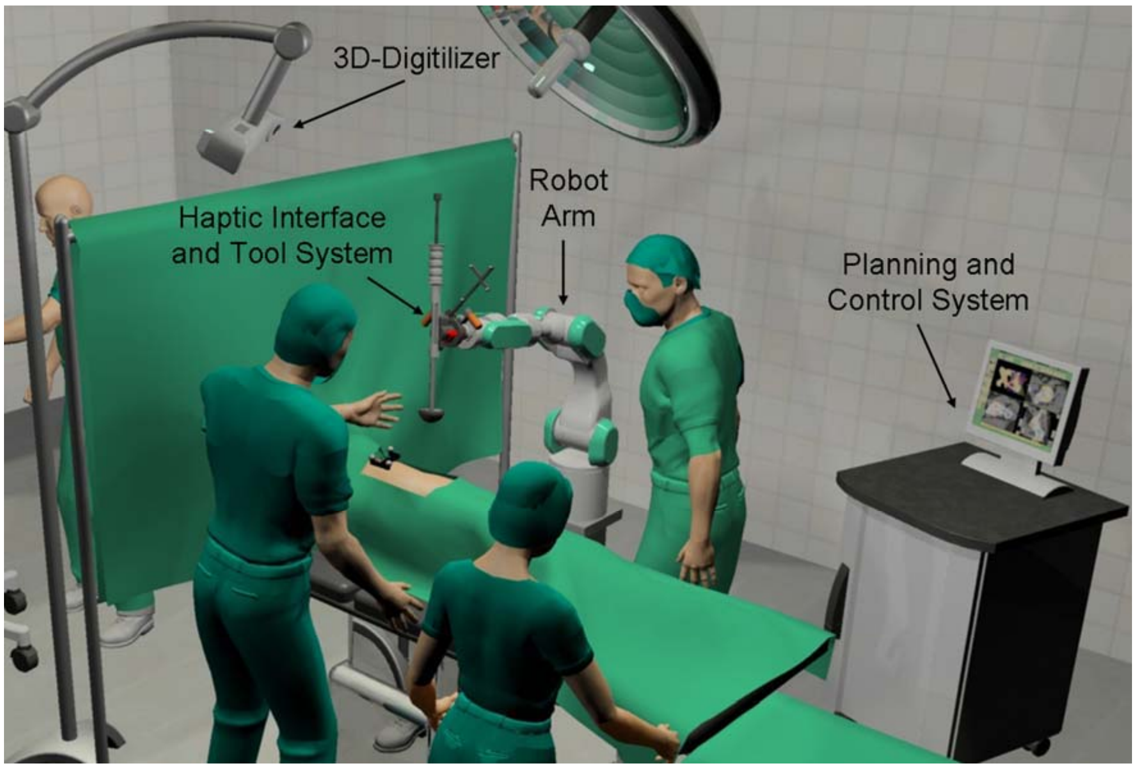


Figure 2.2. modiCAS system in the operating room

In other words, the surgeon was able to move the robot anywhere in the robot's working area, which, for safety reasons, may not be always desired. Another limitation was the fact that the cooperative mode was not singularity robust, i.e., passing through singular configurations of the robot was not possible. Therefore, the hands-on interface could only be used to coarsely positioning the tool by hand in the working area. After that, the system was switched to automatic control and fine positioning was carried out under computer control according to preoperative planning. Switching back to cooperative mode is always possible whenever the surgeon wishes so. Figure 2.2 illustrates the incorporation of the modiCAS system in the operating room (OR).

2.2 System demands

The objective of the modiCAS project is to consolidate a flexible robotic system that provides both navigated assistance and cooperative capabilities to support different Computer Assisted Surgery (CAS) disciplines. The robotic system should provide assistance to the surgeon rather than substituting him/her. Assistance intends to improve the performance of the surgeon instead of delimitate or obstruct him/her. In other words, the surgeon must keep control of the surgical operation all the time while the robotic system simply becomes a tool at his/her disposition, which usage should be as intuitive as possible. Nevertheless, the absence of human mistakes cannot be completely assured. Therefore, surgeon freedom must be limited in a way that forbidden regions become unattainable so that accidental injuries can be prevented. For these reasons, a seamless and secure integration of the system within the OR is considered a paramount issue for a successful assistance and represents an important requirement within the modiCAS project.

System flexibility and safe human-robot interaction are then the two major aspects treated in this work. On the one hand, a new controller software architecture specially designed to improve flexibility of the modiCAS system has to be developed and implemented. On the other hand, a virtual constraining mechanism has to be developed to assure a safe human-robot interaction during cooperative tasks.

2.2.1 Controller software architecture

The spectrum of possible applications of a navigated cooperative robotic assistant system is very broad [28]. However, each application presents particular problems, which demands particular solutions, i.e. particular expectations about how the robotic assistant system must behave. Therefore, the system architecture must be flexible enough to adapt itself to the demands of various surgical scenarios without requiring exhaustive changes in the internal structure. The modiCAS controller software bases its design on the following concept of modularity to cope this requirement:

Modularity – A clear modularization of the different tasks as well as a strategic distribution of them along the system framework, depending of their roll within the system, are key issues to achieve enough system flexibility to cope various applications. But modularity is not only restricted toward applications. Additionally, a modular hardware-interface expands this flexibility towards the system itself. Some resulting advantages are maintainability and scalability. In this way, thinking about upgrading, replacing or even adding a new component must not affect the integrity of the system.

2.2.2 Human-robot interaction

At the present state of the modiCAS system, the hands-on interface can be used to move the tool about the working space with the surgeon's hands. Returning the tool back to exact operating position has to be done autonomously by the robot. The surgeon has freedom to move the robot end-effector at any time in any direction. Nevertheless, there exists no mechanism to assure that the cooperative motion keeps the tool inside a predefined region or leads it towards a specific desired point. Under such circumstances, no cooperative task can be applied, since no constraints exist that avoid the surgeon to conduct the tool toward forbidden regions where injuries to the patient can occur.

The motivation of the second part of this work is the development of a cooperative modality though which the surgeon is able to freely guide the robot's end-effector with his/her own hands inside some predefined constrained area, but assuring that any movement outward this region becomes unfeasible. Therefore, the concept of virtual fixtures is applied to the hands-on interface. Namely, any movement commanded by the surgeon is virtually constrained along permitted directions. The constrained space can be along a curve, a surface or even inside volumetric shapes. These virtual fixtures are previously defined in the preoperative stage of the surgical intervention. The navigation system makes possible to define such constraints in direct relation to the patient. Besides, the system compliance against surgeons applied forces can vary depending of the proximity to the patient.

In this context, applying virtual fixtures to the simple task of safely moving the tool back and fort of the working area would look as follows: when trying to push the tool out of the working space, first a simple linear movement on the negative direction normal to the operating plane is applied in order to get out of the critical area nearby the patient in a safety way. After certain distance, the virtual constraint is shifted to an inverted conic form giving the possibility to locate the robot out of the way not to obstruct any other activity of the surgeon. On the same way, once the robot is pulled back to the working area, the virtual constraints procure that the final operating position and orientation are safely achieved. In this case no autonomous movement of the robot is required anymore. The virtual-fixtures provide a safety measurement that allows active participation of the robotic system in cooperative tasks during surgical procedures.

2.3 Technical challenges

2.3.1 Controller redesign

The actual stand of the project is the fruit of a combined effort of all member of the modiCAS team along the past years [43], [70], [106], [136]. A lot of experience has been collected through each contribution. Parallel development concerning planning-software, controller-software, navigated patient tracking strategy and hands-on interface did great advance during the first four years. A first prototype was developed and successfully implemented. Even clinical trials were successfully achieved [136].

The controller software structure of this first prototype is shortly explained: The modiCAS software is distributed on two computers. The first one, running a Windows operating system, contains the GUI and all not deterministic tasks. The second computer runs a real-time operating system and comprises all deterministic tasks, such as control loops, data acquisition, and hardware interface. Originally, QNX operating system was the platform used for the real-time computer. The software implementation was based on C++ object oriented programming language. CORBA infrastructure was used for computers interaction, which were connected over a local dedicated network. The GUI of the controller software was developed using Qt framework.

At some point of the development, the modiCAS team realizes that the software architecture based on the original design started to become very complex. Any adaptations to cope new applications required a deep knowledge of the whole framework and therefore implied a considerable extra effort to do small changes. Besides, this turned out to be overwhelming for each new member of the team and considerably slowed the development process. Consequently, a strategic decision came out, namely, the redesign of the control software under a new real-time platform, with a new communication mechanism and mostly important with the main requirement of modularity.

The LabVIEW high level environment based on graphic programming language was chosen for the development of the new controller software. Thanks to the LabVIEW Real Time (RT) module [77], it is possible to develop real-time applications in a conventional desktop personal computer (PC) running the Venturcom Phar Lap

Embedded Tool Suit (ETS), a high-performance micro-kernel real-time operating system [9].

Considering the fact that the modiCAS project has a tight relationship with the University of Siegen, where student collaboration is a common and practicable case, LabVIEW offers the following advantages:

- Easy understandable programming language
- Reduced learning curve
- Cut down considerably implementation time
- C-code can be very easily imported
- Professional technical support
- Desktop PC compatible

Under such circumstances, the modiCAS controller-software redesign implicates the aggregated challenge of achieving a successfully implementation in a completely new platform.

2.3.2 Robot singularities

Robot singularities are special configurations of the robot where its behavior becomes ill conditioned. Near singular configurations some robot joints may present very high velocities yielding into acute behavior. In a cooperative system, where human-machine interaction is highly coupled, such behavior is unacceptable. Although the singularity problem is well known in the field of robotics, commercial industrial robotic system do not jet offer a build-in convincing solution. This situation becomes even worse in the field of cooperative robotics, a field that until now has not been firmly settled for commercial purposes. This means that an alternative strategy must be developed that allows safely cooperative tasks nearby or at singularity configurations.

Two scenarios are supposed: the first case comprises unconstrained cooperative motion of the end-effector. In such a case, slight position deviation when passing through singular configuration are tolerated. High priority is given to the smoothness of the motion rather than accuracy of end-effector's position.

The second case is exactly the opposite i.e., the end-effector motion is virtual constrained. Here, position deviations could mean that the end-effector goes out of the permitted area. Therefore, accuracy has highest priority when passing nearby or through singularities.

3. State of the art

3.1 Surgical robotic systems

The robots can be seen as a mechanism that have complementary capabilities to those of humans, and may be used in a number of ways to augment a surgeon's ability to carry out procedures, either by making existing procedures more accurate, faster, or less invasive or by making it possible to perform otherwise infeasible interventions. In these cases, the advantages often come from exploiting the complementary strengths of human and robotic device; Table 3.1 summarizes strengths and limitations of each of both, humans and robots [125]:

Table 3.1. Complementary capabilities of human and surgical robots

	Strengths	**Limitations**
Humans	Excellent judgment Excellent hand-eye coordination Excellent dexterity at natural human scale Able to integrate and act on multiple information sources Easily trained Versatile and able to improvise	Prone to fatigue and inattention Tremor limits fine motion Limited manipulation ability and dexterity outside natural human scale Cannot see through tissue Bulk end-effector (hands) Limited geometric accuracy Hard to keep sterile Affected by radiation infection
Robots	Excellent geometric accuracy Untiring and stable Immune to ionizing radiation Can be designed to operate at many different scales of motion and payload Able to integrate multiple sources of numerical & sensor data	Poor judgment Hard to adapt to new situations Limited dexterity Limited hand-eye coordination Limited haptic sensing (today) Limited ability to integrate and interpret complex information.

Taylor classified the systems by the role they play in medical applications [122]. He stresses the role of robots as tools that can work cooperatively with surgeons to carry out surgical interventions and identifies five classes of systems:

1. Intern replacements – The system performs assistive tasks that are ancillary to the main surgical procedure and that are frequently performed by surgical interns and other people whose main job is to help the surgeon.
2. Telesurgical systems – The robot's motions are specified directly by the surgeon by means of a joystick, control handle, or similar device. The surgeon used the robot as an extension of his own direct manipulation capabilities. Such systems give the surgeon access to difficult to reach parts of the body or the ability to perform delicate microsurgical tasks without tremor.
3. Navigational aids – The goal is simply to provide the surgeon with accurate positional feedback about the location of surgical instruments relative to the patient's anatomy. These systems are often referred to as CAS, and typically consist of a 3D localizing device such as an instrumented passive manipulator, ultrasound detector, or 3D optical tracker, together with a computer graphics workstation for displaying position relative to volumetric medical images.
4. Precise positioning systems – The robot is used to position a tool guide in the desired position and orientation relative to the target anatomy. For safety reasons, the robot is often turned off during the actual instrument insertion. Although this reduces the chance of unwanted motion at critical times, it does not address the potentially more crucial issue of misregistration.
5. Precise path systems – The robot is moved through a defined path to complete a specific task. For example, a precise machining of bone either using the robot to move the cutting tool or as a means of constraining the surgeon to keep the tool within a predefined volume.

Other authors divide the field by clinical applications [28], [35], [124]. A list of seven clinical areas where robotics has been applied is shown in Table 3.2. [28].

Table 3.2. Clinical application areas and representative robotic developments

Clinical Area	Country	Institution/ Company	System	Reference
Neurosurgery	Switzerland	Univ. of Lausanne	Minerva	[16]
Neurosurgery	USA	ISS / Grenoble Univ. Hospital	NeuroMate	[79]
Neurosurgery	Japan	Univ. of Tokyo	MRI compatible	[91]
Orthopaedic	USA	ISS	ROBODOC	[120]
Orthopaedic	USA	Georgetown / Hopkins	PAKY/RCM	[27]
Orthopaedic	USA	Univ. of Tokyo / Hopkins	PAKY/RCM	[28]
Orthopaedic	USA	Marconi	Kawasaki	[28]
Orthopaedic	UK	Imperial College	Acrobot	[57]
Urology	UK	Imperial College	Probot	[129]
Urology	USA	Hopkins	PAKY/RCM	[117]
Maxillofacial	Germany	Charite	Surgiscope	[83]
Maxillofacial	Germany	Karlsruhe / Heidelberg	RX 90	[57]
Radiosurgery	USA	Accuray	CyberKnife	[6]
Opthamology	USA	Hopkins	Steady Hand	[123]
Cardiac	USA	ISS	da Vinci	[45]
Cardiac	USA	Computer Motion	Zeus	[12]
Cardiac	France	Grenoble	PADyC	[113]

Since various systems in development pretend to cover various disciplines and applications, the classification criterion on this work is rather based on their degree of autonomy i.e. the type and level of interaction between robotic system and the surgeon, distinguishing between three main categories: Autonomous systems, Cooperative Systems and Teleoperative systems.

3.1.1 Autonomous system

An autonomous robotic surgery is the process whereby a robot actually carries out a surgical procedure under the control of nothing other than its computer program. Although surgeons almost certainly will be involved in the planning of the procedure to be performed and will also observe the implementation of that plan, the execution of the plan will not be accomplished by them, but by the robot. The surgeon has always the

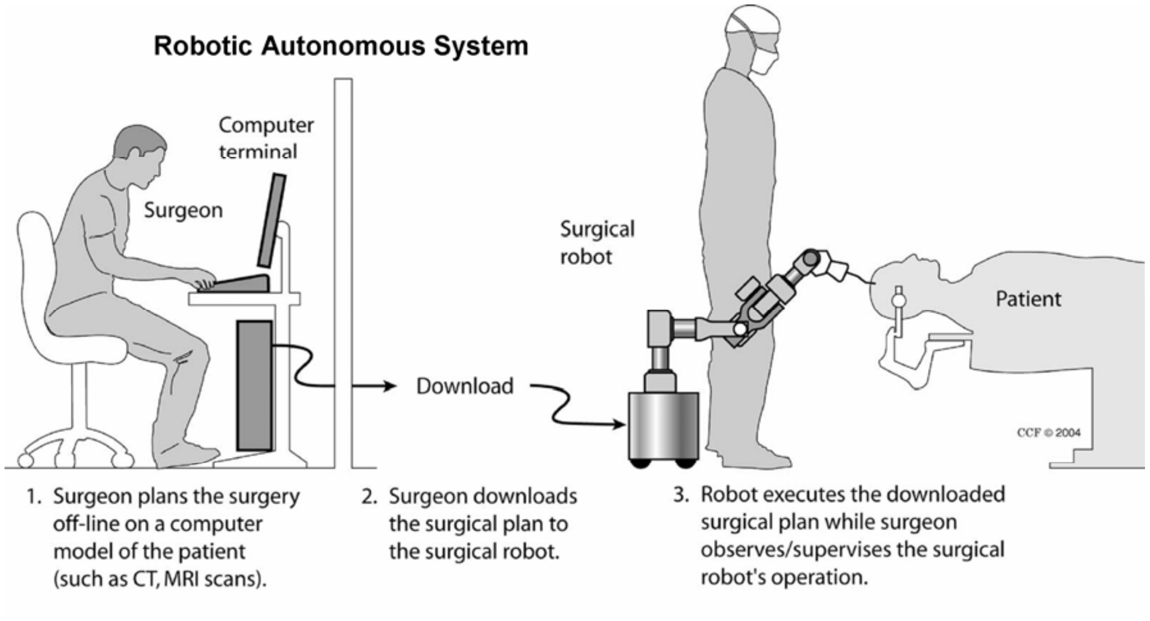
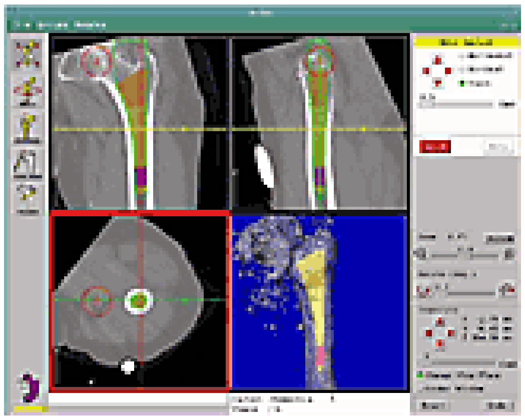

Robotic Autonomous System
Computer terminal
Surgeon
Surgical robot
Patient
Download
CCF © 2004
1. Surgeon plans the surgery off-line on a computer model of the patient (such as CT, MRI scans).
2. Surgeon downloads the surgical plan to the surgical robot.
3. Robot executes the downloaded surgical plan while surgeon observes/supervises the surgical robot's operation.

Figure 3.1. Autonomous System: The robot executes the procedure while surgeon observes/supervises the operation [94]

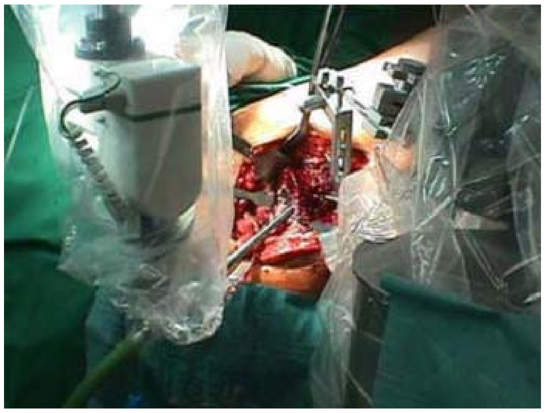
possibility to stop the robotic system and continue manually the interrupted procedure (see Figure 3.1).

The first autonomous surgical systems were designed for orthopaedic surgery. In the U.S., Taylor and associates at IBM began developing the system later known as ROBODOC [120]. This system was further developed clinically by Integrated Surgical Systems (ISS) for total hip replacement procedures. The system consists of three mayor components: a planning workstation (ORTHODOC), the robot itself that does the cutting, and the workstation that guides and controls the robot. Since this system has a number of features also found in other surgical systems, a typical procedure using the system is described:

The surgeon selects an implant model and size based on an analysis of preoperative CT images and interactively specifies the desired position of each component relative to CT coordinates. In the operating room, the robot is moved up to the operating table, the patient’s bones are attached rigidly to the robot’s base through a fixation device, and the registration of the patient with the robot is done either by touching multiple points on the surface of the patient’s bone or by touching pre-implanted fiducial markers whose CT coordinates have been determined by image processing.



(a)



(b)

Figure 3.2. (a)ORTHODOC planning workstation, (b) ROBODOC milling implant cavity for hip replacement surgery (courtesy of ISS, USA)

The surgeon hand guides the robot to an approximate initial position using a force sensor mounted at the robot's end-effector. The robot then cuts the desired shape while monitoring cutting forces, bone motion, and other safety sensors. The surgeon monitors this process by watching a computer screen which shows the progress of the cutting operation. The robot can also be stopped at any time. When the desired shape has been cut, the robot is removed and the rest of the operation is completed by hand in the conventional manner. A picture of the ORTHODOC planning software and ROBODOC milling the cavity for the implant is shown in Figure 3.2.

A number of other robotic systems for use in joint replacement surgery were subsequently proposed, such as the CASPAR system (from ortoMAQUET, Germany) shown in Figure 3.3. The system is based on the industrial robot Stäubli [39], which was very similar to ROBODOC. The system has been used for implantation of hip prosthesis in total-hip-replacement (THR), as well as for anterior cruciate ligament reconstruction [105].

Although these systems successfully achieve the goal of improved fit, there are a number of common difficulties [52]. One very important issue is the complex method for fixing the operating bone structure, which is time consuming to set up and can cause postoperative pain. A related problem is motion of the bone within the fixation device during cutting. Currently, a separate sensing system is required to check for motion; if bone shift is detected, cutting is interrupted and the registration process must be repeated. Several incidents of femur motion can push the surgical time over the limit of acceptability. An improved fixation technique or continuous registration method could eliminate these problems.

Another completely different type of autonomous robotic systems is used for radiosurgery. Stereotactic radiosurgery is a medical procedure that utilizes very accurately targeted, large killing doses of radiation, which has proven to be an effective alternative to surgery or conventional radiation for treating many small tumours and a few other selected medical disorders. Standard stereotactic techniques rely on a rigid metal frame fixed to a patient's skull for head immobilization and target localization. Adler and associates at Stanford University (U.S.) in conjunction with Accuray Inc., U.S., developed the CyberKnife for image-guided radiosurgery [6].

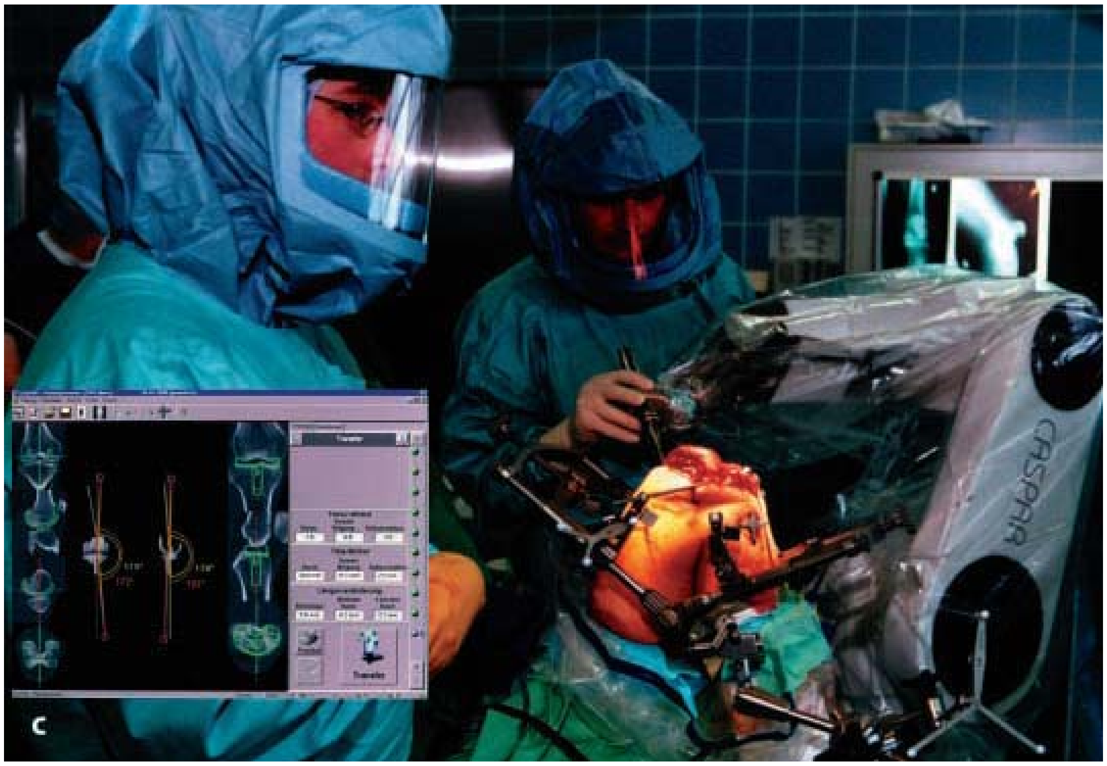


Figure 3.3. CASPAR system in knee operation (courtesy of ortoMaquet)

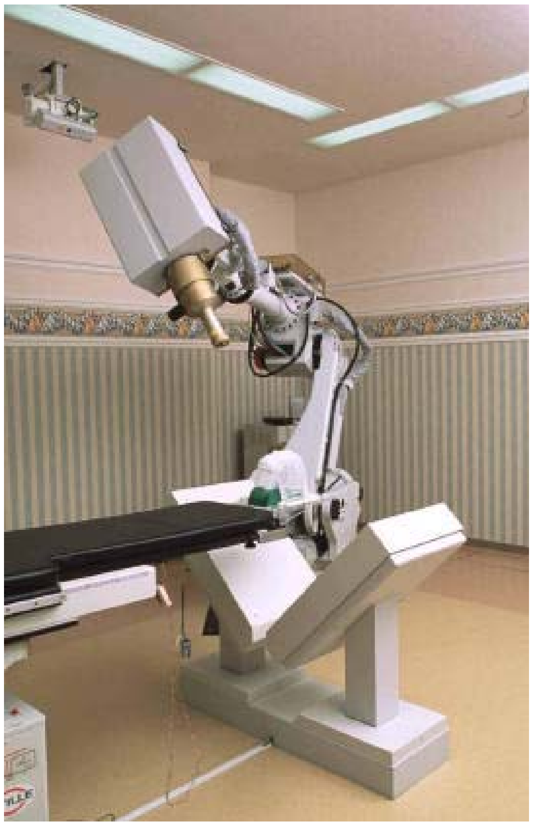


Figure 3.4. CyberKnife robotic radiosurgery system (courtesy of Accuray, USA)

The system consists of a linear accelerator (used to produce a high energy killing beam of radiation), a robot which can point the linear accelerator from a wide variety of angles, and several x-ray cameras to track the patient position. Lightweight.

The robot arm moves the beam through a series of preset positions to maximize the dose to the lesion while minimizing the dose to the surrounding normal tissue. The CyberKnife is shown in Figure 3.4. In contrast with the systems presented until this point, non direct contact is performed with the patient. Nevertheless, the application is not considered to imply less risk than the others. If the goal position is not precisely reached, healthy regions can be damaged.

3.1.2 Cooperative systems

Robot systems operating in collaboration with humans has been an active topic of research during the last two decades. Various control systems have been proposed by Kazerooni *et al.* ([62], [63]) to generate the motion based on the intentional force. Cooperative tasks for industrial applications such as cooperative manipulation ([72], [73]), peg-in-hole tasks [132], has been proposed.

In surgical robotics also cooperative control has been a current topic of research. A cooperative system allows performing surgical procedures interactively, meaning that the surgeon and robot share control [127]. One of the first surgical applications with robotic assistance was in stereotactic neurosurgery [107]. These systems can be included at the border of cooperative system classification. In such systems the robot autonomously positions and fixes a mechanical guide according to a pre-planned trajectory, and then the surgeon uses this guide to introduce the surgical tool (such as a drill, probe, or electrode) while the robot acts as a mechanical guidance imposing a simple and rigid linear constraint. Kwoh et al made the first attempt to use an industrial PUMA 560 robot for the CT-guided brain tumour biopsies [75]. Lesion location was determined from CT images and the robot positioned a biopsy needle using this data. Benabid and colleagues developed in the late 1980s an early precursor to the stereotactic robot marketed as NeuroMate [79]. The current version of NeuroMate (see Figure 3.5) is a commercial product that has been licensed by Integrated Surgical Systems (ISS) and approved by the Food and Drug Administration (FDA). The system has been used in over 1600 procedures since 1989.

Early experiences with surgical robots, such like ROBODOC and other similar systems, showed that surgeons found a form of *hands-on* control to be very convenient and natural for surgical tasks. Under this type of control, the robot undergoes steady-hand manipulations of the surgical instrument while the surgeon controls the whole procedure. The surgeon and robot are jointly performing tasks.

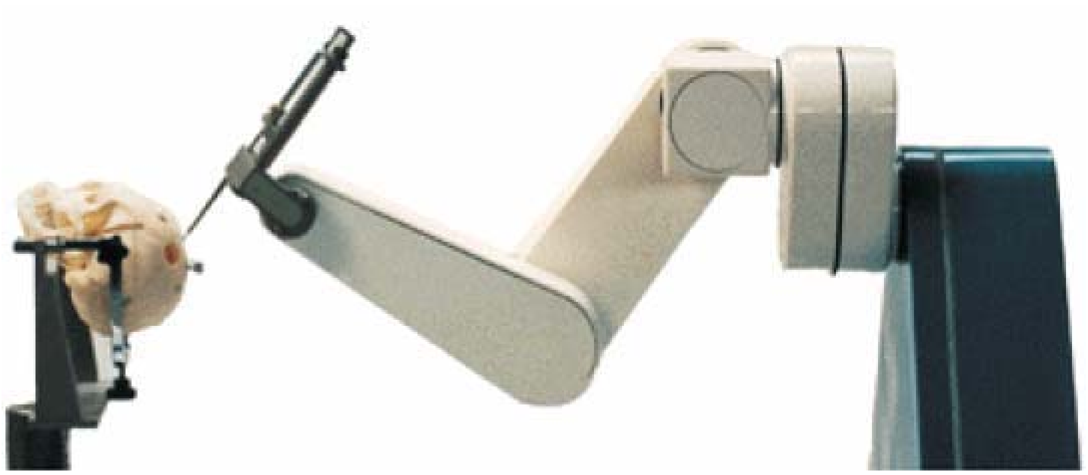


Figure 3.5. Neuromate courtesy of Integrated Surgical Systems

A number of groups have further exploited the idea of creating virtual constraints to help a surgeon align a tool, follow a precise path, maintain a desired force, prevent entering into certain forbidden regions of the workspace, or perform other similar tasks [86], [101], [80]. This concept is normally known as *virtual fixtures* (for further details refer to section 5.2). One example is the Active Constraint ROBot (ACROBOT), which is a small, low-powered, special purpose robot for knee surgery developed by the Imperial College at London (see Figure 3.7) [57]. This robot uses backdrivable motors and transmissions, so it has low mechanical impedance in each axis, allowing the robot to be moved by the surgeon with low force by pushing a handle mounted near the tip of the robot.

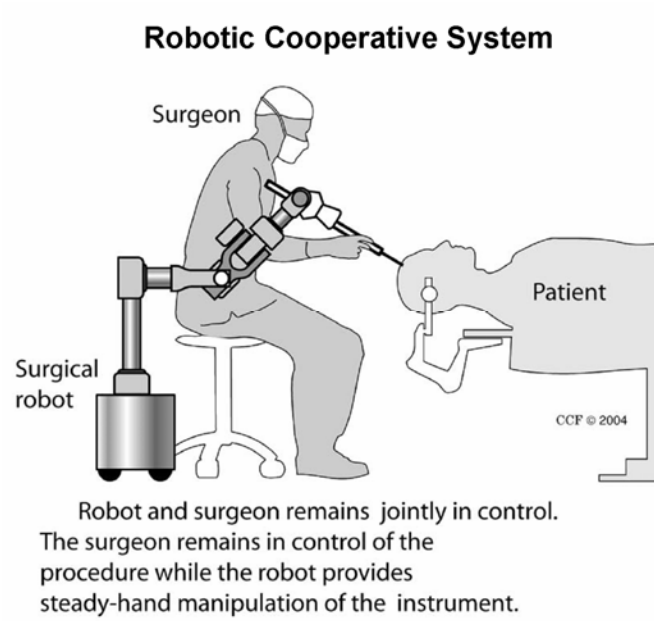
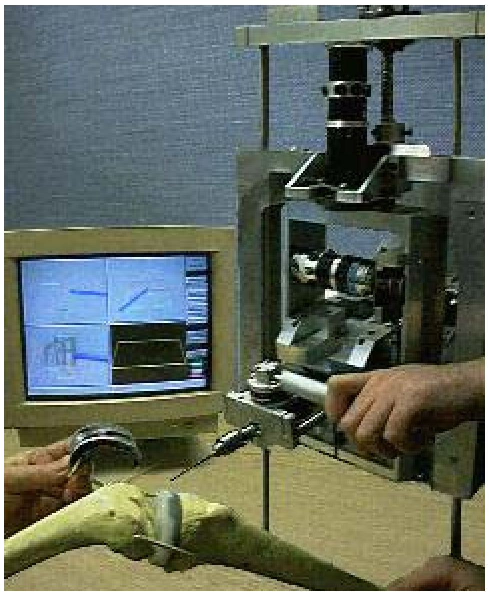

Robotic Cooperative System
Surgeon
Patient
Surgical robot
CCF © 2004
Robot and surgeon remains jointly in control.
The surgeon remains in control of the procedure while the robot provides steady-hand manipulation of the instrument.

Figure 3.6. Cooperative: robot and surgeon remains jointly in control [94]



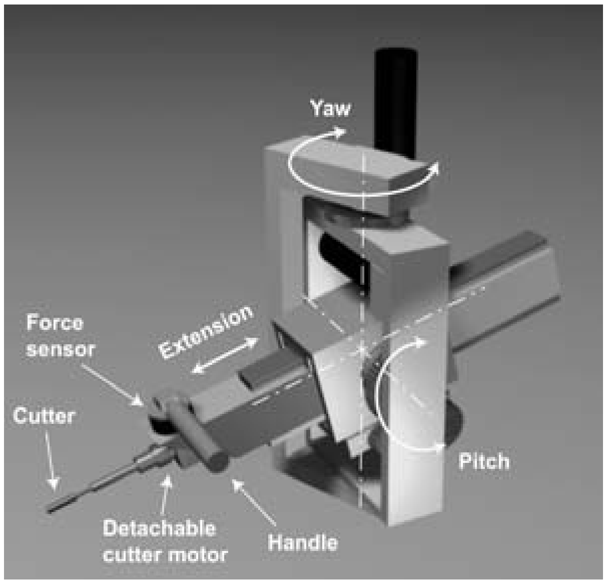


Figure 3.7. ACROBOT, special purpose Hands-On robot for knee surgery

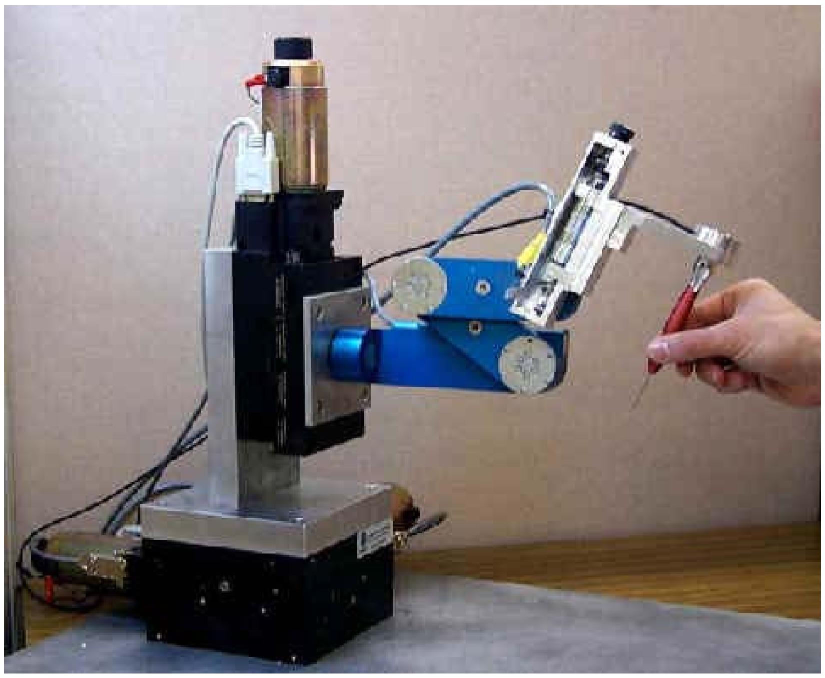


Figure 3.8. JHU Steady Hand robot for microsurgery

The robot is force controlled by adjusting the torque of the motors depending on the force applied by the surgeon, and the current position of the robot in relation to the cutting boundaries. As the user approaches and then contacts a constraint surface defined in the preoperative plan, varies the admittance, i.e. the relationship between the force applied by the surgeon and the torque of the motors, until the edge of the permitted region, where it prevents further motion outward the boundary [46]. Whilst the ACROBOT is currently being used for knee surgery, the system is also suited to a range of orthopedic and soft tissue procedures.

This concept of virtual fixtures has been more recently applied in the Johns Hopkins University (JHU) Steady Hand robot system for micro-manipulation [123]. It is composed of a Cartesian stage allowing three orthogonal translational DOF and a Remote Center of Motion (RCM) stage allowing two orthogonal rotational DOF (see Figure 3.8). This robot is developed to extend human's ability to perform small-scale (sub-millimeter) manipulation tasks requiring human judgment, sensory integration and hand-eye coordination. The tool is held simultaneously both by the surgeon's hand and the robot arm. The robot's controller senses forces exerted by the operator on the tool and by the tool on the environment, and uses this information in various control modes to provide smooth, tremor-free precise positional control and force scaling. Applications of this robot include eye surgery, microvascular surgery and neurosurgery [74].

There are other kinds of systems based on passive mechanisms that have been also implemented, like Cobots (from Cooperative robot) [141], which use mechanical rolling contacts to implement smooth constraint surfaces. The operating system is to use computer-controlled CVTs (continuously variable transmissions) to produce high quality rolling constraints. In some cases, the CVT is no more than a steered rolling wheel [36]. In other cases, the CVT may be a complex mechanism [37]. Although computer steering determines the path of a cobot end point through the cobot's workspace, the computer has no authority over the speed of the endpoint along that path. The speed is determined by the external forces, including those applied by the user and environment, e.g., gravity, and the inherent dynamics of the cobot itself. This means that cobots are passive devices, incapable of transmitting power to the user.

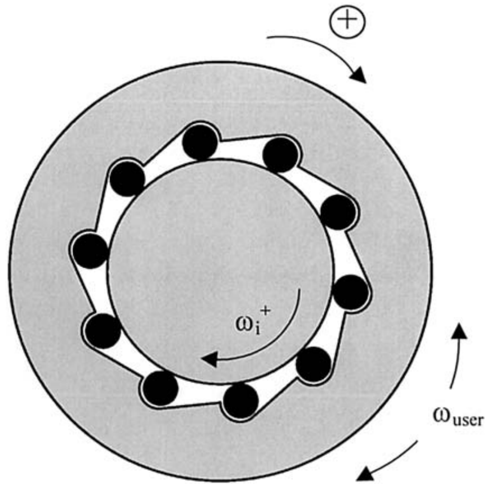


Figure 3.9. Free-wheel mechanism of PADyC

Other passive mechanism is the PADyC (Passive Arm with Dynamic Constraints) [113]. It consists of two free-wheels mounted in opposite directions in association with two motors at each joint in order to provide the different desired constraint effects. A freewheel is very similar to a conventional roller bearing, but it naturally provides the basic function of unidirectional motion. Consider the free-wheel mechanism of Figure 3.9, if the internal part of the free-wheel is fixed ($\omega_i^+=0$), the motion of the external part is blocked on the positive direction, while it is free in the negative one. If a motor is associated with the internal part of the free-wheel and rotates with velocity ω_i^+, then both directions of motion are allowed but ω_{user} is bounded by ω_i^+ in the positive direction. The combination of two free-wheels, with their corresponding motors, for each joint gives the possibility to control velocity in both directions.

The intrinsic safety of such a system is good. Indeed, the joint mechanical design integrates another set of free-wheels and worm screws that respectively guarantee that the arm cannot move autonomously, because the motors cannot drive the joints, and that the user cannot back-drive the motors. Moreover, the joints are naturally locked when unpowered. The operation principle of these passive devices is considered out of the scope of this work and it will be not further discussed in this contribution. Interested readers are encouraged to consult [36], [37], [112], [127], and [141].

3.1.3 Teleoperative systems

In teleoperative systems, well known as telesurgery systems, the surgical manipulator is under direct control of the surgeon with the surgical tools in the form of a robotic manipulator (see Figure 3.10). With an on-line input device that is typically a force feedback joystick (master), the surgeon performs the surgical manipulations, and the surgical manipulator (slave) faithfully follows the motions of the input device in a master-slave control manner to perform the operation [94].

Teleoperation in surgery comes primarily from the need to increase dexterity of the minimally invasive surgery (MIS) inside small body cavities. Telesurgery systems can provide better ergonomics compared with conventional MIS. The robot motions are

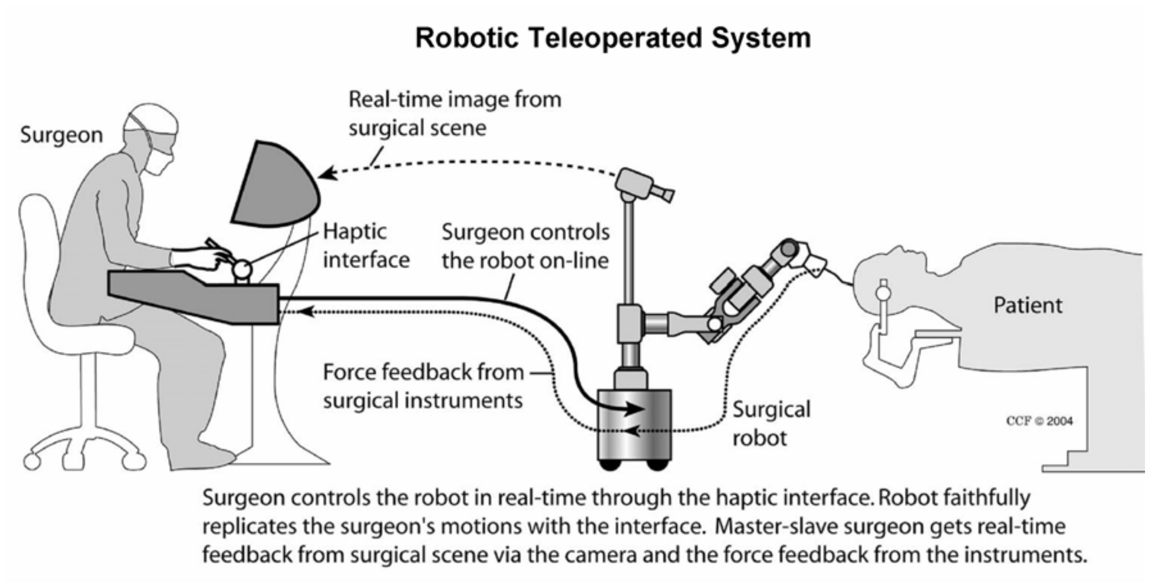

Robotic Teleoperated System
Surgeon
Real-time image from surgical scene
Haptic interface
Surgeon controls the robot on-line
Patient
Force feedback from surgical instruments
Surgical robot
CCF © 2004
Surgeon controls the robot in real-time through the haptic interface. Robot faithfully replicates the surgeon's motions with the interface. Master-slave surgeon gets real-time feedback from surgical scene via the camera and the force feedback from the instruments.

Figure 3.10. Telesurgery system: Surgeon controls the robot in real-time through the haptic interface [94]

specified directly by the surgeon on the basis of intraoperative images taken by the internal camera. In some cases, haptic feedback is also available, although limitations in the ability of current slaves to sense tool-to-tissue forces can somewhat limit this ability.

Teleoperated robots have been used for close to 15 years to assist surgeons in MIS, first, to assist laparoscopic surgery by holding an endoscope (e.g. [110], [121]) and later to manipulate surgical instruments [45]. A notably example of telesurgery systems is the daVinci system [45], by Intuitive Surgical, Inc., USA, which consists of the surgeon's viewing and control console, a control unit, and a three-arm surgical manipulator (see Figure 3.11.a). Although many tools are available, the most salient feature is a three-axis wrist (see Figure 3.11.b), which mimic the motion freedoms of the human wrist. Visual guidance is provided to the surgeon through a stereo endoscope and a 3-D visual display. The overall precision is improved by motion reduction scaling and by filtering involuntary motions caused by tremor.

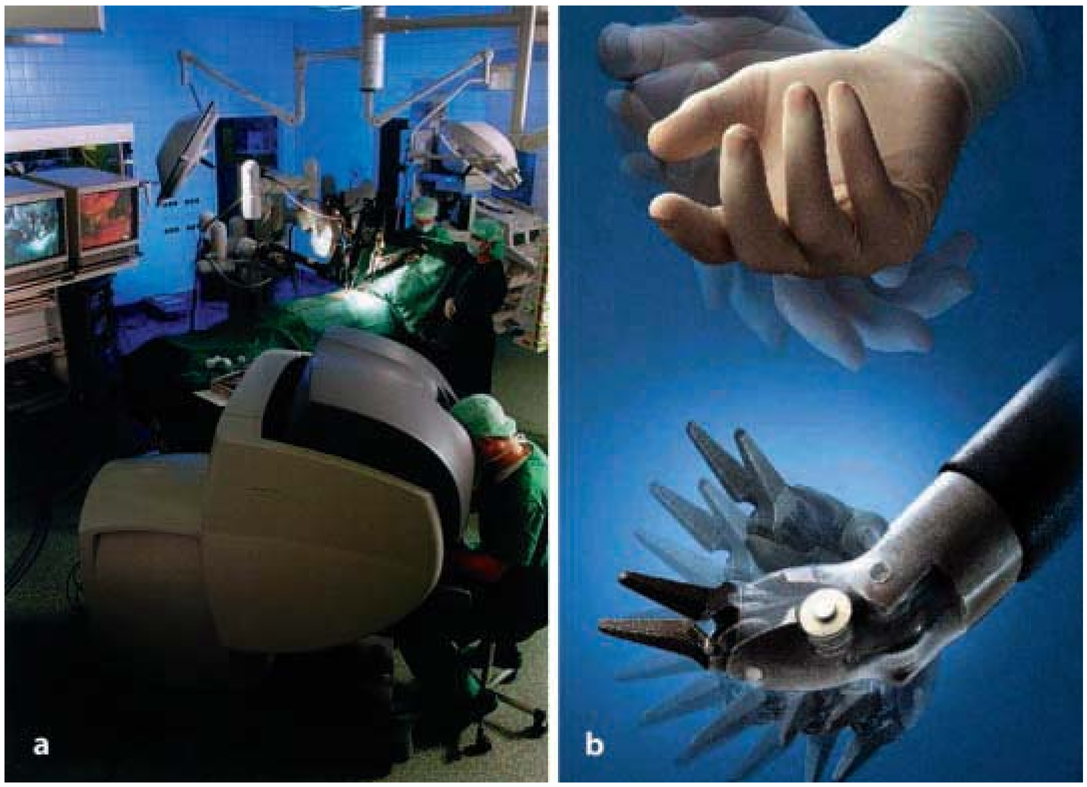
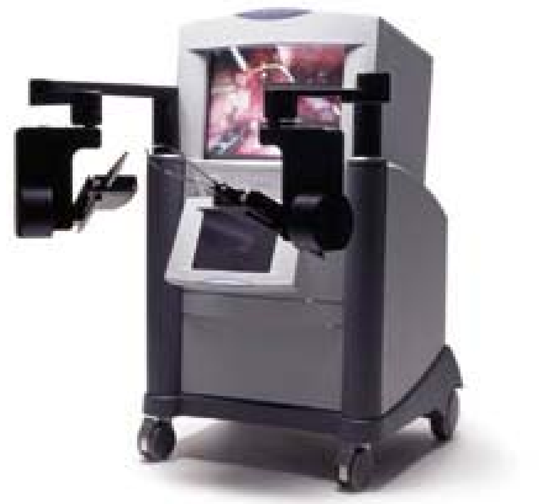


Figure 3.11. (a) daVinci telesurgery system, (b) Endoscopic EndoWrist^{TM} Instrument (courtesy of Intuitive Surgical)


(a)

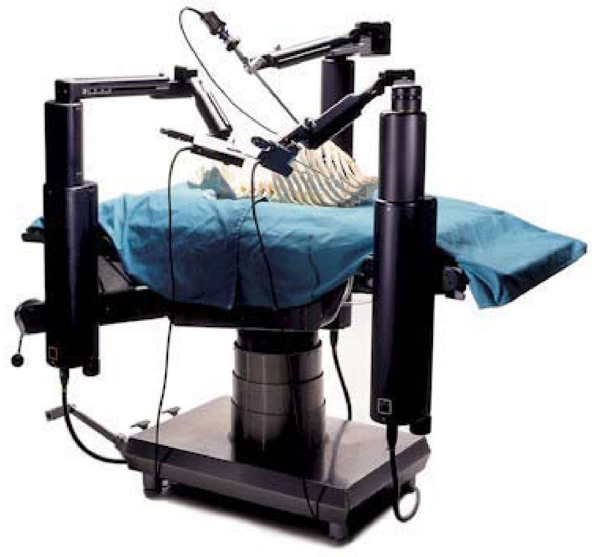

(b)

Figure 3.12. Zeus Telesurgery system from Computer Motion Inc.: (a) Console unit, (b) Zeus robot arms

A similar telesurgery system, called Zeus, has been developed by Computer Motion. This system is composed of a surgeon control console and 3 table-mounted robotic arms (see Figure 3.12). The right and left robotic arms replicate the arms of the surgeon, and the third arm is an AESOP voice-controlled robotic endoscope for visualization. In the Zeus system, the surgeon is seated comfortably upright with the video monitor and instrument handles positioned ergonomically to maximize dexterity and allow complete visualization of the OR environment. The system uses both straight shafted endoscopic instruments similar to conventional endoscopic instruments and jointed instruments with articulating end-effectors and seven degrees of freedom.

A notorious moment for the Zeus system was in February 2001, when a team of surgeons performed a transatlantic laparoscopic operation on a woman in Strasbourg, France, where the surgeon was operating from a hospital 6000 km in New York City, USA. The 54 minute operation was completed without any complications and the patient was discharged two days later [51]. The success of this operation as well as the technological infrastructure set in place highlight major developments in the field of telesurgery (see Figure 3.13).

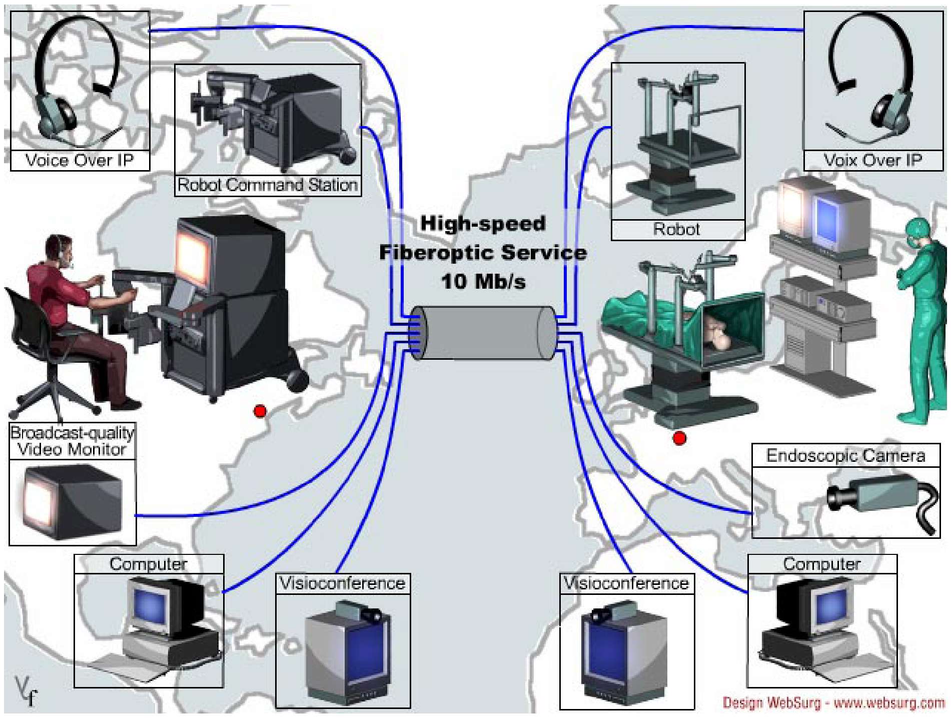


Figure 3.13. Setup of Zeus system at Lindbergh operation 2001 [102]

Technically, much remains to be done before robotic surgery's full potential can be realized. Although these systems have greatly improved dexterity, they haven't yet developed the full potential in instrumentation or incorporated the full range of sensory input. Beside the two commercially available systems, other research groups are working in order to further improve the capabilities of such systems.

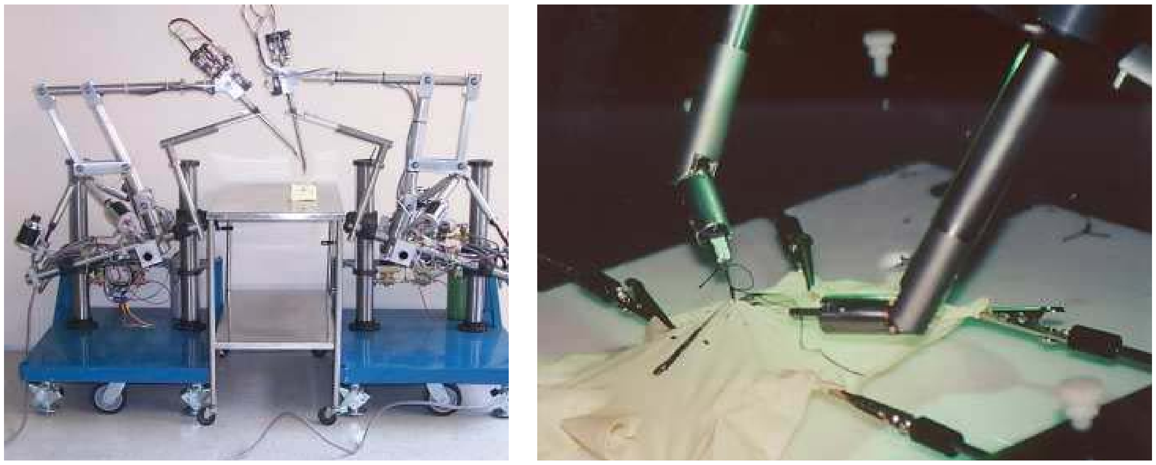
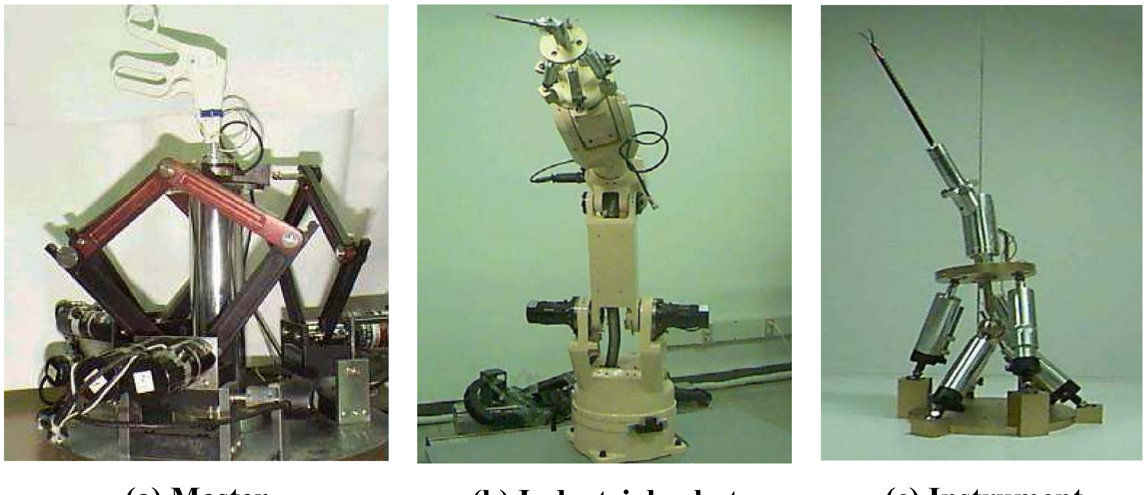


Figure 3.14. The telesurgical workstation for laparoscopy at Berkley [90]

Many future advancements are already being researched [78]. Mentioning some of them, the Berkley system, a joint project between the University of California, Berkeley and the Department of Surgery of the University of California San Francisco, USA is a telesurgical workstation for laparoscopy (see Figure 3.14). The slave is based on a modified Millirobot, while the masters are PHANToM devices. The design of the millirobot is dexterous enough to perform suturing and knot-tying tasks.
The KAIST system at the Korea Advanced Institute of Science and Technology (KAIST) is a microsurgical telerobot system composed of a 6 DOF parallel micromanipulator attached to a macro-motion industrial robot and a 6 DOF force/torque-reflective haptic master device.

The Research Center of Karlsruhe has developed the ARTEMIS system (Advanced Robotic and Telemanipulator System for Minimal Invasive Surgery) [107]. This system consists of the Man Machine Interface with two haptic manipulators, a graphical user interface, 3D video imaging of the operating environment, speech input for controlling the laparoscope, foot pedals and a trackball. And the Work Station with tow telemanipulation units, the TISKA carrier system with surgical effectors and the ROBOX endoscope guidance system.

Although this section is intended as a perspective on the field of medical robotics, it is no longer possible to produce a truly inclusive survey, and much excellent work has gone uncited.



(a) Master **(b) Industrial robot** **(c) Instrument**

Figure 3.15. The telerobotic system for mircrosurgery at KAIST

3.2 Virtual fixtures

The Virtual fixtures (VFs), also found in the literature as synthetic fixtures [111], virtual mechanisms [87], virtual tools [71], are software-generated force and position signals applied to human operators via robotic devices. They help humans perform robot-assisted manipulation tasks by limiting movement into restricted regions and/or influencing movement along desired paths [4].

The type of control strategy used to create virtual constraints may vary depending of the behavior of the physical systems. Along each DOF, instantaneous power flow between two or more physical systems (e.g., a robot and its environment) is always definable as the product of two conjugate variables, an effort (e.g., a force) and a flow (e.g., a velocity). An important physical constraint is that no one system may determine both variables. Thus, along any DOF a robot may impress a force on its environment or impose a displacement or velocity on it, but not both. Consequently, physical systems come in only two types: admittances, which accept effort (e.g., force) inputs and yield flow (e.g., motion) outputs; and impedances, which accept flow (e.g., motion) inputs and yield effort (e.g., force) output. Distinction between admittance and impedance is fundamental to apply the most adequate control strategy. In a dynamic interaction between two physical systems, one must physically complement the other: Along any DOF, if one is of impedance-type, the other must be of admittance-type and vice versa.

Robots can then be considered of either the impedance or the admittance type [5]. Robots of the impedance type, such as typical haptic devices, are backdrivable with low friction and inertia (e.g., PHANToM device). This type of robot can be considered a force source, and is typically controlled using impedance control. An impedance controller outputs actuator forces that are a function of measured robot position/velocity/acceleration. On the other hand, robots of the admittance type, such as typical industrial robots, are non-backdrivable and have large inertia or joint friction (e.g. robots with high rate transmissions in servo-motors). This type of robot can be considered a velocity source and is usually controlled using admittance control. An admittance controller measures an input force, and controls the position (i.e. velocity) as a function of the input force.

Different control strategies for the application of virtual fixtures have been development for both types of robots. Ho *et al.* [50] distinguished between two approaches, which they called the *implicit force control* and the *modified damping control*. In the former, no force sensor is used, and the robot is of the impedance-type. The latter approach uses a force sensor to measure operator's guiding force, which determines the robot's velocity. In this case, the robot is of the admittance-type and the desired velocity of the robot is controlled based on the relative position of the robot, motion constraint, and the direction and magnitude of the guiding force. This is basically the concept of admittance control techniques and can be applied to robots of the admittance-type in a very natural way. Since these non-backdrivable robots move in a highly controlled fashion, one can passively restrict movement in any given direction by simply not commanding any movement in that direction. Based on JHU Steady-Hand Robot, Bettini *et al.* [13], [14] uses admittance control to develop guidance virtual fixtures to assist the surgeon to move the surgical instruments in a desired direction. Their work was focused on 2D geometric guidance motion of the tool tip based on vision information. Funda and Taylor [41] formulated desired motions as sets of task goals in any number of coordinate frames relevant to the task, and optionally subject to additional linear constraints in each of the frames for redundant and deficient robots. Li *et al.* [80] extended Funda's work to generate virtual fixtures for real-time obstacle avoidance, and simultaneously assist the surgeon to perform desired tool motion to accomplish intended tasks by using an optimization-based approach.

Virtual fixtures have also been widely applied to telemanipulators, where a human operator manipulates a master robotic device, and a remote slave robot manipulates an environment while following the commands of the master [109], [104], [1]. Rosenberg [109] implemented virtual fixtures as impedance surface on the master to assist in peg-in-hole tasks. Joly *et al.* [58] simulate a virtual mechanism connected to the master and slave arms via springs and dampers to impose motion constraints to the system. Micaelli *et al.* [87] proposed a decoupled controller for telemanipulators to deal with virtually constrained and unconstrained directions defined by a virtual mechanism. Itoh *et al.* [56] proposes human-machine cooperative telemanipulation bases on the task-oriented virtual tool dynamics which assist a human operator semi-autonomously during a task. Turro *et* *al.* [134] implemented virtual fixtures projecting the operator's Cartesian position on the desired trajectory (called a Proxy) to which the master is bounded, and the slave then tracks either the master or the proxy, depending on the desired level of user control. The approaches just mentioned above were implemented with penalty-based or potential-field methods. These are impedance-type virtual fixtures that act in an active way. Abbott *et al.* [3] implement an admittance controller on teleloperators where the master and slave are impedance-type devices. The virtual fixturing method involves controlling an impedance-type robot using techniques that mimic admittance control. Using this method, a teleoperator of the impedance type, designed to achieve a good sense of telepresence, can also implement virtual fixtures without the stability problems commonly associated with implementing virtual walls using impedance control techniques [2]. Unlike with potential fields, the admittance-type guidance virtual fixtures act in a very passive way, because they do not add energy to the system.

3.3 Singularity robustness

The singularity problem is a well known problem already identified at an early stage of robotics research [135], [140]. Various ways have been devised to handle the problem of singularity, starting from the simple approach of switching into joint space control [119]. Others developed techniques to avoid the singularities [53], [87]. However, avoidance is not always possible when the robot is not redundant with respect to the task. The scope of this work is restricted to real-time singularity robust control methods for the case in which the reference trajectory is not known a priori and the robot is non-redundant. A well-known approach is based on the so-called damped least-squares (DLS) method [92], [137]. This method prevents the joint velocities from becoming excessively high near singular configurations by using a damping factor to control the norm of the joint velocity vector. However, the exactness of the inverse kinematic solution is sacrificed in order to achieve feasibility. Although various methods to compute an appropriate damping factor have been proposed to minimize the deviation error [137], [92], [84], all these methods produce a deviation from the desired end-effector direction in the neighbourhood of the singularity. Moreover, Kircanski *et al.* [68] performed a stability analysis of the DLS method in terms of second-order differential motion and showed that an algorithmic error exist along the singular direction. An overview of the methods for inverse kinematics based on DLS can be found in [34], [26].

Eliminating the component of motion along singular direction to avoid large joint velocity generation has been proposed [24], [25]. Together with the introduction of an operational space formulation, Khatib proposed as solution for kinematic singularities to treat the robot as a redundant mechanism with respect to the motion of the end-effector in the subspace of operational space orthogonal to the singular direction [69]. Control in this subspace is based on operational forces, while null space joint torques are used to deal with the control in the singular directions. Later on, Chang and Khatib [22] introduced the dynamically consistent generalized inverse and compared its performance with the one of the pseudoinverse when performing motions in the null space. Using the dynamically consistent pseudoinverse, the task and null space motion and forces are decoupled. Oetom and Ang [100] eliminate the singular components of the Jacobian, becoming redundant with respect to the task, and used the dynamically consistent inverse to invert the Jacobian. Null space was also used to escape from singularity. Experimental results were obtained with the PUMA 560. A certain trade off between exactness and achievability was necessary in moving out of a singular configuration into a non-feasible path. The dynamically consistent generalized inverse has been successfully used for controlling the null space of redundant manipulators [23].

Kieffer [66] showed that using a higher order approximation, paths passing arbitrarily close to the singularity can be tracked when the end-effector path parameter variable is included as a dependent variable in the formulation. An alternative to Kieffer's path tracking formulations has been proposed by Nenchev *et al.* ([95], [99]), known as the singularity-consistent (SC) path tracking method, which is based on the null-space technique commonly used for redundant manipulators [93]. This method guarantees path tracking at and around a singularity without deviating from the desired direction. The deterioration of motion ability at the singularity reflects on velocity only. Later, the same author proposed a reformulation of the null space based path-tracking method in terms of instantaneous motion, thus avoiding the requirement for path parameterization [97].

Finally, the Adjoint Jacobian approach is an alternative to the SC null space based approach. It was first considered by Senft and Hirzinger [114]. In this method, one splits the inverse of the Jacobian into the adjoint and the determinant of the Jacobian. One important assumption is that the determinant can be factorized. Tsumaki *et al.* [131] have shown that the SC null-space and adjoint Jacobian formulations are directly related with each other [96]. There is certain limitation of these two methods, since they can be applied to a single singularity. In [130] the approach is successfully applied to a 6 DOF robot arm. Motion in a uniform way was possible everywhere in workspace, except at double singularities, e.g. simultaneous shoulder and wrist singularities. A comparative study between SC and DLS [98] show that the DSL may destabilize the system along a degenerated singular direction, while the SC does not.

4. Design of a controller framework

Robotic assistant surgery systems are complex systems that involve many interacting components, including the software, sensors, human-system interfaces and so on. As such, they share the same underlying needs for good system design and engineering practice like modularity, well-defined interfaces, etc.

According to chapter 2 the modiCAS system requires the design and implementation of a modular software framework to provide a flexible usage of the different functionalities to cope different surgical application. Further requirements are software maintainability, scalability, reusability and a robust and flexible hardware interface. The remaining of this chapter describes a modiCAS controller software architecture designed to fulfil these requirements.

4.1 System architecture

The concepts of simplicity, flexibility and scalability represent key concepts so that the development of suitable solutions for various applications becomes feasible in a pragmatic way. A clear modularization of the different implicated tasks and a strategic distribution of them along the system framework, depending on their role within the system, are paramount issues to fit these requirements.

In this context, let us now distinguish between two types of tasks. The first are the application-tasks, which, as suggested by the name, are application specific tasks that belong to a high level implementation. These make use of lower level tasks to attain their goals. The application-tasks may be a sequence of steps required to complete a surgical procedure, or may represent a state-machine with interchangeable modalities that become available to the surgeon during operation.

The functional-tasks are the second kind of task and comprise all fundamental services that the system is able to provide. They are the low level tasks that the application-tasks use to complete their objective. The functional-tasks may be configurable but they are essentially fixed within the system and have an explicit aim, e.g. commanding the robot to reach a desired position in the Cartesian space. Some functions may look more like an operating mode, such like the virtual constrained cooperative mode or the tracking mode (see section 4.5).

The modiCAS controller software framework consists of a client-server architecture which literally separates the system physically into two parts: on the one side a multi-purposes server mounted on a real-time embedded target computer that contains all the available functional-tasks used for a proper interaction with the different components of the system, i.e. hardware interface, data acquisition, robot controller, trajectory generation, kinematics transformations, etc. These tasks may have a single execution or run periodically in an independent loop which here is referred to as *task-loop*. A priority level is associated with each task-loop so that the most time critical tasks can always take control of the processor when needed. The modiCAS system makes use of LabVIEW Real-Time (RT) Module to guarantee real-time behaviour [77]. The embedded target consists of a normal PC running the Venturcom Phar Lap Embedded Tool Suit, a real-time operating system [9].

The other main part of the system is the client, which runs on a host computer and communicates with the RT target through an Ethernet connection. This contains the GUI together with application oriented high level routines (application-tasks). These routines make use of the basic functionalities provided by the server (functional-tasks) to achieve a specific goal. The server is headless, so any action must be commanded by the client. Furthermore, the client is also responsible for receiving data coming from the server for display, storage or other processing. Thus, client and server interact with each other using a command-based network communication here referred to as the command interface. Figure 4.1 illustrates the different framework modules of the modiCAS system and their interrelationship. Each module is detailed explained in the remaining sections of this chapter.

4.2 Command interface

The Command Interface is an Application Programming Interface (API) used to get access to all functionalities available in the RT-Target of the modiCAS framework. It consists of four separate modules, two at each side of the communication: the Command Sender Module (CSM) and Data Receiver Module (DRM) at the client, and the Command Parser Module (CPM) and Data Transmitter Module (DTM) at the server.

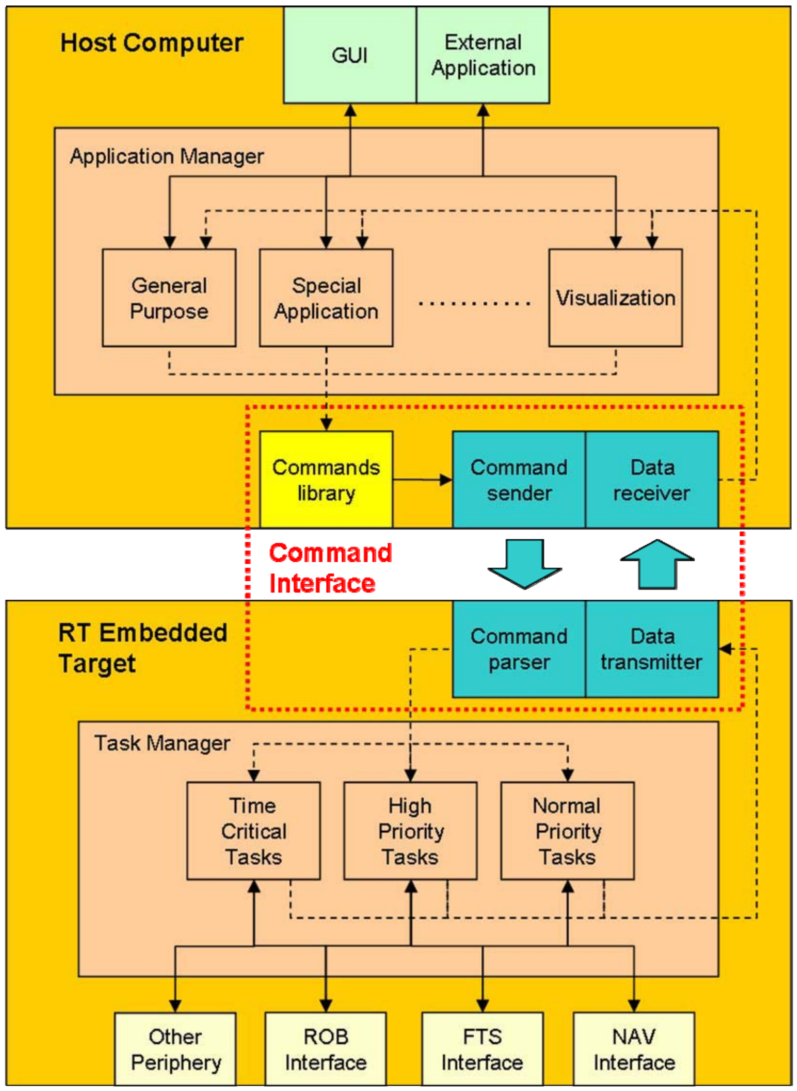


Figure 4.1. Command-based architecture of modiCAS framework

Additionally, a command library comprises all available commands that can easily be called by any routine of in the application-tasks. This abstraction provides a very clear interface for the application developer at the moment of implementing a new application. The explanation of the command interface begins with the message protocol used for the communication. This provides the basic knowledge needed to understand requirements for the construction and usage of the different commands available in the commands library. The further explanation of CSM and CPM as well as their interaction between each other is illustrated by tracking the data flow which occurs each time a command is executed. Finally, DTM and DRM are easier to understand, since they use the same principle of communication as CSM and CPM.

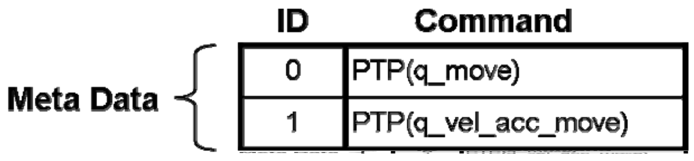
4.2.1 TCP/IP message protocol

The command-based communication is grounded on a simple TCP/IP Messaging Protocol, whereby the TCP/IP protocol is the most common method for sharing information between computers through a network. The communication protocol has the following characteristics:

- Easily packages and parses data
- Hides the TCP/IP implementation details
- Minimizes network traffic by sending data only when it is needed
- Minimizes impact on the overall overhead and throughput
- Ability to send and receive many data types

In every messaging protocol there is some data overhead associated with parsing the data stream on the receiving side. Sending a complete set of meta information with every package adds significant overhead. In order to minimize the communication overhead while sending essential information with each packet, the server creates a separated Meta Data List (MDL) containing one identification tag for each command associated to a command ID (created with the index of the command in the list). Each tag of the list corresponds to a unique and predefined type definition[1] used to parse the transmitted data. Figure 4.2 shows an MDL with only one command having two different instances, each of which corresponds to a different set of input parameters and has its own type definition. Notice however that both entries execute the same task.

[1] Type definition is LabVIEW-specific mechanism to identify the correct data type for each instance of a custom data structure.



Meta Data

ID	Command
0	PTP(q_move)
1	PTP(q_vel_acc_move)

Figure 4.2. Example of MDL of commands with two instances of the same command

The server sends the MDL to the client once at the very beginning of communication. Then, each time a command is transmitted, a packet is constructed using the format presented in Figure 4.3. Every packet includes 48 bits of overhead corresponding to the data size and command ID. These are concatenated to the command data. The transmission packet is converted into a flattened data string of binary values, adequate for TCP/IP network communication. The incoming package at the receiving side is then unflattened using the type definition corresponding to the specific command ID. This protocol is more efficient and has higher throughput when transmitting large data payloads.

4.2.2 Command library

The command interface provides an easy-to-use command library that can be used by the different applications in the host computer to transmit a command to the target. This permits the programmer to implement its final application using the available functions of the system without taking care of the complicated systematic details implicit in each task. Table 4.1 shows a list of the some general purposes commands available for their usage within the modiCAS system.

Data Size (32 bits)	Command ID (16 bits)	Data

Figure 4.3. Transmission Packet Format

Table 4.1. List of commands for general purposes functions

Command	**Description**
ABORT	Cancel any running operation
CVEL	Moves the robot in Cartesian space with velocity commands
EXIT	Finishes any running operation
GET[2]	Gets the value of a specified variable from the RT-Target
GVF	Enters in Guidance Virtual Fixture mode
INIT	**Initializes the command interface with the RT-Target**
JVEL	Moves the robot in joint space with velocity commands
LIN[2]	Moves the robot in Cartesian space with position commands
PTP[2]	Moves the robot in joint space with position commands
RESET	Clean any error message present in the RT-Target
SET[2]	Sets the value of a specified variable from the RT-Target
START	Sets the RT-Target into one of the different running modes
STOP	Stops the actual running mode
TRK	Enters in tracking mode
UNLOCK	Set/Release the brakes of the robot

The **INIT** command opens the TCP/IP network connection between host and RT-target. It has to be executed before any other commands can be used.

4.2.2.1 Command implementation

Normally, the main operation of every command (except **INIT**) consists of gathering the input data, converting it to string format and forwarding it to the CSM. It is the responsibility of each command to use the correct type definition according to the command tag to convert the input data into the right string format. If the string does not match with the type definition at the receiving computer, the command does not proceed. Figure 4.4 shows the typical internal structure of a command.

[2] Multiple instances available.

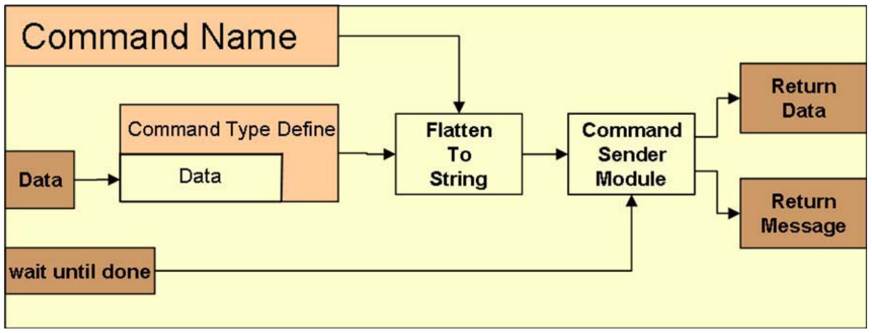


Figure 4.4. Command internal structure

Each command may have multiple instances, each of them with different number and type of arguments. As illustrative example, suppose that one application moves the robot using Point to Point (**PTP**) command. Desired joint position, maximum velocity and maximum acceleration can be given as input arguments of one command instance to define the movement profile. Then, if subsequent movements having the same profile are required, a different instance of the **PTP** command, with only desired joint position as input parameter, can be used and the last movement profile used is maintained by the server. Multiple instances can also be useful when different notations of an input argument are possible. For example, when moving the robot in Cartesian space to a target pose (position and orientation) with the **LIN** command, the desired orientation may be given either in the form of rotation matrix or any other notation, such as roll-pitch-yaw. A different instance of **LIN** could be implemented for each case.

The completion of some tasks at the target computer requires a certain not predefined time. For example, a **PTP** command requires the robot to reach the target pose before the command is considered to be completed. Depending on the application, an application-task may have to wait until completion of the commanded task before continuing with the next step or may continue doing other operations afterwards independent of whether the functional-task has been completed or not. The *wait until done* feature is included within the commands for such cases. If it is active, the CSM blocks until the corresponding completion-acknowledge is transmitted back by the server. A similar procedure occurs when feedback information is expected by the command. The CSM blocks until the data is transmitted back and then forward it to the command, which gives it as output.

4.2.3 Command sender module

The command sender module (CSM) executes the low level operations required to set the transmission packet in the correct format (see Figure 4.3). It is executed once each time a command is called. The tag of the command, together with the input flattened data string (if applicable), are given as input parameters. If the command name does not match with any of the MDL, this is ignored; otherwise, the data packet is constructed and then transmitted through the network connection.

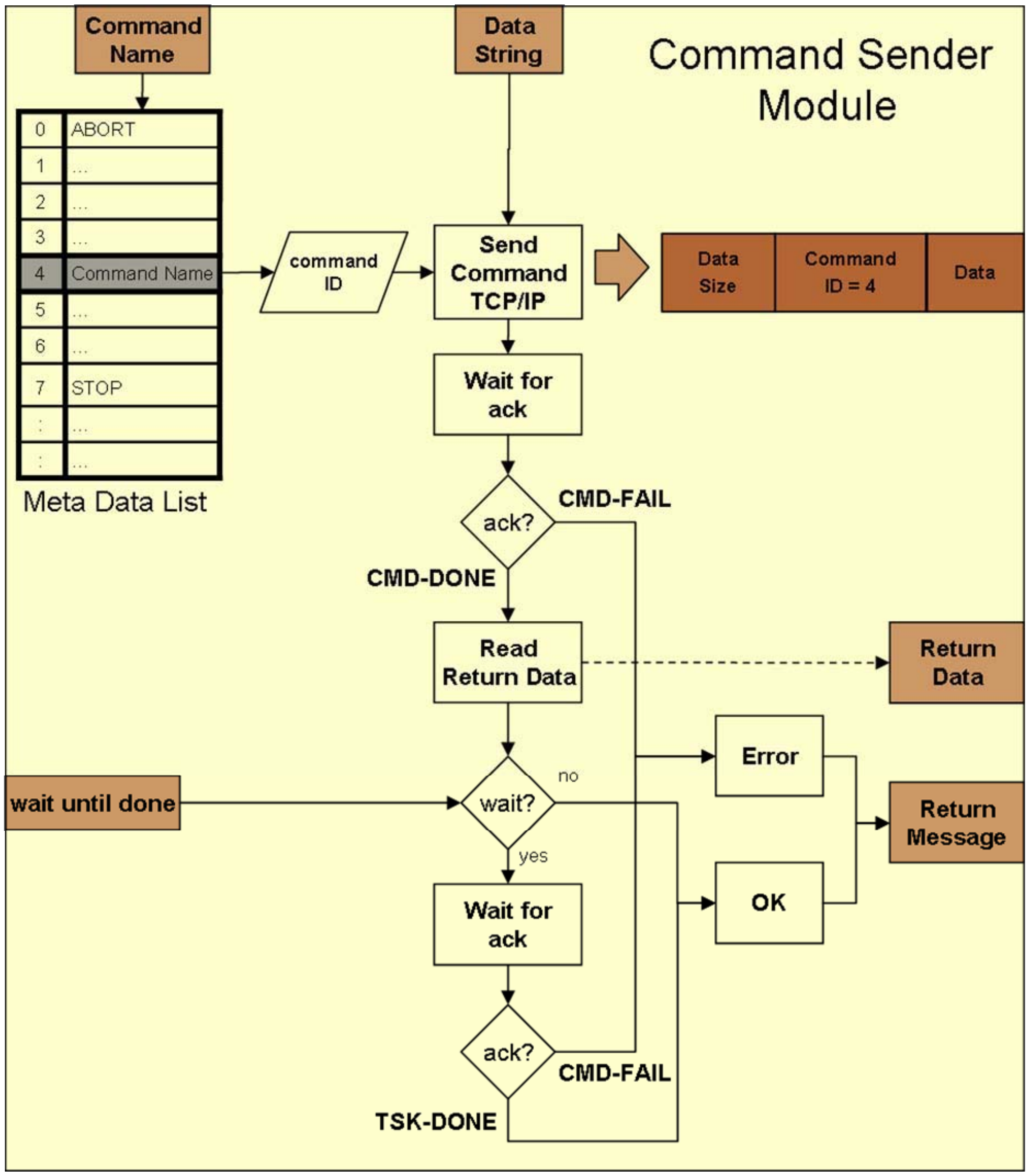


Figure 4.5 Interaction between command and command sender module

Once the packet is sent, the CSM waits for acknowledgment coming from the server. Three types of acknowledgment are possible:

- CMD-DONE – command successfully received and processed.
- CMD_FAIL – command failed.
- TSK-DONE – task successfully finished.

The CSM is unblocked only after a command execution (CMD-DONE) or a command fail (CMD-FAIL) is confirmed by the server. If the feature *wait until done* is active, the CSM blocks once again until the task completion (TSK-DONE) is also confirmed. If the command expects feedback data from server, such as in **GET** command, after the notification is received, the CSM looks for the data in the corresponding shared variables and forward it to the command function. The received data at this point is still in string format. The command function is responsible for decoding the data to the corresponding format.

4.2.4 Command parser module

The command parser module (CPM) is a key element for the safe performance of the system. It consists of an internal asynchronous loop running in the RT-target with normal priority. It sleeps until a new command needs processing. This assures an efficient performance since the CPM will consume very little CPU time if there are no incoming commands and it will not interrupt any other task with higher priority when a command has to be processed. Figure 4.6 shows the command parser module construction.

The TCP/IP message protocol explained in chapter 4.2.1 is used to retrieve the command name from every incoming data package. The feasibility of the command in consistence with the actual state of the system must be assured before this is processed.
The selection of the handler for the incoming command is implemented as a case selector with a separate case for each command. Such architecture is very scalable, because the incorporation of a new command simply requires adding a new case

matching with the tag of the command. Inside each case, the corresponding type definition is used to decode the incoming data.

Additional plausibility tests may also be required, e.g. joint limits, maximal desired joint velocity exceed, target position out of working space, and so on. The specific plausibility test depends on the type of command.

If any of these inspection procedures fails, the command is not executed and a warning message is transmitted back to the Host, otherwise, the CPM handles the command. Finally, it informs the Host of the successfulness/unsuccessfulness of the command execution. (CMD-DONE / CMD-FAIL). Notice that the commands where *wait until done* is active also expect the task execution acknowledgment (TSK-DONE) which may be produced by the corresponding task-loop.

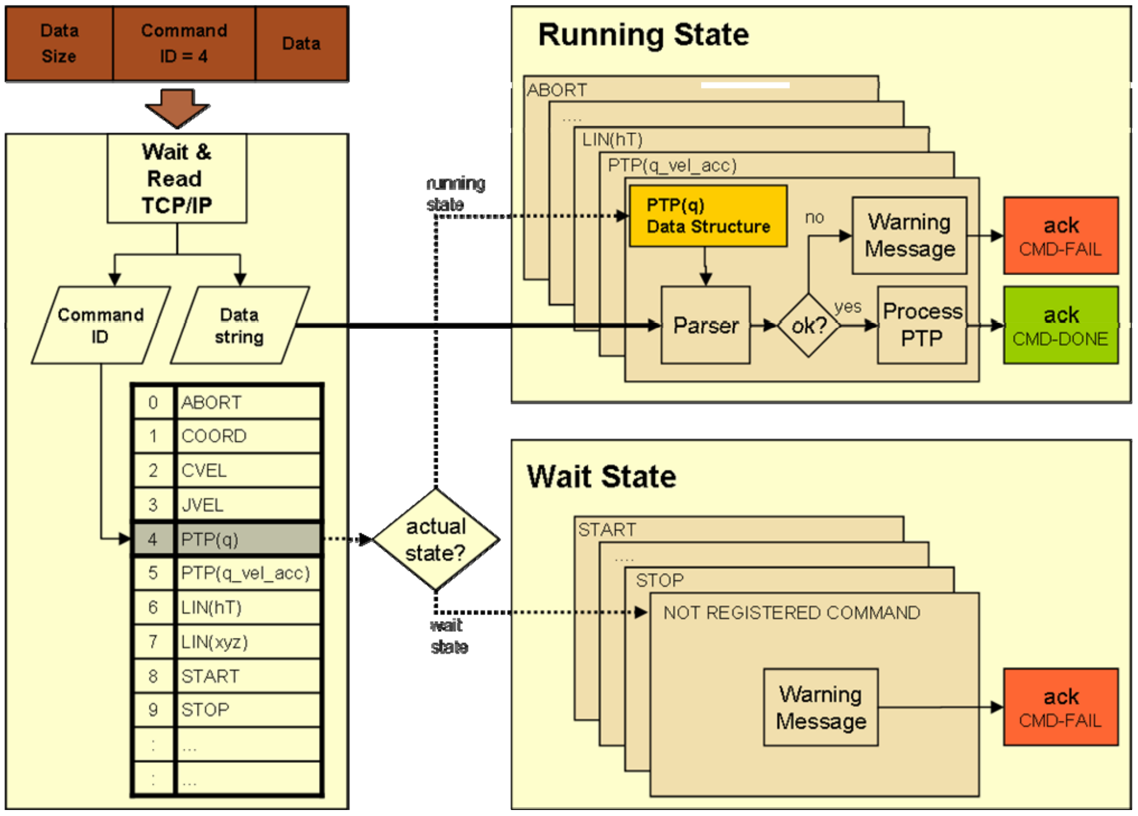


Figure 4.6. Command parser module: A PTP(q) command is received, but it is only processed if the current state is comprises the command

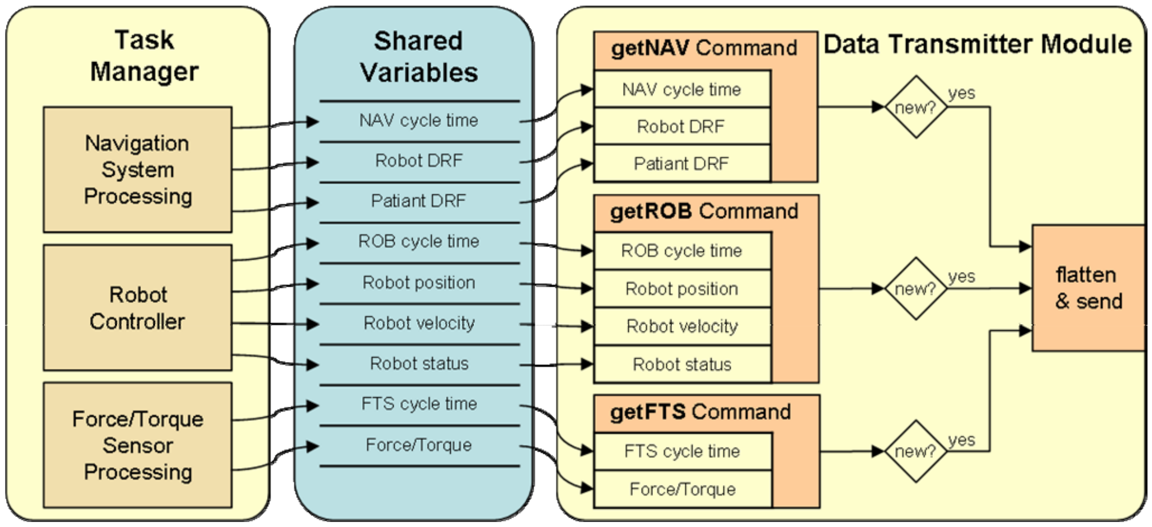


Figure 4.7. Data distribution inside the data transmitter module

The main purpose of the command parser loop is to manage available functional-tasks and distribute incoming data to the various task-loops[3] of the target application. In general, no other actions should be done inside the CPM. Restricting this loop only to task management and data distribution makes the target application very responsive to host commands.

4.2.5 Data transmitter module

The data transmitter module (DTM) is a normal priority loop running periodically in the target computer. This continuously sends important information to the host computer, such as current status of the different components of the system, error information, etc. The DTM retrieves data from other higher priority task-loops and forwards it to the client. The data processing and transfer occurs only when new data are available. This allows managing data coming from different asynchronous loops without sending repeated data to the host.

The conversion of data into flattened data strings for TCP/IP communication follows the same message protocol already explained in chapter 4.2.1. The separate transmission of a significant number of individual data values would decrease the efficiency of the communication due to the overhead included in each package.

[3] Communication with these tasks is implemented via real-time shared variables.

Therefore, all data values contained in each task-loop are gathered together into a type definition. Thus, one packet per loop is transmitted with a specific command name.

4.2.6 Data receiver module

The data receiver module (DRM) is responsible for receiving and eventually processing each incoming packet from the server. Its implementation is very similar to the one at the CPM. This module is driven at the rate of incoming data. This guarantees that no data is lost and no CPU time is wasted polling for incoming data. The execution frequency usually is much higher than in the CPM, since the data stream generated by the DTM at the server is transmitted periodically, as opposed to the sporadic sending of commands from the client.

In this situation, the host application must receive and process packets at high rates. Processing may involve logging to disk, performing some analysis, etc. Ideally, packet processing should always be completed in time to go back and retrieve the next packet. Therefore, normally the processing inside the DRM is limited to writing into shared variables and eventually some notation conversions. If additionally processing is required, it may be necessary to send the data to asynchronous task-loops that handle each particular operation. The data distribution to the different task-loops is done through functional variables[4] available for all running loops. Figure 4.8 shows the implementation of the DRM.

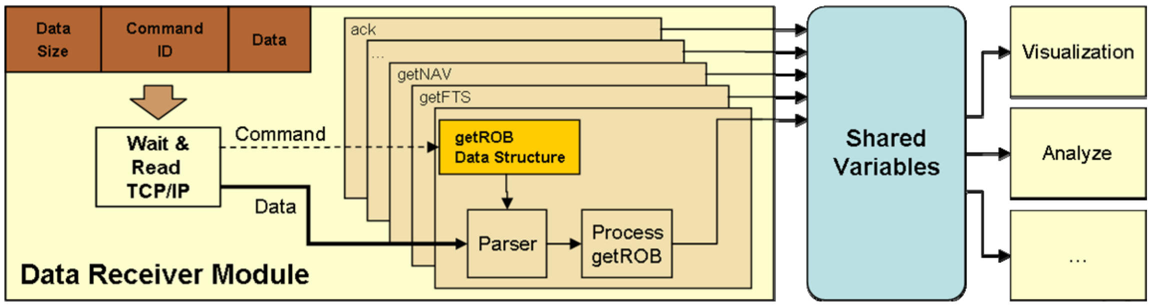


Figure 4.8. Data receiver loop at a Host application

[4] Communication mechanism in LabVIEW that allow controlled access to data or resources.

4.3 Target computer

The Real-Time (RT) Embedded Target acts as a server which provides all system fundamental tasks that require a real time behavior, such as hardware interface, data acquisition and processing, control loops, among others. High determinism is a characteristic of real-time systems and guarantees that the calculations and operations occur in time and on time. Deterministic applications are valuable not so much for their speed, but rather for their reliability in consistently responding to inputs and supplying outputs with very little jitter.

Now, it is important to differentiate between deterministic tasks and non-deterministic tasks. Therefore, each task has to be evaluated to define whether it is time critical or not. For instances, a control loop and safety monitoring are considered time critical because both need to execute on time every time to ensure accuracy. Communication between computers is not time critical because a computer may not respond on time every time. Likewise, data logging is not time critical because an accurate time stamp can identify when the data is collected or calculated.

The server program comprises different tasks with different execution priorities depending on how deterministic each task has to be. The concept of multithreading is required in order to apply these priority levels to the different tasks. Multithreading expands the idea of multitasking. The latter refers to the ability of the operating system to quickly switch between tasks, each of them be an entire application, giving the appearance of simultaneous execution of those tasks. Each application runs for a small time slice before yielding to the next application. Multithreading extends this idea into the applications, so that specific operations within a single application can be subdivided into individual threads, each of which can run in parallel. Thus, in a multithreaded program, the application might be divided into various threads, each of which has a priority level. This is useful in the case where some of the tasks must behave deterministically while others do not.

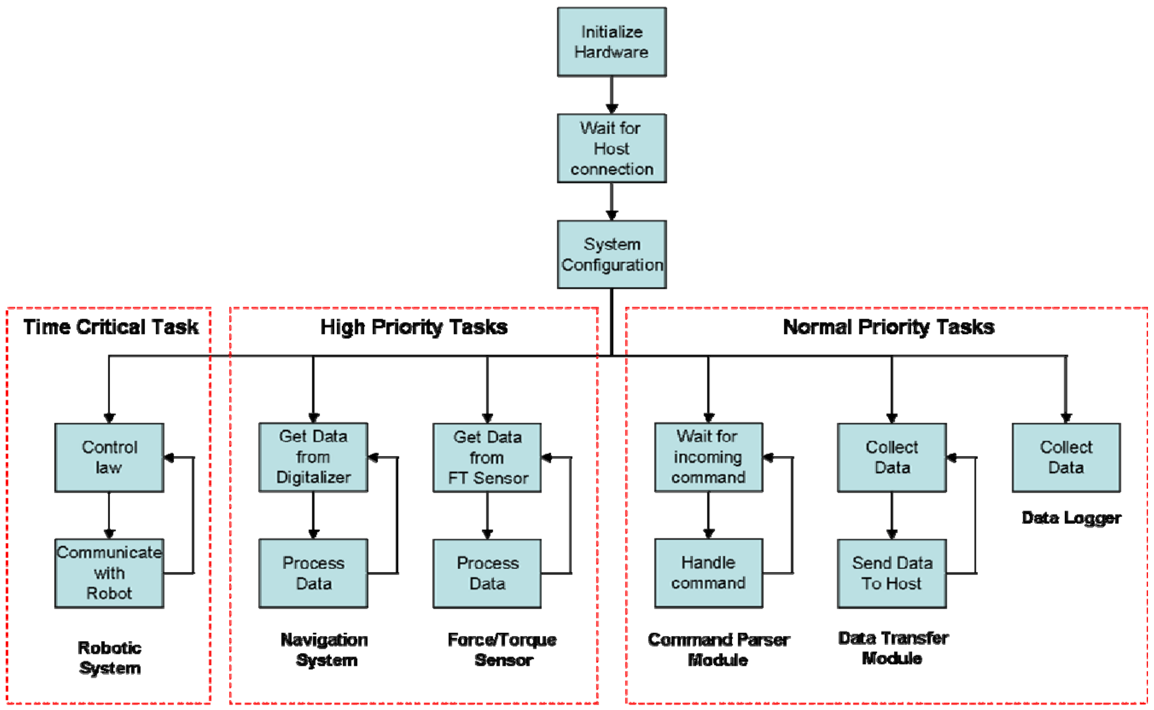


Figure 4.9. Task distribution among the different threads in RT-Target application

The real-time operating system of the LabVIEW Real-Time Module implements a combination of two methods for scheduling threads [77]:

- Round-robin scheduling – Applied to threads of equal priority. Equal shares of CPU time are allocated between the equal priority threads.
- Preemptive scheduling – Any higher priority thread that needs to execute immediately suspends execution of all lower priority threads and begins to execute.

Figure 4.9 shows the task distribution among the different threads depending on its priority level. One thread runs for each priority level. All tasks of the same priority are executed in the corresponding thread. The modiCAS framework uses three different priority levels:

- *Normal Priority Thread*: The tasks included here are non-deterministic and only execute while the deterministic tasks are sleeping. Basically, the command interface modules (CPM and DTM) belong to this category. Additional tasks, such as data logging, may also have this priority.

- *High Priority Thread*: This thread includes multiple tasks requiring a deterministic behavior. Although multiple tasks are running in the same thread, these can be executed in asynchronous loops, each of which having different cycle time. Examples of tasks running with this priority are the data acquisition and signal processing of the FT-sensor and the navigation system.
- *Time Critical Priority Thread*: In general, a deterministic application has a primary deterministic task that preempts all others. The time critical priority thread contains this task, with the particular characteristic that if any task running in here goes to sleep, the entire thread would sleep too. Hence, other tasks running on the thread would be forced to sleep and cease execution until the original loop wakes up. Therefore, only one task, namely the most critical one, runs with this priority. This task is the control loop of the robotic system.

4.3.1 State machine

The execution flow of the different states of the modiCAS system is controlled by the state machine (see Figure 4.10). The possible active tasks at a given time vary depending on the actual state of the system and the demands of the running application. A description of each state is given below.

Initialization-state: This state executes only once, directly after launching the main process. Hardware initialization and default internal variables setup happens in this state. After successful initialization, the process changes to configuration state.

Configuration-state: After entering the configuration-state, the process sleeps and waits for client's attempt of connection. Once this occurs, the MDL is transmitted to the client and application-dependent configuration parameters coming from the client are loaded. Some of these parameters describe which task-loops are to be used by the particular application at running-state, tool parameters (tool center point, center of mass, weight, etc.), assignation of RBs for the navigation system to recognize, among others.

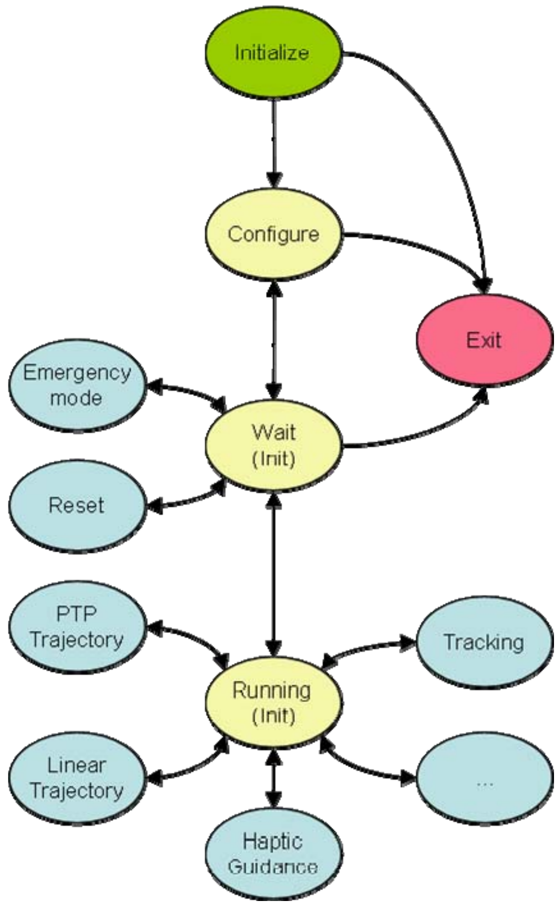


Figure 4.10. State machine of RT target

Finally, the command interface is initialized, i.e. the connection ID for the TCP/IP communication is forwarded to the CPM and the DTM. After successful initialization, an INIT-DONE acknowledgment is transmitted to the client and the process is sent to wait-state. If some error occurs in between, the connection is closed and the process is directly sent to exit-state where it is properly terminated.

Wait state: The wait-state is the previous state before getting the system into operation. At this point, only a limited number of special tasks can be executed:

- Reconfiguration of the system – At any point of a procedure, the host may require changing the settings of the system (e.g. when changing tool) in order to fit the demands of the particular application.

- System reset after error – If some recoverable error occurs during operation, the system automatically changes to wait-state and notifies the host about the source of the error. Leaving this state is only possible after the system is reset to normal status, i.e. each component of the system (robot, navigation system, etc.) must work properly. Only then, the system can enter the running-state. If the error is unrecoverable, the system switches to exit-state where all resources are closed and the process is terminated.
- Emergency mode – A special mode is available during wait-state that permits to unlock the robot joint brakes without starting the servo-motors. This may be required for emergency situations where the joints have to be unlocked and moved by hand.

Running-state: All system functions become available at running-state. The client application accesses them by means of the command library. Basically, when entering running-state, the server runs the task-loops required to provide the different functions (see section 4.5 for a description of available functions). A task manager is used to administrate the different active task-loops during this state. The task manager is further explained in the following section.

4.3.2 Task manager

The task manager is a task container which, depending on the demands of the actual client application, takes care of the setup, starting and termination of the required task-loops. Further configuration parameters are forwarded to the respective task-loops. This means, if the client application requires specific components of the system, this is simply specified within the configuration parameters either at client-server connection (during configuration-state) or later on in the wait-state. Once the running-state becomes active, the task manager immediately launches the required task-loops, distributing the configuration parameters to each of them in the form of input arguments. Each loop then executes cyclically, with the cycle time specified within the configuration parameters. If one of the task-loops is terminated, either due to error occurrence or commanded by the client, the task manager notifies the remaining task-loops in order to terminate each of them and finally conclude the running-state.

4.3.3 Task-loops

A task-loop is a piece of code that executes cyclically inside a loop once it is started by the task manager. Its purpose is to carry out one or more specific tasks. The task-loop is divided into three main stages:

- *Initialization stage*: This stage executes only once at the very beginning when the task-loop is started. Any kind of operation required for the correct execution of the task is performed here. Depending on the specific task, such operation could comprise variable initialization, hardware initialization and task configuration among others.
- *Execution stage*: This stage runs periodically until it is terminated either because of internal error or by the task manager. This stage comprises the operations required to perform the tasks. If multiple tasks are available, these are separated in sub-states that can be alternatively switched. Common operations for all tasks can be included before and after the sub-state execution (see Figure 4.11).
- *Termination stage*: This stage executes only once before the task-loop is completely terminated. All hardware and software resources can be safely closed before leaving the task-loop.

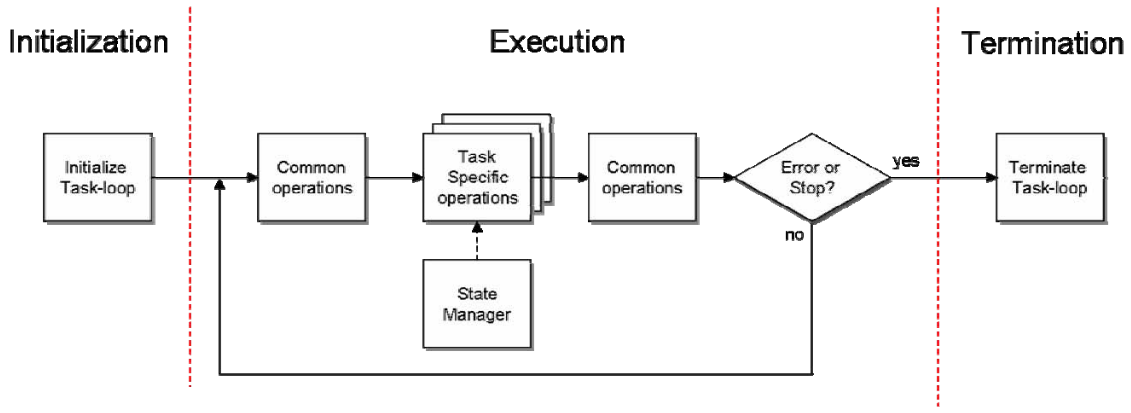


Figure 4.11 - Task loop flowchart

One single task-loop dedicated to each active component of the system is recommended but not mandatory. The incorporation of new components to the system would imply the implementation of its corresponding dedicated task-loop. All tasks related to this component are included in the form of sub-states.

A task-loop containing multiple states has to have one transition sub-state (Init) which has to be executed once each time the loop goes from one state to the other. The Init sub-state executes any procedure required to assure a safety transition from one sub-state to the other. A state manager (see section 4.3.4) is used to safety switch between the different sub-states going through the Init sub-state in between.

Three main components are available in the actual stand of the modiCAS system, each of them having one dedicated task-loop:

- *TCL-ROB Task-loop*: The Time Critical Loop of the ROBot arm (TCL-ROB) contains the main control loop. Several sub-states are internally available, each of them having an adequate control strategy to execute a specific task. Thus, it is possible to fulfill various demands coming from the different applications. An explanation of each sub-state is given in Section 4.5, where the available functions of the system are explained.
- *HPL-FTS Task-loop*: The High Priority Loop for the data acquisition of the Force/Torque Sensor (HPL-FTS) has a priority level lower than the TCL-ROB, i.e. the former is preempted each time the latter executes. In the same way, the HPL-FTS will preempt all tasks with lower priority, such as the CPM, the DTM and all existing normal priority tasks. All other task-loops with the same priority level (high priority level) are scheduled using the Round-Robin method. The HPL-FTS takes care of the data acquisition and signal conditioning of the F/T sensor mounted at the end-effector of the robot arm. The signal conditioning steps comprise: voltage-to-force transformation, mean value calculation, drift compensation, filtering and gravity compensation. Notice, however, that the task-loop contains only one sub-state (i.e. non state manager is required), since each calculation is performed sequentially one after the other every cycle time.

- *HPL-NAV Task loop*: The High Priority Loop for the NAVigation system (HPL-NAV) has the same scheduling behavior as the HPL-FTS task-loop. This task-loop is responsible for the acquisition and processing of data coming from the navigation system. The resulting data is written to the corresponding shared variables so that these can be retrieved by other task-loops. A detailed description related to the navigation system is out of the scope of this work. The interested reader may consult [106].

Notice that additional task-loops can be incorporated for other aims, such as analysis or data logging.

4.3.4 State manager

A state manager is an interface module used to remotely switch between the different sub-states of a specific task-loop. It assures a safe transition from one sub-state to the other passing first through the Init sub-state (see Section 4.3.3). There exists one state manager per each multi-states task-loop. If a task-loop only has one sub-state, it requires no state manager at all.

In the actual status of the modiCAS framework, only the TCL-ROB task-loop provides multiple tasks. Hence, only one state manager is available. However, the concept can be applied to any further task-loop with multiple sub-states if required.

On the one side, the state manager can be called from any running loop intending to change the actual sub-state of the associated task-loop. The CPM illustrates a very common case where the state manager is called any time a new command coming from Host demands the TCL-ROB to switch between sub-states. On the other side, the associated task-loop also calls the state manager internally at every cycle time to identify the actual sub-state.

The usage of this module inside the time-critical priority loop simultaneously with another lower-priority-loop without a synchronization mechanism could compromise determinisms, since if one loop accesses the module, no other loop can access it until the first loop releases it. When the access of the time-critical loop to its state manager module is blocked, forcing the loop to wait, this introduces jitter to the application and compromises its determinism. In order to avoid this behavior, the time-critical loop skips the module if another loop is using it, and the output value obtained the last cycle is used instead. This mechanism is not needed in lower priority loops; these can wait until the module is released.

4.4 Hardware interface selector

Keeping the framework flexible for expansion and maintenance is a major objective of this work. This counts also for the hardware interface. Thinking about upgrading or even replacing any component of the system (whether it is a data acquisition board, the navigation system or even the robot) must not affect the integrity of the software framework. Therefore, the interface to each main component of the system (robot, navigation system and force/torque sensor) is encapsulated into a selector object. This object may contain multiple instances of the same component, each of which contains a technical variation of the specific hardware. But all instances inside a selector represent the same component.

For example, the hardware interface of the FT sensor depends on different factors, such as sensor manufacturer and model, sensor calibration, data acquisition board type and so on. If several FT sensors are at disposition, the system should be able to work properly with all of them; moreover, the system should not notice the difference when using any of them. Each instance of the FT Sensor Selector corresponds to each of these available variants and contains the respective implementation details. If a new sensor becomes available, a new instance with the particular implementation is simply introduced to the FT Sensor selector.

Each instance of a selector has a singular name through which it can be identified dynamically and selected during program execution. The selector object comprises a set of services common to all instances. The task-loops utilize such services for interaction with the component. Number and type of services depend on the system component. Providing that some instances support more services than others, if a service is called which is not supported by the running instance, the selector notifies it with a warning message.

Notice that this abstraction also gives the possibility to completely substitute any component by a virtual analogy. In other words, it is possible to simulate each component of the system without making it notable for the rest of the program. This may be useful for different purposes at different levels. For instance, at the development level simulation may help for performance analysis of some components of the system. At the application level, a plausibility analysis of some trajectories inside the working area can be done with a simulated robot before these are applied to the real robot.

Interfaces for the robot arm, the navigation system as well as for the force torque sensor are implemented within the modiCAS system using the interface selector concept just described in this section. However, detailed explanation of the interface with each component of the system is out of the scope of this work

4.5 Target functions

The modiCAS framework design as explained so far has the main objective of making a set of diverse functionalities available to a higher level of development, namely, the application layer, where these can be used to support different surgical applications. It has already been pointed out that the application layer is located at the host computer and uses a set of commands to get access to these functionalities. In the following sub-sections, some control strategies are presented which are directly related to some of these commands. This means, each time that a command is called, the corresponding control strategy is activated. TCL-ROB, HPL-NAV and HPL-FTS task-loops contain the implementation of the various strategies. Some of them require only one task-loop while others need the collaboration of more than one.

4.5.1 Joint velocity controller

The joint velocity (JVEL) controller is the most simple control strategy implemented within the modiCAS framework. It practically forwards the desired velocity value to the robot servo driver. This value is expected to be in the joint space. The controller checks position and velocity limits before it forwards the velocity set point. The reference signal applied in this state can have different sources, such as the GUI at the host, or other

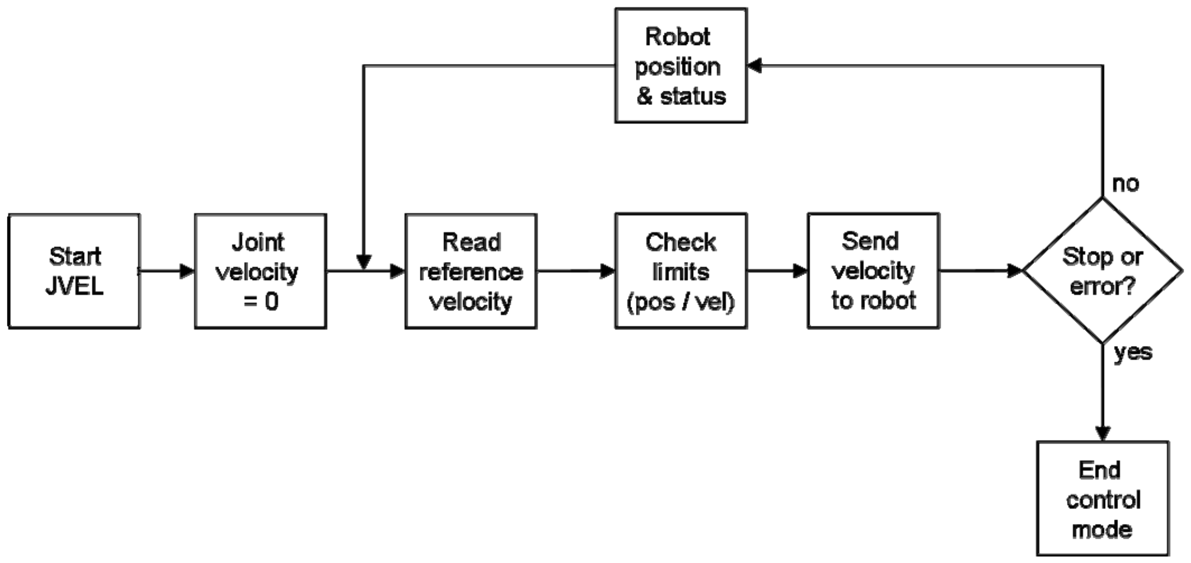


Figure 4.12. Flow diagram of Joint Velocity Controller (JVEL)

peripheral device, like space mouse or joystick. Figure 4.12 shows the flow diagram of the joint velocity controller, which is activated any time the **JVEL** command is used.

4.5.2 Cartesian velocity controller

The Cartesian velocity (CVEL) controller responds to the **CVEL** command. It accepts velocity set points in the Cartesian space, those that are transformed to the corresponding joint space. The Singularity Robust (SR) inverse velocity kinematics further explained in chapter 0 makes possible to pass through singular robot configurations without producing extremely high velocity values that could lead acute movements of the robot. Notice also that the command velocity can be applied to different reference frames that may correspond to the Tool Center Point (TCP) of any given tool mounted at the end-effector. Figure 4.13 shows the corresponding flow diagram of the implementation.

This control strategy may be useful in such applications where teleoperative manipulation of the robot arm is planned, where the robot represents a slave mechanism been remotely controlled by a master device, such as a Phantom device. Notice that the concept of virtual constraints can be included easily to delimitate the workspace and avoid entering in forbidden regions.

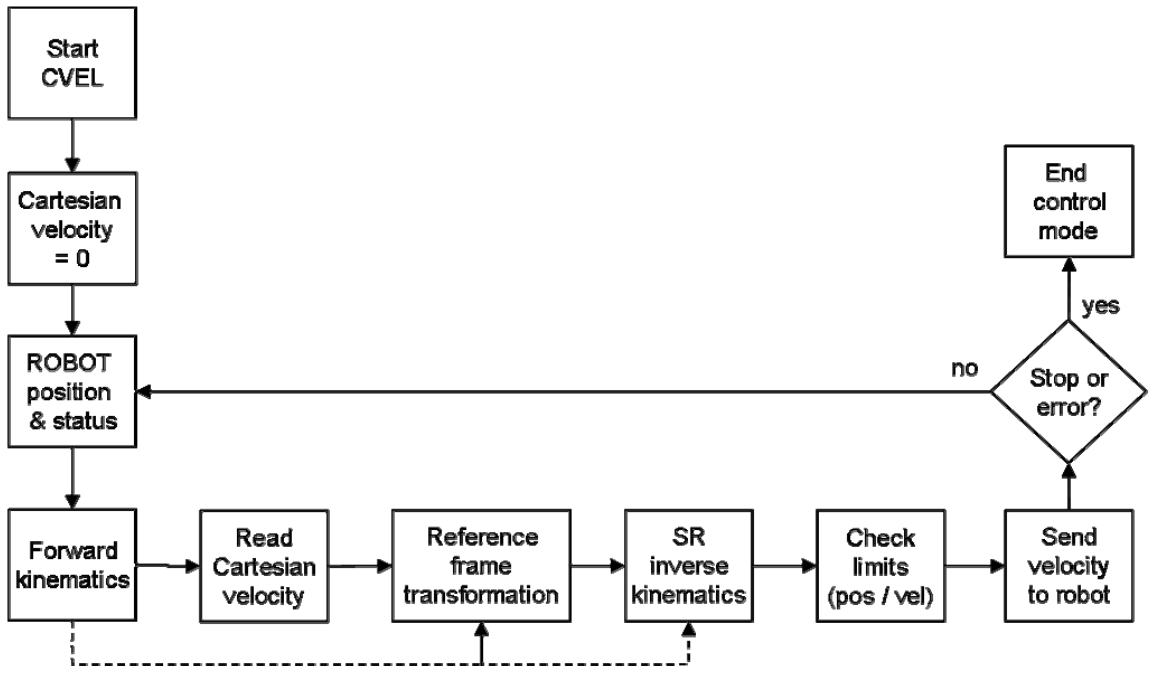


Figure 4.13. Flow diagram of Cartesian Velocity Controller (CVEL)

4.5.3 Joint position controller

The robot is intended to reach a desired position by following a position reference in the joint space. Therefore, a synchronous minimal traveling time Point to Point (**PTP**) trajectory in the joint space with trapeze velocity profile is generated [65]. Then a controller in the joint space is applied to follow such trajectory. Each time the host submits a new desired set of joints position, the controller resets the parameters of the interpolator so that it starts to deliver the interpolated trajectory on-line during the next cycles until the goal position is reached. A position controller (only proportional action) takes care of calculating the velocity command needed to follow the reference position which is finally transmitted to the servo driver. Once the robot reaches its aim, an acknowledgment signal (TSK-DONE) is generated and transmitted to the client (see Figure 4.14).

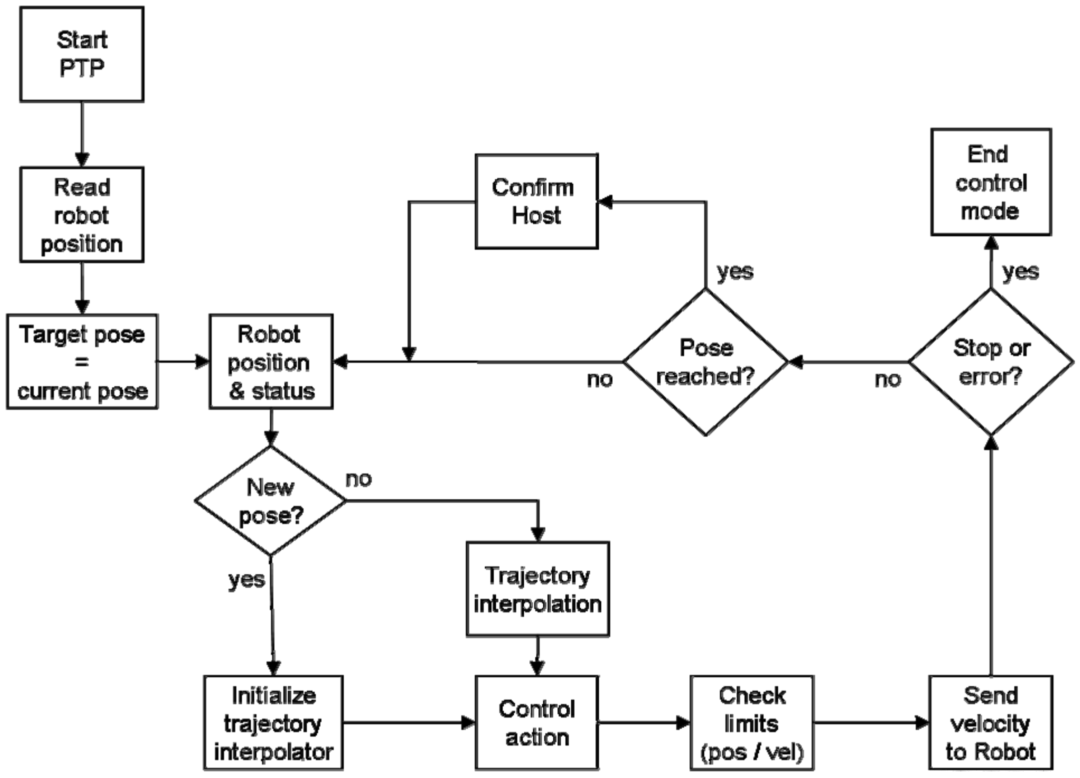


Figure 4.14. Flow diagram of Point to Point joint position controller (PTP)

4.5.4 Cartesian position controller

The Cartesian position controller, activated with the **LIN** command, is implemented to follow linear trajectories defined in the Cartesian space. Therefore, a trajectory interpolator generates a minimal traveling time cubic polynomial trajectory for both position and orientation of the end effector in the Cartesian space [65]. The orientation interpolation is based on a quaternion representation which, contrary to other notations such as the Euler angles, is numerically stable and free of singularities [89]. The resulting trajectory represents the set-point of a quaternion based feedback controller [40] that yields into linear and angular velocities in Cartesian space. These are then transformed to the joint space by using the SR inverse kinematics of chapter 0. The execution proceeds in a similar way as the one just explained for the PTP controller starting from the submission of the desired pose and finishing with the acknowledgment from the server when the aim is achieved (see Figure 4.15).

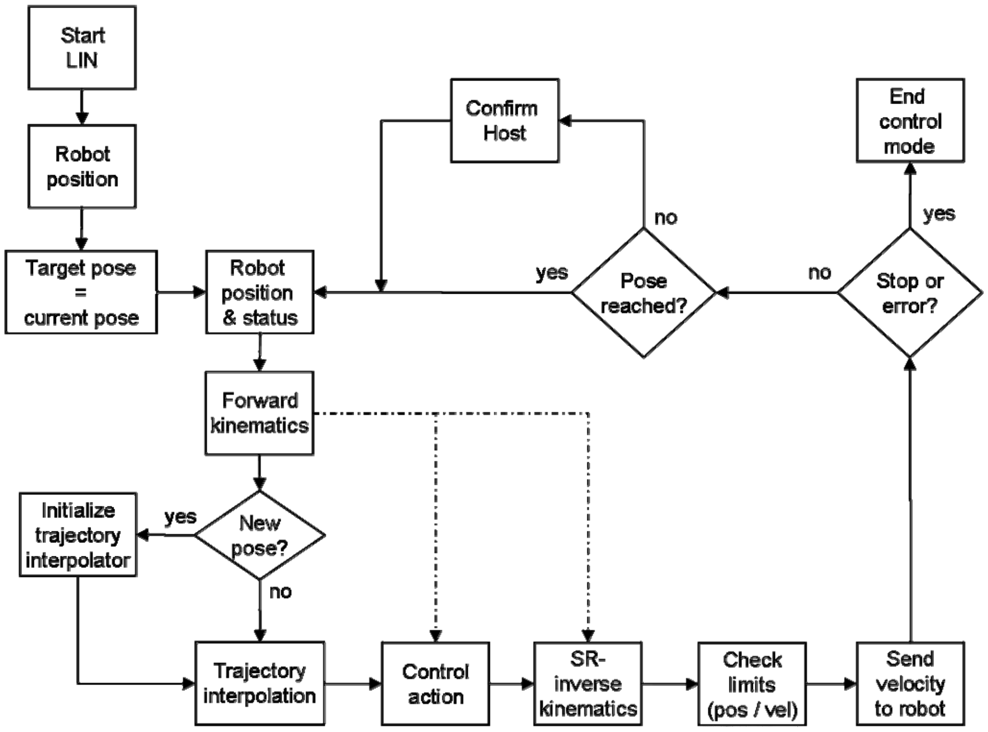


Figure 4.15. Flow diagram of linear trajectory Cartesian position controller (LIN)

4.5.5 Guidance virtual fixture controller

The guidance virtual fixture (GVF) controller is intended for cooperative tasks where the surgeon can move the robot directly with the hand. This control strategy is the first example of a task requiring more than one task-loop, i.e. the HPL-FTS and TCL-ROB task-loop. The former acquires the applied forces at the end-effector, processes them and finally passes them to the time critical loop. The applied forces/moments are converted to linear/angular velocities and then separated into two complementary subspaces of preferred and non-preferred directions depending of the virtual constraints previously defined by the user. If non constraints are specified, the robot can be freely moved along the 3D space; otherwise, the allowed movements will depend on the virtual definition. The resulting velocities are then transformed to the joint space and transmitted to the robot. The purpose of this section is only to present a general overview of the cooperative mode. A detailed explanation of the virtual constrained guidance controller using virtual fixtures and the singular robust strategies are presented in chapters 5 and 0, respectively.

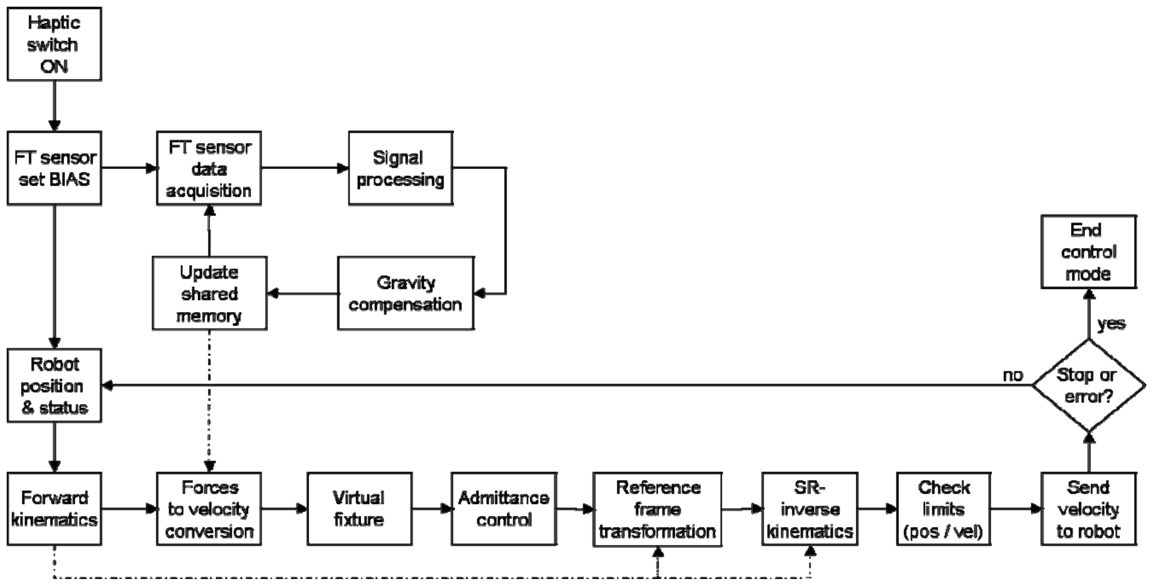


Figure 4.16. Flow diagram of Guidance Virtual Fixture controller (GVF)

4.5.6 Patient tracking controller

The patient tracking (TRK) controller automatically tracks possible movements of the patient and drives the robot to compensate them. It makes use of the navigation system task-loop (HPL-NAV) together with the **TRK** sub-state of the robot task-loop (TCL-ROB). This mode requires that RBs are mounted on both, robot arm and patient, and that the position and orientation of the RB related to the robot end-effector is well-known[5]. The position of the TCP with respect to the robot end-effector is also well known. All this information together with the measurements provided by the navigation system make possible to calculate required transformations that give a fixed relationship between patient and tool (see Figure 4.17), which is calculated only once when starting the tracking controller.

Then, the main objective of TRK controller is to keep this relationship constant at real-time during the surgical intervention. A patient movement causes a deviation which is automatically compensated. That makes the robot able to maintain the optimal tool position all the time during the operation. Although the patient tracking is a special

[5] This is computed by performing well defined calibration movements during the initialization process of the system [70].

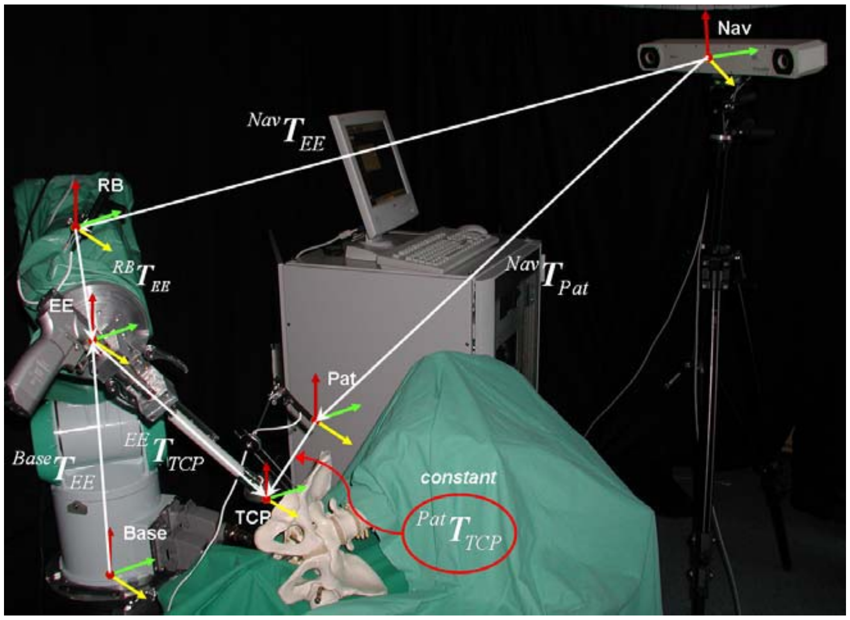


Figure 4.17. Coordinate systems used for tracking controller

feature of the modiCAS system, a deep insight in its development is considered beyond the scope of this work. Further related information can be consulted in [136], [70], [106]. Figure 4.18 shows the flow diagram of the TRK controller.

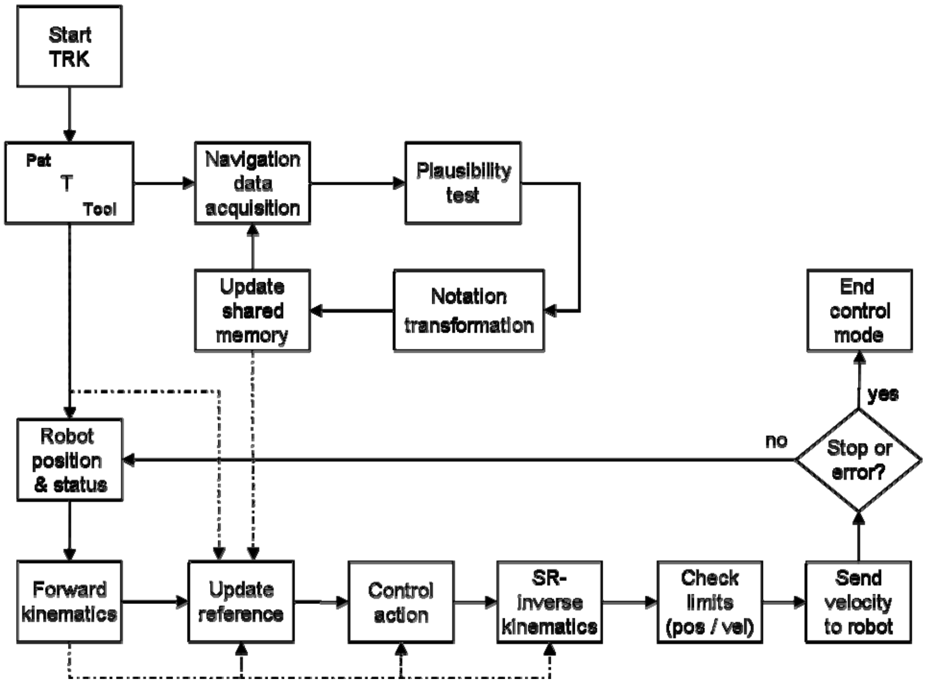


Figure 4.18. Flow diagram of patient tracking controller (TRK)

4.6 Host computer

The client skeleton is based on an event-based producer/consumer design [76], which allows creating efficient and flexible applications. An event is an asynchronous notification that something has occurred. Events can originate from the user interface, external I/O, or can be generated programmatically, e.g. acknowledgments coming from the server generate events signals that are handled in a similar way as the rest of events occurred within the client.

The host-target communication, i.e. the command interface, has already been presented in section 4.2. In this section, the command interface is contemplated only as a library of commands that can be used at any time to request services from the server. Feedback information coming from the server is stored directly in internal variables that are available to the whole client.

Besides the command interface, four modules are distinguished within the basic client skeleton, which work together to manage the whole execution at the client. These are described in the following sub-sections.

4.6.1 GUI producer

The GUI Producer is responsible for detecting any user request coming from the GUI, i.e. when a user changes the values of a control, moves or clicks the mouse or presses a key. Each of such actions produces a particular event. The GUI Producer wakes up when an event occurs and sleeps in between. This minimizes processor usage without sacrificing interactivity. When a GUI event occurs, this is identified and a new programmatically generated message event is directly produced to further notify the event consumer to handle the event. Notice that this construction allows having different sources to produce the same message event. This may be useful when the GUI allows multimodal interaction, for example, in cases where the same action can be generated by clicking on a tool bar button or through the menu bar or a running-time menu appearing after clicking on the right mouse button over a graphic (just to mention some of them).

4.6.2 External producer

The External Producer is responsible for detecting any user request coming from an external source, such as the modiCAS planning software. Notice, however, that any other source that complies with the communication protocol could produce events. The communication is based on TCP/IP protocol and uses XML language in order to provide a universal interface which is flexible enough for further expansion.

The External Producer opens a communication channel and sleeps until external program request connection. In a similar way as in the GUI-producer, the request can be either forwarded to the event consumer or directly to the target computer.

4.6.3 Event consumer

The Event Consumer is responsible for managing the application-tasks upon request. But it does not execute any task by itself; it rather delegates the work to independent application-tasks by dynamically starting and stopping them. The amount of application-tasks and their appearance depends on the application. These may have a GUI or not, depending on whether user interaction is required or not.

The event consumer may realize additional operations in order to keep a consistent behavior along the whole application. Some of these may imply sending direct commands to the target, actualization of the toolbar menu, requests to GUI-handler (see next section) among others. It is important to avoid long time executions inside the loop.

4.6.4 GUI handler.

The GUI Handler updates the appearance of the GUI every time it is remotely commanded by another loop. For example, let us suppose that the main window contains a section with controls to move the robot to a specific desired position. and that certain calibration procedure is started by the execution-loop. For safety reasons, the GUI-loop is requested to blind out this section during calibration so that no movement can be commanded. Figure 4.19 shows the basic modules contained in a client application. Additionally, the various application-tasks may run parallel to these modules. More details are presented in the next section.

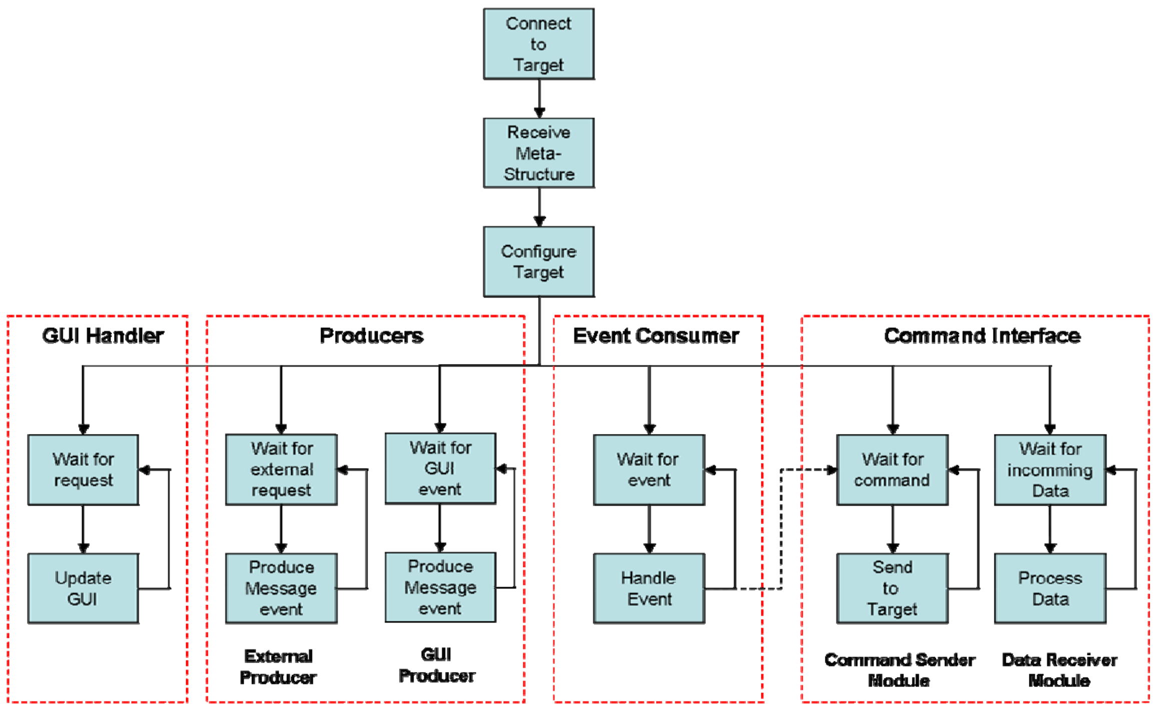


Figure 4.19. Host application

4.7 Modular distribution

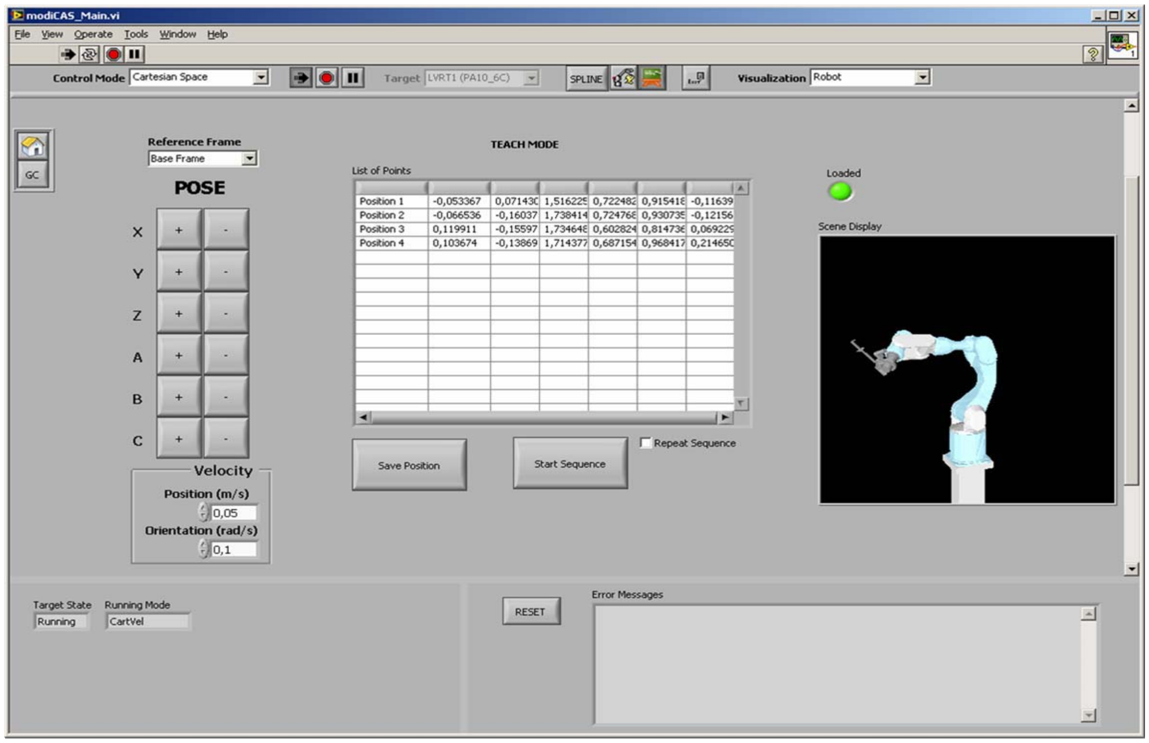
The main skeleton discussed in section 4.6 gives the possibility to manage different applications, but their execution actually occurs in separate loops. It is important, however, to have control over the number of tasks running at the same time. The design of an optimal ergonomic GUI for surgical applications is out of the scope of this work. Nevertheless, a first proposal is presented, which has been implemented to exemplify the usability of the client structure. Notice, however, that a deeper analysis of this issue is recommended for further design.

The GUI comprises components normally found in conventional software applications. It consists of one main window divided into four sub-sections (see Figure 4.20):

- *System Status Section*: The System Status Section is the only one with fixed elements that appear all the time giving feedback information about the status of the target computer and the system components. The other three sections are actually sub panels being able to contain different kinds of widgets which appearance depends on the running application task. Each sub-panel has a task execution module used by the event consumer to load and unload tasks on each of them. More details about execution modules are given in section 4.8. Which task runs in which sub-panel depends on the purpose of the task.

- *General Purpose Section*: The General Purpose Section is a sub-panel containing widgets that directly get access to general functions, such like **PTP**, **LIN**, **CVEL**, and **JVEL**. There exists one widget per functionality, each of which has its own GUI with a particular appearance adjusted to cover the specific demands. All these are dynamically interchangeable.
- *Application Section*: The Application Section is a sub-panel containing widgets with the main task, the contest of which depends of the application. As illustrative example, Figure 4.20 shows the Teach Mode application, where a list of multiple positions of the robot can be managed (saved, loaded, deleted, etc.), and actual execution of the sequence of positions can be commanded to the robot.
- *Visualization Section*: The system feedback visualization is contained in this section. Any kind of visualization can be shown here, whether it is the robot's position in joint space, or Cartesian space, in the form of graphics or 3D-representation, all kind of visualization is executed within the Visualization sub-panel. Just to give an example, a widget can contain a 3D-representation of the robot arm together with the position of RB detected by the navigation system that can be updated on-line with the feedback information coming from the target (see Figure 4.20).

The management of the whole sub-panels is possible through both, the menu bar and a toolbar.



Control Mode: Cartesian Space
Target: LVRT1 (PA10_6C)
Visualization: Robot
Reference Frame: Base Frame

Velocity
Position (m/s): 0,05
Orientation (rad/s): 0,1

List of Points

Position 1	-0,053367	0,071430	1,516225	0,722482	0,915418	-0,11639
Position 2	-0,066536	-0,16037	1,738414	0,724768	0,930735	-0,12156
Position 3	0,119911	-0,15597	1,734648	0,602824	0,814736	0,069229
Position 4	0,103674	-0,13869	1,714377	0,687154	0,968417	0,214650

Target State: Running
Running Mode: CartVel

Figure 4.20. Graphical User Interface of the client running at host computer

4.8 Execution and task modules

An execution module is the interface used by the event consumer to dynamically starting and terminating application tasks. There exist two types of execution modules depending on the type of task they manage:

- *Independent tasks*: These tasks run either in an independent window or they do not have GUI at all. The execution module can run multiple tasks simultaneously, executing them in parallel loops. It keeps a reference to every launched task so it can be terminated properly.
- *Sub-panel tasks*: These tasks have a GUI executing within one of the sub-panels previously presented. Each sub-panel has its own execution module. Only one task per sub-panel is able to run at any time. This means, when a new task is launched inside a specific sub-panel, any task running inside is first terminated.

All tasks are designed in an individual widget as a separate task module independent of the execution module. The GUI contained inside each task module has to be locally controlled. This means that the main GUI Handler has only influence on the controls and indicators belonging to the main window. It can change properties of the sub-panel as a whole, but not of each control and indicator contained inside the sub-panel. Notice that even the hidden task modules are contained in a widget, but they are never shown to the user.

5. Human-robot cooperation

5.1 Hands-on interface

The Hands-on Interface is activated only when the surgeon presses one of the two switches available in the ergonomic handle mounted at the tip of the robot. This is especially designed to provide easy access to any of both hands. The force applied to the grab is measured by the FT-sensor mounted just behind the holding mechanism. Figure 5.1 shows the whole end effector mechanism comprising tool holder, handler and force-torque-sensor.

The specific tool is mounted just after the handle mechanisms, therefore it will also influence on the applied forces measured by the FT-sensor. This load depends of the orientation of the tip due to gravitational forces. Then, erroneous values would be acquired once the orientation of the tool is changed. Hence, online gravity compensation must be done during the cooperative mode in order to obtain only the forces applied by the surgeon. This is achieved by doing an off-line calibration for each tool to determine its dead load parameters (mass and centre of mass). This has to be done only once for each tool, and then it is automatically loaded each time the tool is changed. Both off-line calibration and on-line compensation are based on J. Heindl approach [48].

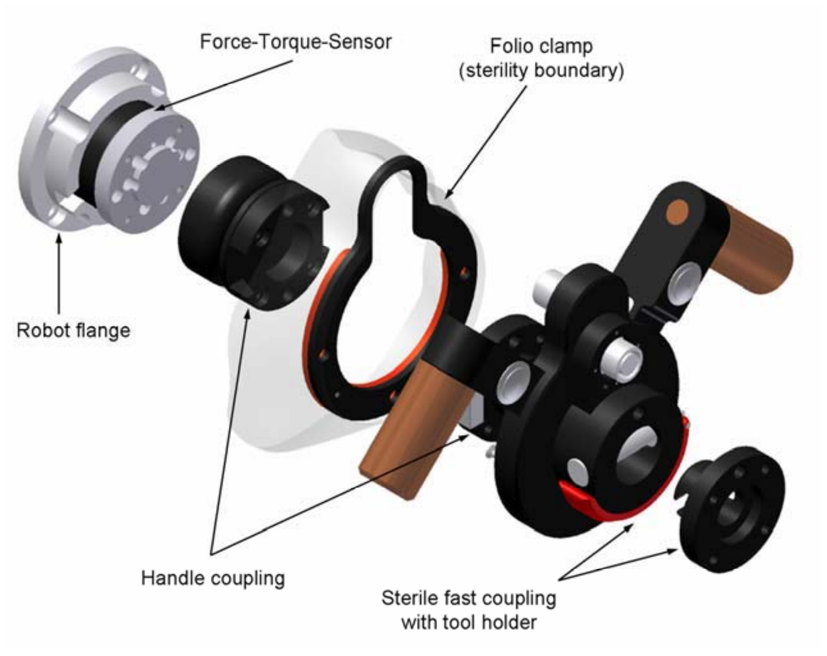


Figure 5.1. Handle system with rapid tool-exchange mechanism.

At software level, the High Priority Loop for the data acquisition of the Force/Torque Sensor (HPL-FTS), presented in section 4.3.3, takes care of the data acquisition and processing of the signals coming from the F/T sensor. The data processing steps comprise: voltage-to-force transformation, mean value calculation, drift compensation, filtering, and gravity compensation. Each calculation is performed sequentially one after the other (see Figure 5.2).

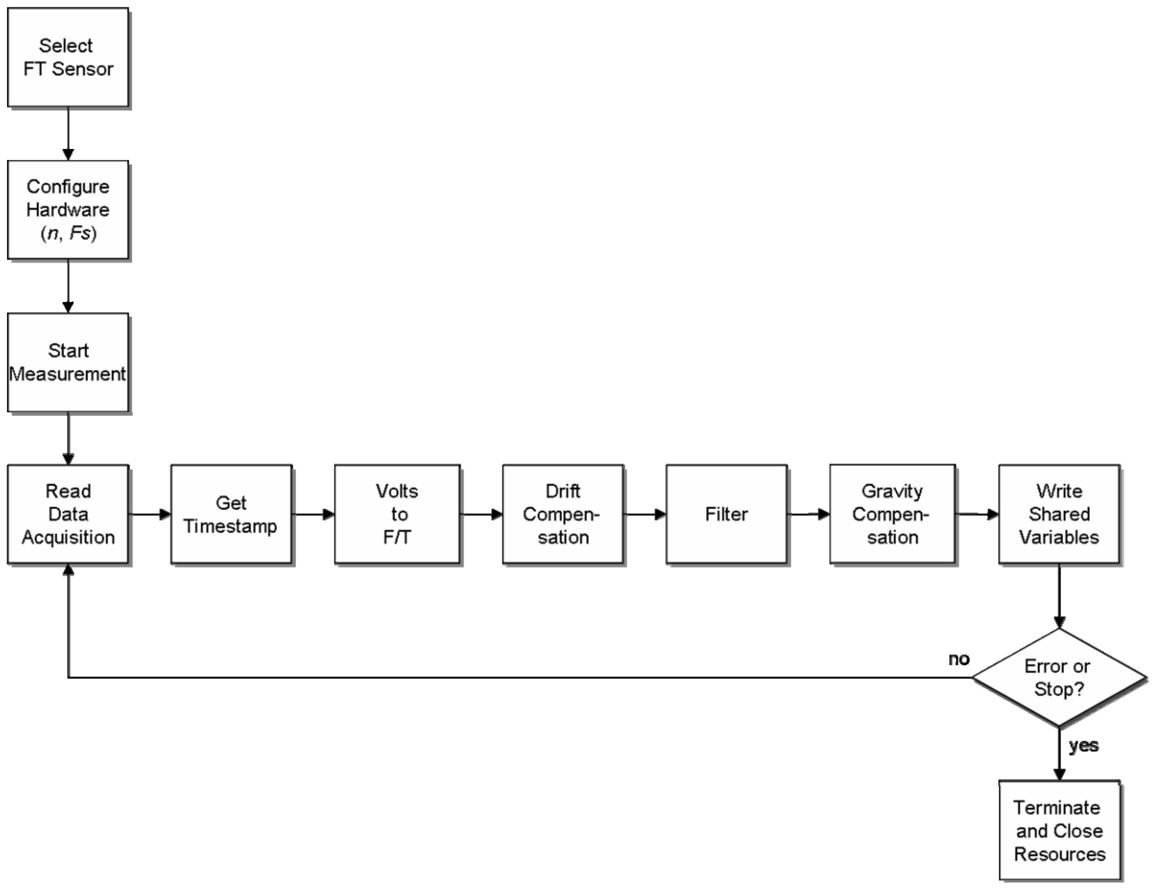


Figure 5.2 Flowchart of High Priority Loop for Force Torque Sensor data acquisition and processing (HPL-FTS)

5.2 Virtual fixtures description

Virtual Fixtures (VF) are essentially the separation of the 3D working space into two complementary subspaces, one containing all the preferred directions, and the other containing the non-preferred ones. A VF can be composed of one or more directions, the combination of which permits different anisotropic movements. Each of these directions is hereby defined as single *virtual unit*.

Let us distinguish between two types of virtual units, namely the linear virtual unit **l** and the angular virtual unit **α**. The former is a vector in $\Re^3$ that defines a specific direction in the Cartesian space along which the displacement of the robot's end-effector is permitted. The latter, also a vector in $\Re^3$, specifies an arbitrary axis in the Cartesian space, about which a rotation of the end-effector is possible. Now let us define a subspace U of $\Re^6$ containing all preferred directions for both translation and rotation. Let $\boldsymbol{S}^l$ and $\boldsymbol{S}^\alpha$ be two subsets of $\Re^3$ comprising the linear independent set of vectors that span U for position and orientation respectively:

$$\begin{aligned}\boldsymbol{S}^l &= \{\mathbf{l}_1, ..., \mathbf{l}_p\},\\ \boldsymbol{S}^\alpha &= \{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_k\},\end{aligned} \tag{5.1}$$

where $p,k \leq 3$. Now, let $\boldsymbol{D}$ denotes the $6\times(p+k)$ instantaneous preferred direction matrix comprising the elements of S^l and S^α,

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{S}^l & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{3\times p} & \boldsymbol{S}^\alpha \end{bmatrix} = \begin{bmatrix} (\mathbf{l}_1 \mid ... \mid \mathbf{l}_p) & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{3\times p} & (\boldsymbol{\alpha}_1 \mid ... \mid \boldsymbol{\alpha}_k) \end{bmatrix} \tag{5.2}$$

such that:

$$\boldsymbol{P}_U = Ran(\boldsymbol{D}) = \boldsymbol{D}(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T \tag{5.3}$$

The orthogonal projection $\boldsymbol{P}_U$ acts as the identity of U, i.e. any vector **x** in this subspaces has $\boldsymbol{P}_U\mathbf{x} = \mathbf{x}$. The subspace U is the exact range of this projection. Furthermore, there exists an orthogonal complementary subspace V that contains all the non-preferred directions. Every vector **x** in V has $\boldsymbol{P}_U\mathbf{x} = 0$. This is the *null space* also called *kernel* of the projection. Its corresponding projection operator is given by

$$\boldsymbol{P}_V = Ker(\boldsymbol{D}) = \boldsymbol{I} - \boldsymbol{P}_U \tag{5.4}$$

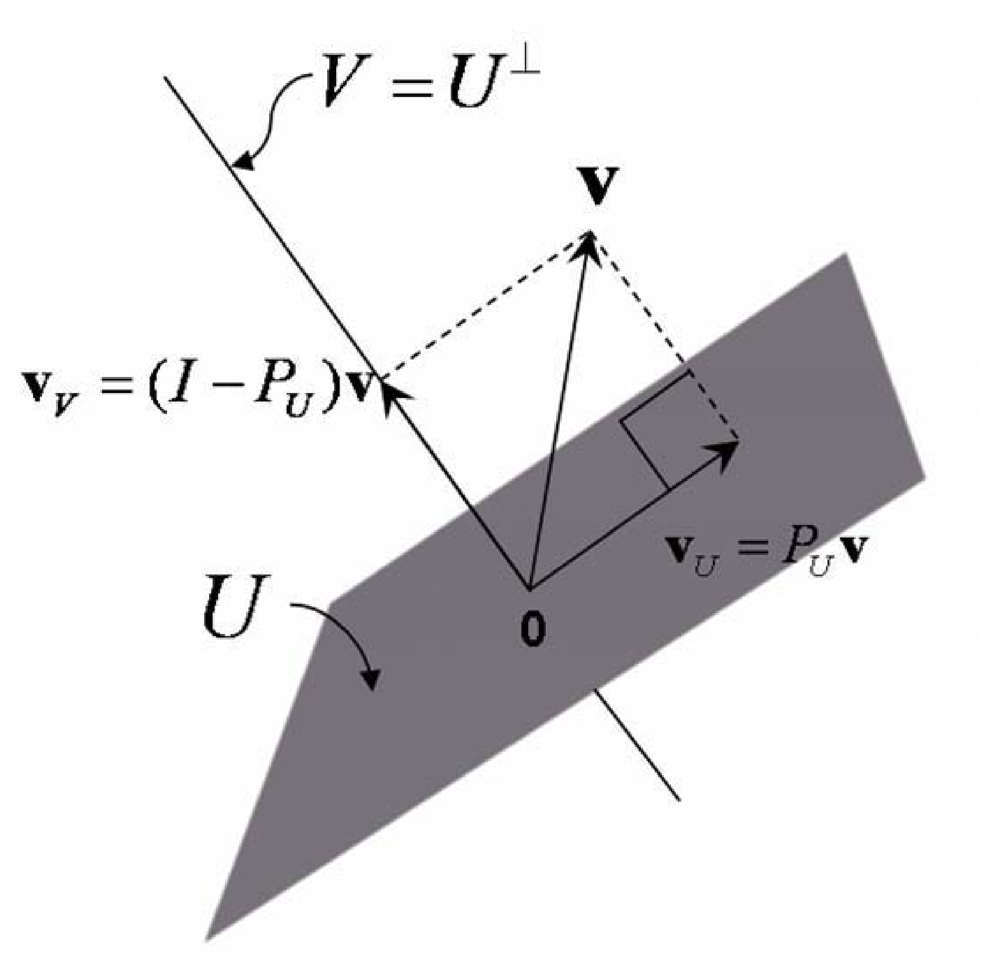

$V = U^{\perp}$
$\mathbf{v}$
$\mathbf{v}_V = (I - P_U)\mathbf{v}$
$\mathbf{v}_U = P_U\mathbf{v}$
U
$\mathbf{0}$

Figure 5.3. Projection onto the subspaces of preferred directions *U* and of non-preferred directions *V*

Operators (5.3) and (5.4) have the following properties [116]:

- symmetry: $Ran(\boldsymbol{D}) = Ran(\boldsymbol{D})^T$
- idempotence: $Ran(\boldsymbol{D}) = Ran(\boldsymbol{D})Ran(\boldsymbol{D})$
- scale invariance: $Ran(\boldsymbol{D}) = Ran(k\boldsymbol{D})$, where $k \neq 0$
- orthogonality: $Ker(\boldsymbol{D})^T Ran(\boldsymbol{D}) = 0$
- completeness: $rank\left(\alpha Ker(\boldsymbol{D}) + \beta Ran(\boldsymbol{D})\right) = n$, where $\boldsymbol{D}$ is $n \times m$ and $\alpha, \beta \neq 0$
- equivalence of projection: $Ran(Ker(\boldsymbol{D})f)f = Ker(\boldsymbol{D})f$

The above statements are also valid if *Ran*(***D***) and *Ker*(***D***) are exchanged. Also the following equivalence may be useful:

- $Ran(Ran(\boldsymbol{D})) = Ran(\boldsymbol{D})$
- $Ker(Ker(\boldsymbol{D})) = Ran(\boldsymbol{D})$
- $Ran(Ker(\boldsymbol{D})) = Ker(Ran(\boldsymbol{D})) = Ker(\boldsymbol{D})$

The resulting $\boldsymbol{P}_U$ and $\boldsymbol{P}_V$ create a mechanism which can be used within the system control law in order to determine whether the applied forces at the end-effector are pointing in a preferred direction or not. These measured forces are expressed directly in the robot's end-effector coordinate system. This means, the virtual units in ***D*** must be also defined with respect to this frame. Nevertheless, if the robot kinematic is well known, as well as the relationship of the different possible reference frames with respect to the robot base frame, it is then possible to define each virtual unit with respect to one of the different coordinate systems. This implies, however, that the calculation of ***D*** must be executed every cycle time.

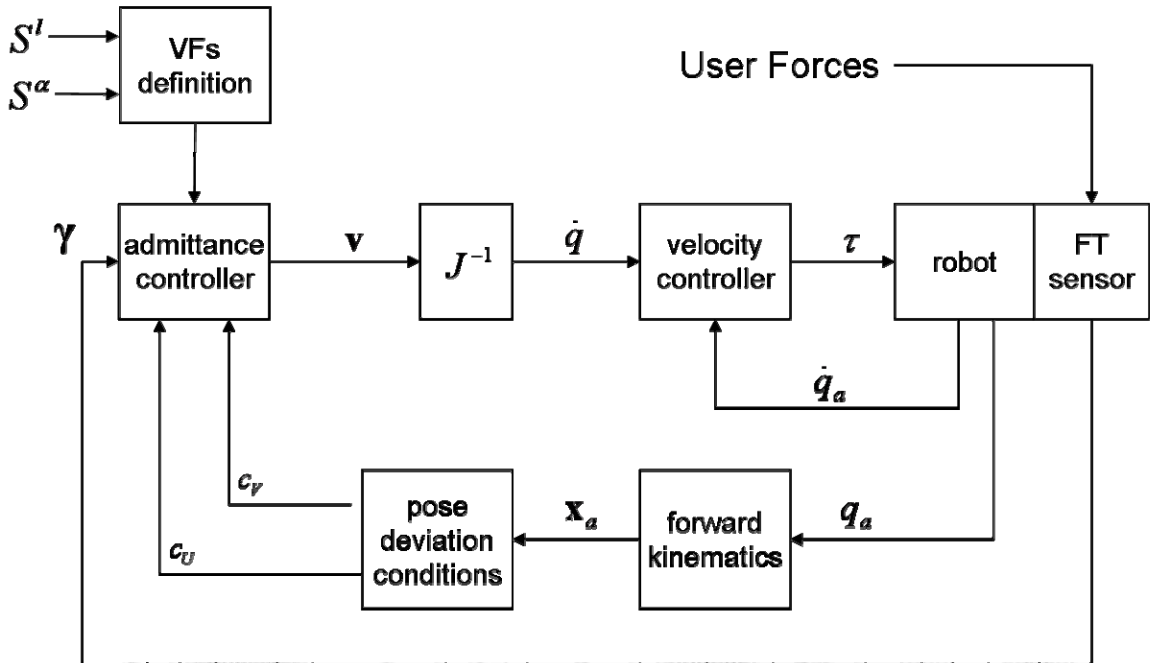


Figure 5.4. Control loop for cooperative robot system

5.3 Admittance controller

The control strategy for the cooperative mode essentially consists of two control loops: an inner velocity control loop at the joint level, and an outer admittance controller that modulates the end-effectors linear and angular velocities as a function of the applied forces. These velocities are then mapped to the joint space and further forwarded to the inner loop (see Figure 5.4). The general form of an admittance controller is:

$$\dot{\mathbf{x}} = c\boldsymbol{\gamma} \tag{5.5}$$

where $\dot{\mathbf{x}} = [\dot{\mathbf{p}}^T \quad \boldsymbol{\omega}^T]^T$ represents the linear and angular velocity of the end-effector. The scalar admittance gain $c \in [0,1]$ establishes the compliance level of the system. The vector $\boldsymbol{\gamma} = [\mathbf{f}^T \quad \boldsymbol{\tau}^T]^T$ contains the forces/moments applied at the end-effector.

In equation (5.5), the robot compliance has an isotropic behaviour, since the coefficient *c* affects all directions in the task space in the same way. The objective of the virtual fixtures is to provoke an anisotropic behaviour with different level of compliance on the preferred directions and the non-preferred ones. Therefore, the projection operators expressed in equations. (5.3) and (5.4) are incorporated into equation (5.5) together with an additional coefficients $c_U, c_V \in [0,1]$ leading into the following expression:

$$\dot{\mathbf{x}} = c_U(\boldsymbol{P}_U\boldsymbol{\gamma} + c_V\boldsymbol{P}_V\boldsymbol{\gamma}) = c_U(\boldsymbol{P}_U + c_V\boldsymbol{P}_V)\boldsymbol{\gamma} \tag{5.6}$$

The coefficient c_V regulates the amount of compliance on *V*. The resulting effect is a *guidance virtual fixture* that helps the user to move the end-effector along a desired path or surface defined by *U*. Different values of c_V will influence the level of guidance. If $c_V = 0$, the subspace *V* is completely eliminated, i.e. a hard guidance level along *U* is present. At the other extreme, with $c_V = 1$, there is no distinction between preferred and non-preferred directions, i.e. no guidance at all is present. Values in between will create the effect of soft guidance. The global compliance of the system against applied force can be regulated by means of c_U. This is useful when defining boundaries along preferred directions.

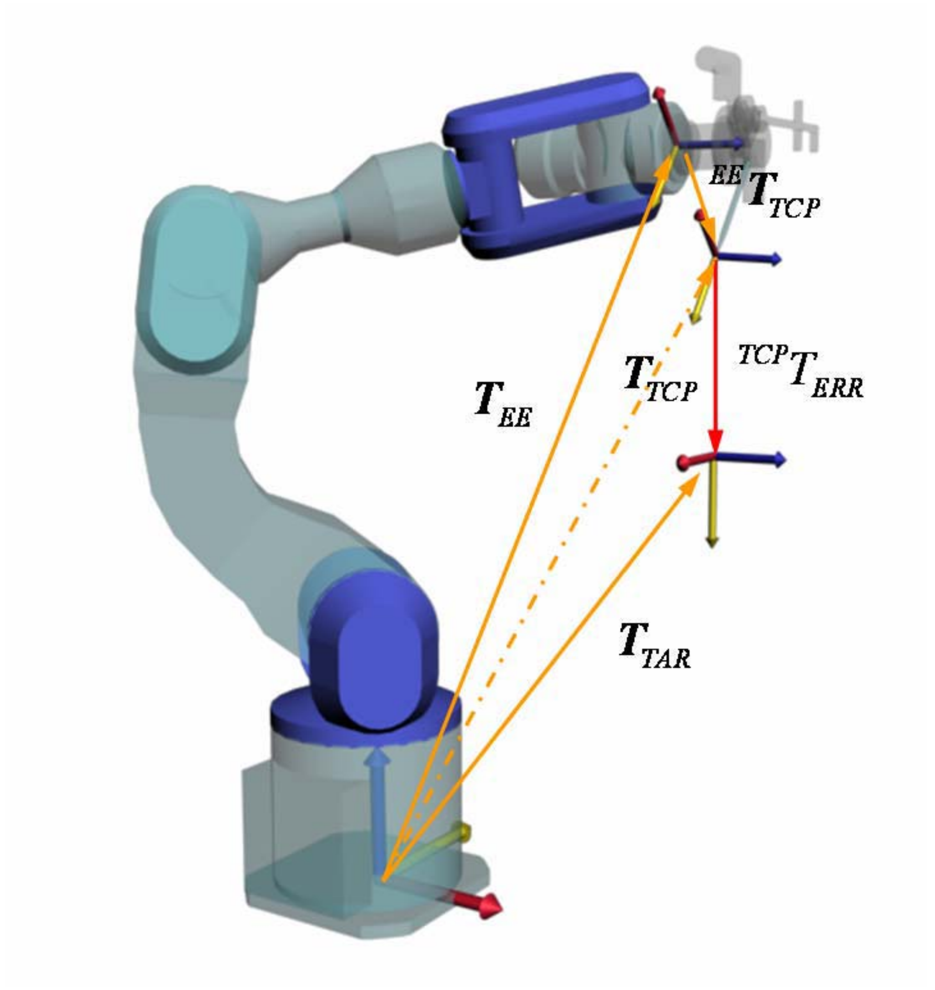


Figure 5.5. System reference frames

The compliance coefficients (c_U, c_V) are scalars that affect all the Cartesian components of the end-effector in the same manner. If these coefficients are substituted by a matrix form ($\boldsymbol{C}_U, \boldsymbol{C}_V$), where each matrix is a 6×6 diagonal matrix, then the different Cartesian components can be separately controlled.

$$\boldsymbol{C} = \begin{bmatrix} c_x & 0 & 0 & 0 & 0 & 0 \\ 0 & c_y & 0 & 0 & 0 & 0 \\ 0 & 0 & c_z & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & c_\varphi \end{bmatrix} \tag{5.7}$$

Where c_x, c_y, c_z and $c_\alpha, c_\beta, c_\varphi$ are position and orientation components, respectively. This is useful, for instance, when the compliance behaviour of the translational and rotational components have to be controlled completely independent from each other.

5.4 Deviation error

Expression (5.6) allows the user to move the end-effector in preferred directions despite its actual position and orientation. However, there is normally a desired reference pose (position and orientation) to which the virtual fixture is referred. Any deviation of the end-effector from the reference along the non-preferred directions is considered an error, and it has to be compensated. This error compensation is regarded as a reaching target problem acting on *V*. The reference target pose defines the position and orientation that is to be reached and maintained. This pose is also the center reference point of virtual fixture definition. Then, if the target pose varies upon the time, the virtual fixture moves with it.

Let $\mathbf{e}_p$ and $\mathbf{e}_r$ be the position- and orientation-error vectors, respectively. These vectors quantify the deviation of the actual Tool Centre Point (TCP) pose ${}^{Base}\boldsymbol{T}_{TCP}$ from the desired target pose ${}^{Base}\boldsymbol{T}_{TAR}$[6]. Each of both homogenous transformations having the form:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R}_{3\times3} & \mathbf{p} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \tag{5.8}$$

where $\boldsymbol{R}$ is the rotation matrix and $\mathbf{p} = (x \quad y \quad z)^T$ is the position vector. It is assumed that target pose $\boldsymbol{T}_{TAR}$ is already defined with respect to the base frame, and using the robot kinematics the $\boldsymbol{T}_{TCP}$ can be straightforward calculated as follows (see Figure 5.5):

$$\boldsymbol{T}_{TCP} = \boldsymbol{T}_{EE}\, {}^{EE}\boldsymbol{T}_{TCP} \tag{5.9}$$

[6] In the remaining of this document, for the sake of notation simplicity, when the reference frame is the base of the robot, the upper prefix of the transformation is omitted, e.g. $\boldsymbol{T}_{EE} = {}^{Base}\boldsymbol{T}_{EE}$. The same applies for rotation matrices $\boldsymbol{R}$ and position vectors $\mathbf{p}$.

where ${}^{EE}\boldsymbol{T}_{TCP}$ is the constant homogeneous transformation matrix from the TCP to the robot's end-effector, $\boldsymbol{T}_{EE}$ defines the end-effector with respect to the robot base frame and is calculated using the forward kinematics relationship of the robot arm.

The translational-error $\mathbf{e}_p$ is calculated by subtracting the translational vector of the homogenous transformations $\boldsymbol{T}_{TCP}$ and $\boldsymbol{T}_{TAR}$ as follows:

$$\mathbf{e}_p = \mathbf{p}_{TAR} - \mathbf{p}_{TCP} \tag{5.10}$$

In the case of orientation-error, $\mathbf{e}_r$ represents the axis of rotation in which the error is to be compensated and its norm represents the angle of rotation. This can be calculated using quaternions theory (see Appendix A). The error in terms of rotation matrices is defined as:

$$\boldsymbol{R}_{ERR} = \boldsymbol{R}_{TAR}^{-1}\boldsymbol{R}_{TCP} = \boldsymbol{R}_{TAR}^{T}\boldsymbol{R}_{TCP} \tag{5.11}$$

Applying quaternion representation, the orientation error can be expressed as:

$$\begin{aligned}\phi_{ERR} &= \bar{\phi}_{TAR}\phi_{TCP} \\ &= \begin{bmatrix} \eta_{TAR} & \boldsymbol{\varepsilon}_{TAR}^{T} \\ -\boldsymbol{\varepsilon}_{TAR} & \eta_r \boldsymbol{I}_{3\times 3} - \boldsymbol{S}(\boldsymbol{\varepsilon}_{TAR}) \end{bmatrix} \begin{bmatrix} \eta_{TCP} \\ \boldsymbol{\varepsilon}_{TCP} \end{bmatrix}\end{aligned} \tag{5.12}$$

Since (5.12) is expressed with unit quaternions, an axis of rotation $\boldsymbol{\varepsilon}'$ and angle of rotation φ can be derived by using the expression $\phi = \begin{bmatrix}\cos\varphi & \boldsymbol{\varepsilon}'\sin\varphi\end{bmatrix}^T$. Finally the orientation error $\mathbf{e}_\mathrm{r}$ is calculated as follows:

$$\mathbf{e}_r = \varphi\boldsymbol{\varepsilon}' \tag{5.13}$$

The orientation error $\mathbf{e}_\mathrm{r}$ indicates the axis of rotation, and $\|\mathbf{e}_r\|$ describes the magnitude of the rotation about this axis.

Any deviation from $\boldsymbol{T}_{TAR}$ within the subspace U is not considered an error, since it occurs along a preferred direction. However, deviations along V do represent an error. Thus, only the error along non-preferred direction is calculated using the projection operator P_V.

$$\mathbf{e}_V = \boldsymbol{P}_V \begin{bmatrix} \mathbf{e}_p \\ \mathbf{e}_r \end{bmatrix} = \begin{bmatrix} \mathbf{e}_{Vp} \\ \mathbf{e}_{Vr} \end{bmatrix} \tag{5.14}$$

5.5 Boundary conditions

Any deviation along non-preferred direction is considered an error; on the contrary, deviations along preferred directions are permitted. However, it might be the case that a specific limit must not be crossed over. Thus, boundary conditions are defined in a similar manner as in equation (5.14), but now considering only the projection on the preferred subspace *U*.

$$\boldsymbol{\delta}_U = \boldsymbol{P}_U \begin{bmatrix} \mathbf{e}_p \\ \mathbf{e}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_{Up} \\ \boldsymbol{\delta}_{Ur} \end{bmatrix}, \tag{5.15}$$

Expression (5.15) gives a measure of how far the TCP is from the target pose $\boldsymbol{T}_{TAR}$ inside the subspace *U*. This value is considered a distance measurement rather than an error. Thus, the value of c_U in equation (5.6) can be subjected to some boundary conditions based on the vector $\boldsymbol{\delta}_U$:

$$\tilde{c}_U(\boldsymbol{\delta}_U) = \begin{cases} c_U, & \boldsymbol{\gamma} \cdot \boldsymbol{\delta}_U > 0 \\ \dfrac{\| \boldsymbol{\delta}_U \|}{\mu_U}, & r_U - \| \boldsymbol{\delta}_U \| \geq \mu_U \\ 0, & \text{otherwise} \end{cases} \tag{5.16}$$

where r_U is the boundary magnitude along preferred directions, and μ_U is a threshold so that $0 < \mu_U < r_U$ defines a transition region that smoothes the compliance response of the robot toward the boundary.

A similar treatment can be given to the compliance coefficient c_V of the admittance control law expressed in equation (5.6) to control the compliance behaviour of the TCP on the subspace of non-preferred directions. It has been previously explained in Section 5.3 how this coefficient can influence in the level of guidance of the TCP. In this Section, its usage is further extended to achieve the volumetric virtual fixture. Thus, independently of the virtual shape, a new coefficient $\tilde{c}_V(\mathbf{e}_{Vp})$ is conditioned by the error magnitude $\| \mathbf{e}_{Vp} \|$ as follows:

$$\tilde{c}_V(\mathbf{e}_{Vp}) = \begin{cases} c_V, & \|\mathbf{e}_{Vp}\| > r_V \\ c_V - \left[\dfrac{r_V - \|\mathbf{e}_{Vp}\|}{\mu_V}\right]^n (c_V - 1), & \begin{array}{l}(r_V - \|\mathbf{e}_{Vp}\| < \mu_V \le r_V) \\ \text{and} \quad (\boldsymbol{\gamma} \cdot \mathbf{e}_{Vp} > 0)\end{array} \\ 1, & \text{otherwise} \end{cases} \tag{5.17}$$

where r_V is the boundary along non-preferred directions, and μ_V is a threshold so that $0 < \mu_V < r_V$ defines a transition region that smoothes the compliance response of the robot toward the boundary. Basically, if the TCP is located outside the virtual shape ($\|\mathbf{e}_{Vp}\| > r_V$), than $\tilde{c}_V(\mathbf{e}_{Vp}) = c_V$, which is not necessarily equal to zero. This means that the guidance level can still be regulated for the region outside of the volumetric virtual fixture. Inside the virtual shape, the TCP can be freely moved, except when a motion directed outward while been in the transition region. In such case, the compliance is gradually reduced until the boundary is reached, where the first condition applies. Finally, equation (5.6) is rewritten as:

$$\dot{\mathbf{x}} = \tilde{c}_U(\boldsymbol{P}_U + \tilde{c}_V \boldsymbol{P}_V)\boldsymbol{\gamma} \tag{5.18}$$

5.6 Manual error compensation

The manual compensation is based on the previous work by Bettini *et al.* [13], it relies on the input forces applied by the user to compensate for the deviation errors. Basically, in the presence of an error the virtual preferred directions are redefined to consider such error, creating a new virtual fixture that make possible to compensate for the error. Thus, a new instantaneous preferred direction $\boldsymbol{D}_e$ is defined, which considers the directions required to compensate any translational and rotational deviation from *U*:

$$\boldsymbol{D}_e = (1 - k_d)\boldsymbol{P}_U\boldsymbol{\gamma} + k_d\|\boldsymbol{\gamma}\|\mathbf{e}_V, \qquad 0 < k_d < 1 \tag{5.19}$$

The combination of the applied foces $\boldsymbol{\gamma}$ pointing at preferred direction (obtained by means of the projection operator $\boldsymbol{P}_U$) and the error vector $\mathbf{e}_V$ yields into a virtual direction that returns the TCP to the subspace *U*. The constant k_d regulates how strong is the influence of the error vector $\mathbf{e}_V$ in the new virtual preferred direction, i.e. how quickly the error is compensated. When the TCP lies within the subspace U, the second term of equation (5.19) vanishes. Now, using the new preferred direction $\boldsymbol{D}_e$ to recalculate the projection operators (5.3) and (5.4) and introducing them in the control law of equation (5.6) results in a control law equivalent to a pure subspace motion constraint [60]:

$$\dot{\mathbf{x}} = \tilde{c}_U (\boldsymbol{P}_{Ue} + \tilde{c}_V \boldsymbol{P}_{Ve}) \boldsymbol{\gamma}, \tag{5.20}$$

where

$$\begin{aligned} \boldsymbol{P}_{Ue} &= Ran(\boldsymbol{D}_e) \\ \boldsymbol{P}_{Ve} &= Ker(\boldsymbol{D}_e) \end{aligned},$$

Notice however, that the definition of the new preferred direction to compensate the error is not sufficient to guarantee that the error is minimised, i.e. the surgeon is still able to apply a force in the negative error direction, which would increase the error. Therefore, the applied forces pointing toward the negative error direction are filtered out by using the following condition:

$$\boldsymbol{\gamma}_e = \begin{cases} \boldsymbol{P}_U \boldsymbol{\gamma} & \mathbf{e}_V \bullet \boldsymbol{\gamma} < 0 \\ \boldsymbol{\gamma} & otherwise \end{cases} \tag{5.21}$$

Substituting $\boldsymbol{\gamma}$ with $\boldsymbol{\gamma}_e$ in equation (5.20) guaranties that only applied toward the error compensation are effective without affecting the forces pointing in the preferred directions.

5.7 Autonomous error compensation

The error compensation presented in equation (5.20) depends on the applied forces $\boldsymbol{\gamma}$. This means, the error is compensated for only if the user applies a force in the $\mathbf{e}_V$ direction; otherwise, the error remains present. For some cases, this compensation occurs intuitively, for example, in translation movements along a predefined direction, the user automatically corrects any possible error by pushing in the path direction. However, there may be cases where non compensation is induced at all, such as when making pivot rotations about the TCP at a constant target position, ideally the position remains fixed, but in reality slight deviations in the position occurs. Although the error is detected by the system and the $\boldsymbol{D}_e$ is defined, the compensation takes place only after the proper force is applied, though the act of rotating demands rather applied moments than forces. Consequently, the error remains present and even increases before the user can observe it and apply a compensation force.

Nevertheless, by adding one term to equation into (5.18), an automatic compensation of the deviation error can be achieved independently of the applied forces without affecting the virtual fixtures. The new expression looks as follows:

$$\dot{\mathbf{x}} = \tilde{c}_U(\boldsymbol{P}_U + \tilde{c}_V \boldsymbol{P}_V)\boldsymbol{\gamma} + k_V \mathbf{e}_V \tag{5.22}$$

The error term in equation (5.22) does not depend on the input forces $\boldsymbol{\gamma}$ anymore. Notice that rather than combining the error vector with the virtual definition D as in manual compensation (equation (5.19)), it is compensated with a simple linear control law ($k_V\mathbf{e}_V$). The gain k_V modulates the rate of response of the compensation. With this approach, the surgeon has still complete control inside U, while the robot assures that the reference target pose is maintained.

5.8 Virtual fixtures classes

5.8.1 Reference target

Given the TCP Cartesian position, both position and orientations components are intended only to reach a target $\boldsymbol{T}_{TAR}$. Therefore, $\boldsymbol{S}^l = \boldsymbol{S}^\alpha = \{0\}$, i.e. only the error-related term in $\boldsymbol{D}_e$ in equation (5.19) is needed to reach the desired target pose. The behaviour of this virtual fixture is illustrated in Figure 5.6.

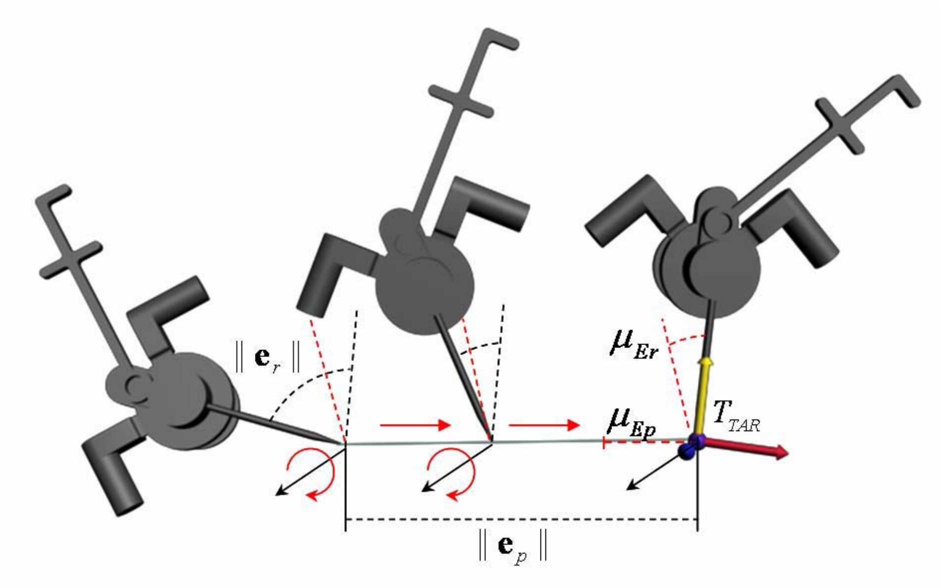


Figure 5.6. Reference target

Notice that if only either TCP position or orientation is intended to reach a desired goal during the execution of a cooperative task within a predefined virtual fixture, the only required condition is that the corresponding subset ($\boldsymbol{S}^l$ or $\boldsymbol{S}^\alpha$ for position or orientation, respectively) becomes empty subsets $\{0\}$. A transition region can be defined when $\|\mathbf{e}_r\|$ approaches zero in order to smooth the gain discontinuity when reaching the target. Then, $\tilde{c}_U$ is defined as:

$$\tilde{c}_U(\mathbf{e}_V) = \begin{cases} \dfrac{\|\mathbf{e}_V\|}{\mu_E}, & \|\mathbf{e}_V\| \leq \mu_E \\ 1 & \textit{otherwise} \end{cases} \tag{5.23}$$

Position and orientation can be independently controlled if, instead of $\tilde{c}_U$, the matrix notation of the compliance coefficient $\tilde{\boldsymbol{C}}_U$ (as explained in section 5.3) with different parameter values of conditions (5.23) for each component is used. For instance, two different transition region thresholds μ_{Ep} and μ_{Ev} can be separately defined for position and orientation components, respectively (see Figure 5.6). Once the target pose is reached, the condition (5.21) assures that this is maintained.

Note that the autonomous compensation cannot be applied to this purpose due to safety reasons. The main idea of autonomous compensation is to avoid leaving the preferred subspace rather than commanding the TCP to a specific target, which would mean that the robot executes large movements by its own.

5.8.2 Move along an axis

This virtual fixture class limits the TCP movements along a reference line in 3D space (see Figure 5.7). First, a virtual unit vector $\mathbf{l}_1 = \begin{bmatrix} l_x & l_y & l_z \end{bmatrix}^T$ is defined. This indicates the direction of the preferred axis with respect to the TCP coordinate system. If this is given with respect to other reference frame, a suitable transformation is required. Besides, the reference frame $\boldsymbol{T}_{TAR}$, defined with respect to the base frame, has to be specified, since it is here where $\mathbf{l}_1$ has its origin. The subset $\boldsymbol{S}^l = \{\mathbf{l}_1\}$ together with the admittance control law expressed either in equation (5.20) or (5.22) (for manual or autonomous compensation, respectively) yield into a constrained movement along a line.

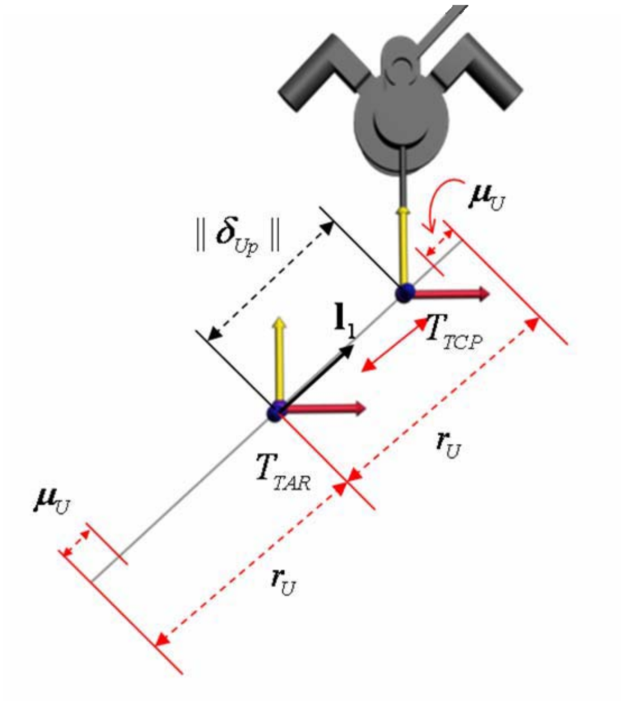


Figure 5.7. Move along a line

pointing at $\mathbf{l}_1$ and passing through the origin of $\boldsymbol{T}_{TAR}$. If the actual position of the TCP is off the path, the control law drives it back to the line. Concerning orientation, if $\boldsymbol{S}^{\alpha} = \{0\}$, then the orientation of the TCP reach the one defined by T_{TAR} and remains constant, otherwise, the possible rotations depend on its contents.

Starting from the origin defined by $\boldsymbol{T}_{TAR}$, the TCP can move along the axis $\mathbf{l}_1$ a distance of $\|\boldsymbol{\delta}_U\| \leq r_U$ in either positive or negative direction.

5.8.3 Rotate around an axis

In the *rotate around an axis* class, the TCP is forced reach to a predefined pose $\boldsymbol{T}_{TAR}$ and constrained to rotate only around the axis defined by a virtual unit vector $\boldsymbol{\alpha}_1$ while keeping its orientation perpendicular to the line (see Figure 5.8). In a similar way as in *move along a line*, the subset $\boldsymbol{S}^{\alpha} = \{\boldsymbol{\alpha}_1\}$ is defined. This virtual fixture can simultaneously constraint TCP movements together with a translational virtual fixture if $\boldsymbol{S}^{l} \neq \{0\}$. In such a case, both type of VFs would share a common target pose $\boldsymbol{T}_{TAR}$.

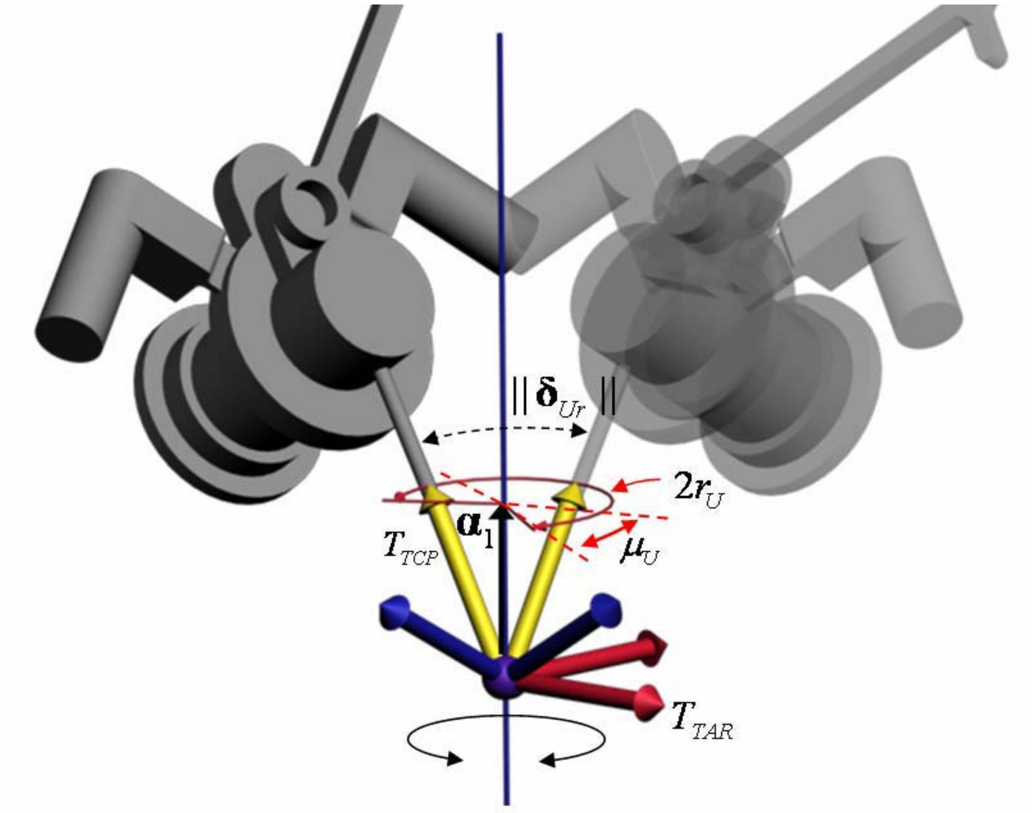


Figure 5.8. Rotate around one axis

5.8.4 Move along a plane

The *move along an axis* translational class can be easily extended to confine the TCP to move on a plane (see Figure 5.9). Let us define the subset $\boldsymbol{S}^l = \{\mathbf{l}_1, \mathbf{l}_2\}$, where $\mathbf{l}_1$ and $\mathbf{l}_2$ are two non-zero linear independent vectors. Then $\mathbf{P} = Ran(\boldsymbol{S}^l)$ is the plane passing through the origin and the vectors $\mathbf{l}_1$ and $\mathbf{l}_2$ in which the TCP is constrained. The reference pose $\boldsymbol{T}_{TAR}$ represents the origin of this plane. It is important to observe that the orientation of $\boldsymbol{T}_{TAR}$ is not directly related with the orientation of the plane but rather with the desired orientation of the TCP. Plane's orientation is given with respect to the TCP coordinate system, since it results from $\mathbf{l}_1$ and $\mathbf{l}_2$ which are expressed with respect to the TCP frame.

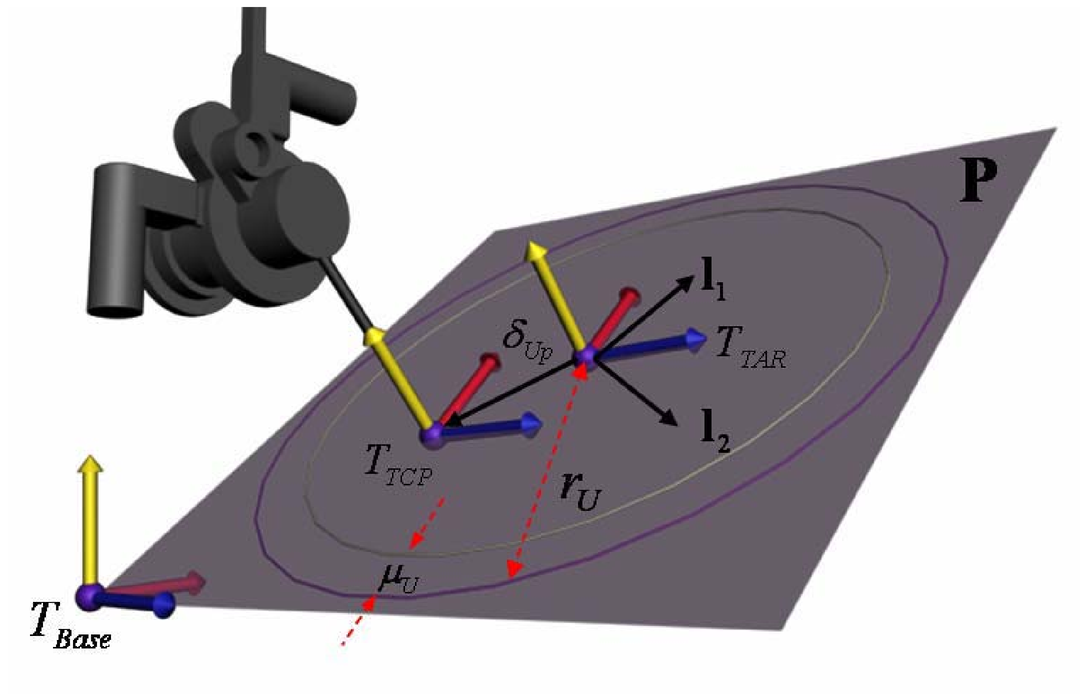


Figure 5.9. Extension to plane

The boundary conditions (5.16) will affect at any direction along the plane in the same manner. This means, under such conditions, the limits of the plane are defined by a circumference of radius r_U, which boundary cannot be crossed over. At distance $\|\boldsymbol{\delta}_U\| \geq r_U - \mu_U$ from the origin inside this circumference the robot compliance gradually decreases until the boundary is reached where the robot compliance becomes zero. Other kind of boundaries may demand different condition definitions, e.g. projection of **δ** onto the virtual unit vectors $\mathbf{l}_1$ and $\mathbf{l}_2$ would make possible to establish two condition parameters for the definition of independent boundaries at each direction, thus having square shaped limits instead of a circumference.

5.8.5 Rotate around two axes

One can define more than one axis of rotation simultaneously (see Figure 5.10). Notice however that, in the same way as in translation, three linear independent vectors $\boldsymbol{S}^{\alpha} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3\}$ are enough to span the whole $\Re^3$. In other words, a rotation in all directions would be then possible.

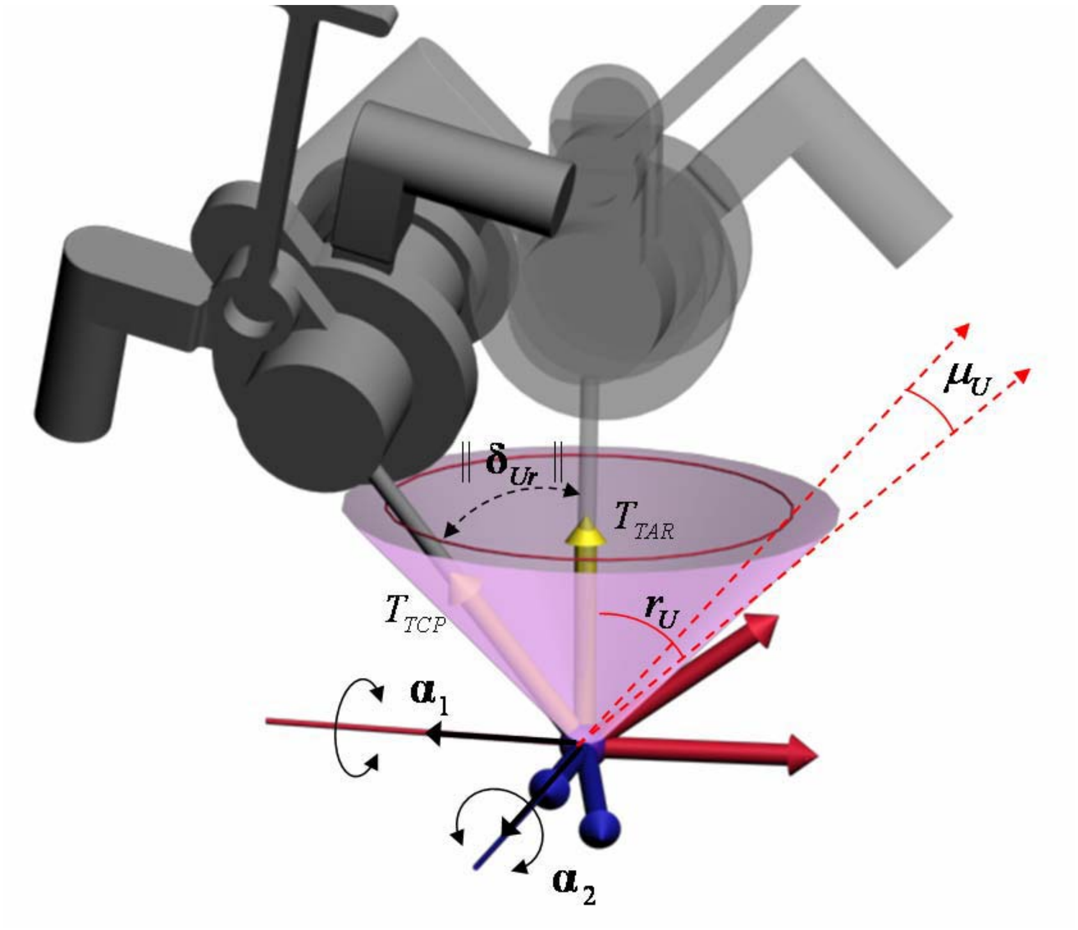


Figure 5.10. Rotate around two axes

5.8.6 Extension to volume

The virtual fixtures discussed until now constraint the motion into $\Re^1$ or $\Re^2$ basically hinge upon the definition subsets $\boldsymbol{S}^l$ and $\boldsymbol{S}^\alpha$. However, extension of constrained motions in $\Re^3$ using only the subspace of preferred directions becomes rather limiting. More flexibility is achieved if the non-preferred directions are also considered during the design of the different possible virtual fixtures. It has been pointed out in Section 5.5 that manipulation of $\tilde{c}_V$ makes possible to extend the concept to volumetric fixtures since it defines the compliance behaviour of the TCP on the subspace of non-preferred directions. The idea of a volumetric fixture is to confine the motion to a closed volumetric region without been able to leave it.

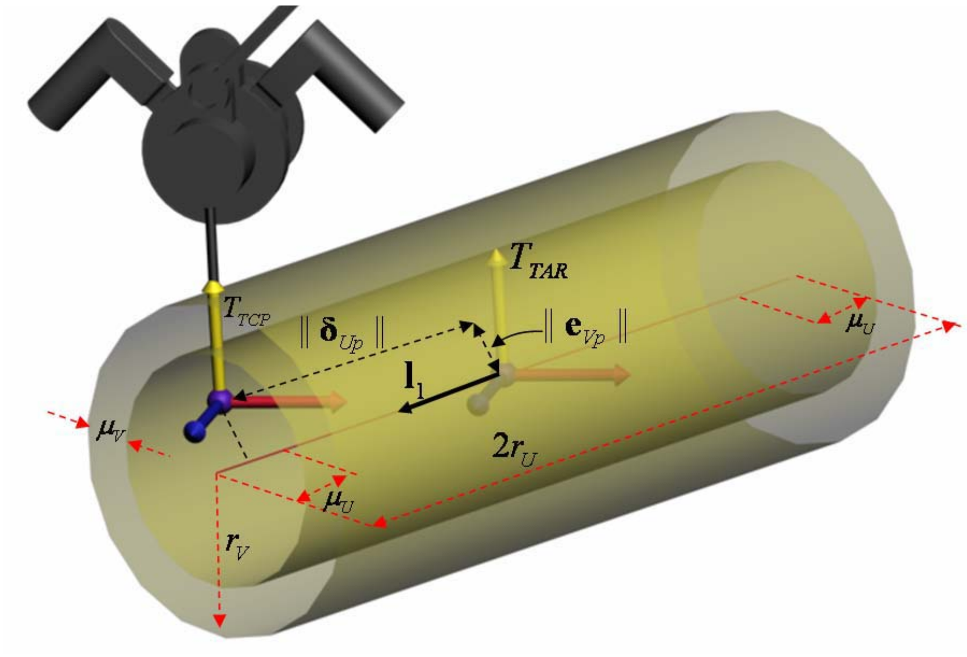


Figure 5.11. Virtual cylinder

The concept of volume fits very well for virtual fixture of the translational type, but not so for rotational ones. In the second case, one may wonder whether it make any sense at all to apply the concept or not. If the answer is no, then the compliance of the rotational components of the end effector can be controlled independently with a matrix notation, as stated in section 5.3. Hence, the first three elements of the diagonal matrix are subjected to conditions (5.17) while the last three elements may be subjected to a different conditioning criterion. Furthermore, the shape of the volumetric fixture strongly depends on the content of the translational subset $\boldsymbol{S}^l$. The geometrical interpretation is obtained in a natural way for each set of linear virtual vectors.

5.8.6.1 Virtual tube

The virtual cylinder is the direct extension of the virtual line explained in Section 5.8.2. Practically, the definition of $\boldsymbol{S}^l$ and $\boldsymbol{T}_{TAR}$ remains the same. But now, the conditions given in (5.17) are additionally considered for the compliance coefficient influencing the non-preferred directions. In this way, deviations in the directions perpendicular to the line smaller than r_V become also possible. Notice however that the magnitude of $\mathbf{e}_{Vp}$

does not increase for deviations from $\boldsymbol{T}_{TAR}$ along preferred directions. This gives finally the impression of having a virtual cylinder of diameter equal to $2r_V$ and length equal to $2r_U$. The thresholds for the transition region near the borders are defined with ε_V and ε_U for the cylinder periphery and circular bases, respectively. The virtual cylinder shaped fixture around a reference line in the 3D space is illustrated in Figure 5.11.

5.8.6.2 Virtual cone

The virtual cone is a special case of virtual fixture that permits to reach or leave a point in the 3D space while keeping the TCP inside a virtual cone. The direction of the cone axis is given by $\mathbf{l}_1$, and the cone's apex is taken from the reference position contained in $\boldsymbol{T}_{TAR}$. First, the subspace U along the line is calculated $\boldsymbol{S}^l = \{\mathbf{l}_1\}$ as usual. Then, deviations of the actual position from the target position $\boldsymbol{\delta}_{Up}$ and $\mathbf{e}_{Vp}$ are obtained for both preferred and non-preferred directions, respectively. A new boundary condition is then defined. Its value depends on in which size of the reference point the TCP is located. The virtual cone can only be projected at the positive side.

$$r_V = \begin{cases} \|\boldsymbol{\delta}_{Up}\| \tan(\phi), & \boldsymbol{\delta}_{Up} \cdot \mathbf{l}_1 \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.24}$$

where ϕ is the cone opening half-angle with respect to $\mathbf{l}_1$ (see Figure 5.12). The compliance conditions (5.17) are then applied. The subspace U just defined is useful for the calculation of the error components, but no so for driving the TCP. If this is introduced to the admittance controller, free movement inside the cone is successfully achieved, but at the boundaries discontinuity behaviour appears when trying to move the TCP along the periphery of the cone toward to its apex. Therefore, a new subspaces $\tilde{\boldsymbol{P}}_U$ and $\tilde{\boldsymbol{P}}_V$ using $\boldsymbol{S}^l = \{\mathbf{e}_p\}$ are calculated, which project onto a vector pointing always directly to the apex. This still provide free motion inside the cone but now permits a fluent motion along the periphery toward cone's apex. The control law for this special case is:

$$\dot{\mathbf{x}} = \tilde{c}_U (\tilde{\boldsymbol{P}}_U + \tilde{c}_V \tilde{\boldsymbol{P}}_V)\boldsymbol{\gamma} \tag{5.25}$$

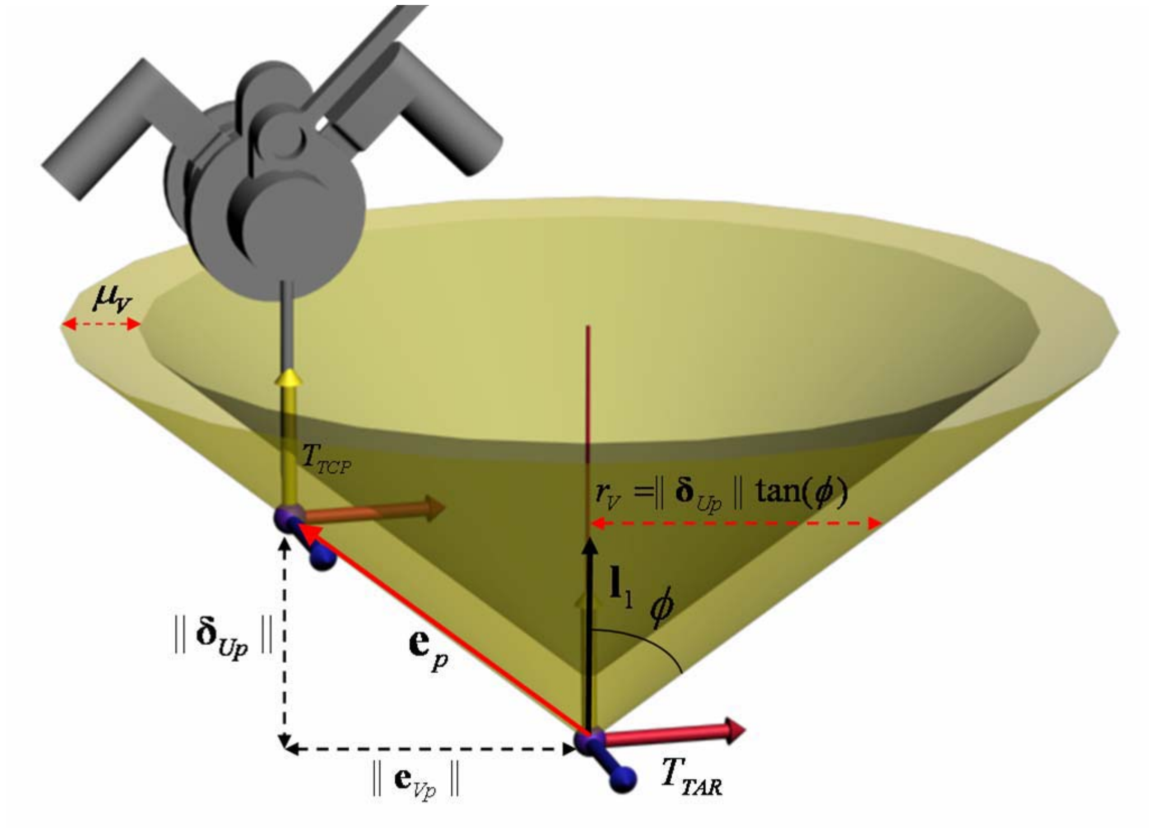


Figure 5.12 Virtual Cone

5.8.7 Reference trajectory

All virtual fixtures are defined with respect to a reference target $\boldsymbol{T}_{TAR}$. It has been shown that depending of the virtual definition, moving the TCP along or above one or more directions starting from this Cartesian reference pose is possible. Any deviation along non-preferred directions is considered an error and thus compensated. Now, if the target $\boldsymbol{T}_{TAR}$ varies along the 3D space, more complex virtual constrained trajectories can be generated. In this section, the reference target position $\mathbf{p}_{TAR}$ is described by a parametric function which permits to dynamically change the current target position as a function of the actual end-effector position. Therefore, a trajectory is generated using parametric spline functions together with arc-length parameterization (see Appendix B). A detailed description of the generation of such parametric functions is out of the scope of this work. Its main focus is the performance for the error minimization with respect to the virtual fixtures rather than the generation of the reference target. Nevertheless, the usage of such parametric functions expands the potentiality of the virtual fixture tremendously, since it make possible to define virtual constrained environments which are more complex than the ones presented in this work. A basic explanation of the working principle is given below. The reader interested in more detailed information is encouraged to consult [7], [8], [49], [138] and [139].

Let $\mathbf{p}(s)$ being a parametric spline curve describing the reference trajectory:

$$\mathbf{p}_{TAR}(s) = \left[x(s) \quad y(s) \quad z(s)\right]^T, \quad 0 \leq s \leq L, \tag{5.26}$$

where s denotes arc-length, L is the arc-length of the whole trajectory, and $x(s)$, $y(s)$, and $z(s)$ are Catmull-Rom spline functions with equidistant knots $\{s_0, s_1, ..., s_n\}$ with $s_0 = 0$ and $s_n = L$ (see Appendix B). If $\mathbf{p}_{EE} = [x_{TCP} \quad y_{TCP} \quad z_{TCP}]^T$ is the actual position of the TCP, and $\mathbf{p}_{TAR}(\hat{s})$ the closest point of trajectory to $\mathbf{p}_{EE}$ (Figure 5.13). Finding $\mathbf{p}_{TAR}(\hat{s})$ can be seen as an optimization problem [139]:

$$\xi(\hat{s}) = \min_{s \in [0,L]}(\xi(s)), \tag{5.27}$$

where

$$\xi(s) = (x(s) - x_{TCP})^2 + (y(s) - y_{TCP})^2 + (z(s) - z_{TCP})^2. \tag{5.28}$$

The quadratic minimization method is used to this aim (see Appendix C).

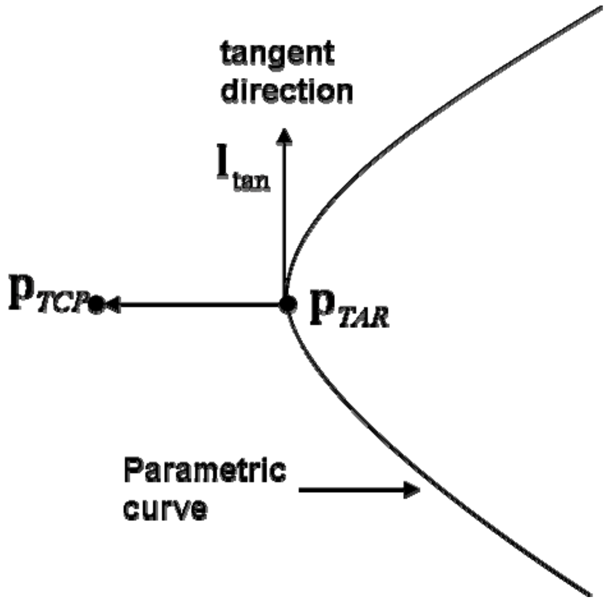


Figure 5.13 Closest point of spline curve to $\mathbf{p}_{TCP}$ and its tangent vector.

Movements of the TCP along $\mathbf{p}_{TAR}(s)$ are possible by defining the normalized tangent direction $\mathbf{l}_{tan}$ of the trajectory at the actual reference position $\mathbf{p}_{TAR}(\hat{s})$ as the virtual preferred direction, i.e. $\boldsymbol{S}^{l} = \{\mathbf{l}_{tan}\}$. The tangent direction $\mathbf{p}'_{TAR}(\hat{s})$ can be easily obtained once $\mathbf{p}_{TAR}(\hat{s})$ is known (see Appendix B), so finally:

$$\mathbf{l}_{tan} = \frac{\mathbf{p}'_{TAR}(\hat{s})}{\left\|\mathbf{p}'_{TAR}(\hat{s})\right\|} \tag{5.29}$$

The difference between actual position of TCP $\mathbf{p}_{TCP}$ and $\mathbf{p}_{TAR}(\hat{s})$ is considered a Cartesian position error:

$$\mathbf{e}_{\mathbf{p}}(\mathbf{p}_{TCP}) = \mathbf{p}_{TAR}(\hat{s}) - \mathbf{p}_{TCP} \tag{5.30}$$

5.9 Experimental evaluation

The experimental evaluation of the virtual fixtures presented below was tested using the modiCAS system. All tests were executed with the 6 degrees-of-freedom (DOF) PA10-6C robot arm, form Mitsubishi, Japan, and the mini45 force-torque sensor from ATI Industrial Automation, USA, mounted on the robot's end-effector. The inner velocity control loop is included in the servo driver of the robot system and runs with a frequency 1538 Hz. The outer admittance control loop, running at 200 Hz, was implemented on the RT-target running the LabVIEW-RT module. Position and orientation errors are calculated independently as follows:

$$\begin{aligned} \left\|\mathbf{e}_{p}\right\| &= \sqrt{\mathbf{e}_{xpV}{}^{2} + \mathbf{e}_{ypV}{}^{2} + \mathbf{e}_{zpV}{}^{2}} \\ \left\|\mathbf{e}_{r}\right\| &= \sqrt{\mathbf{e}_{xrV}{}^{2} + \mathbf{e}_{yrV}{}^{2} + \mathbf{e}_{zrV}{}^{2}} \end{aligned} \tag{5.31}$$

5.9.1 Manual error compensation

The following experiment analyses the behavior of manual compensation (expression (5.20)) in the presence of position deviation. A target position $\boldsymbol{T}_{TAR}$ was defined at $\mathbf{p}_{TAR} = [621 \quad 0 \quad 548]^{T}$ (mm), and $\mathbf{rpy}_{TAR} = [-90 \quad 0 \quad -90]^{T}$ (deg), where *rpy* denotes the roll-pitch-yaw notation of the orientation of the TCP [29]. The TCP was defined exactly at the end-effector of the robot, i.e. ${}^{EE}\boldsymbol{T}_{TCP}$ contains an identity rotation matrix and position coordinates equal to zero. The robot's end-effector was located at $\mathbf{p}_{EE} = [621 \quad 0 \quad 554]^T$ and $\mathbf{rpy}_{EE} = \mathbf{rpy}_{TAR}$, which represents a deviation of 6 mm along Z-axis from $\boldsymbol{T}_{TAR}$. A VF was created to move the end-effector along Y-axis with respect to the base frame while keeping the orientation constant, i.e. $S^l = \{\boldsymbol{R}_{TCP}{}^T \mathbf{l}_1\}$ and $S^\alpha = \{0\}$, where $\mathbf{l}_1 = [0 \quad 1 \quad 0]^T$. The experiment consists of moving the end-effector back and forth by hand along the preferred direction. Several tries were executed with different values of gain $k_d \in [0,1]$. Figure (5.14) shows the influence of k_d onto equation (5.20). No error compensation occurs when $k_d = 0$, which can be observed in the behavior of the end-effector along the X-axis. It can be seem in the Z-axis that increasing the value of k_d yields into a faster compensation of the position error. Notice however that the end-effector orientation deviates considerable from the desired one despite the value of k_d (see Figure 5.15). Finally, notice in Figure 5.16 that, although the orientation components are not influenced by k_d during translational movements, the position error is reduced when incrementing the value of k_d. A notable performance enhancement occurs for values up to $k_d = 0.9$, while higher values produce no a significant improvement.

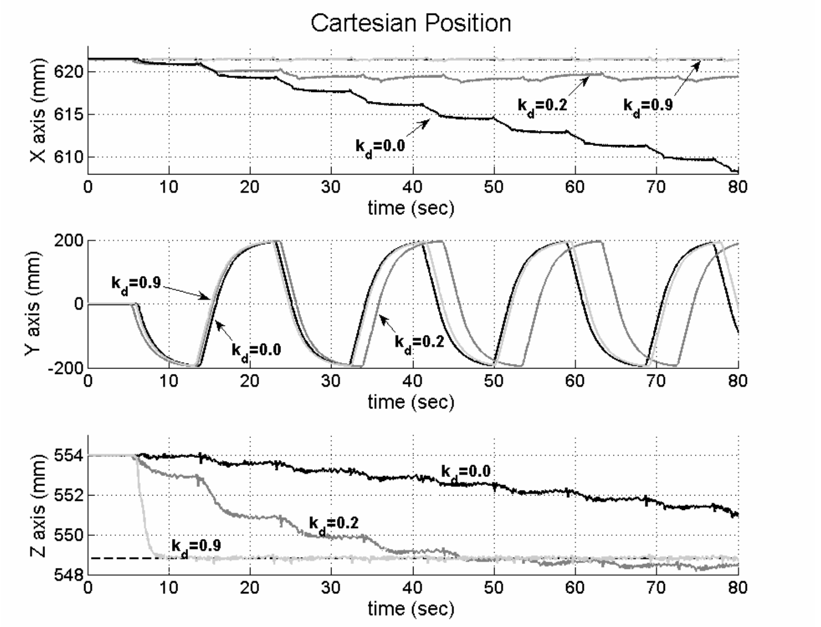


Figure 5.14. Influence of gain k_d on manual error compensation of end-effector position along Z-axis while moving it along Y-axis w.r.t. world coordinates.

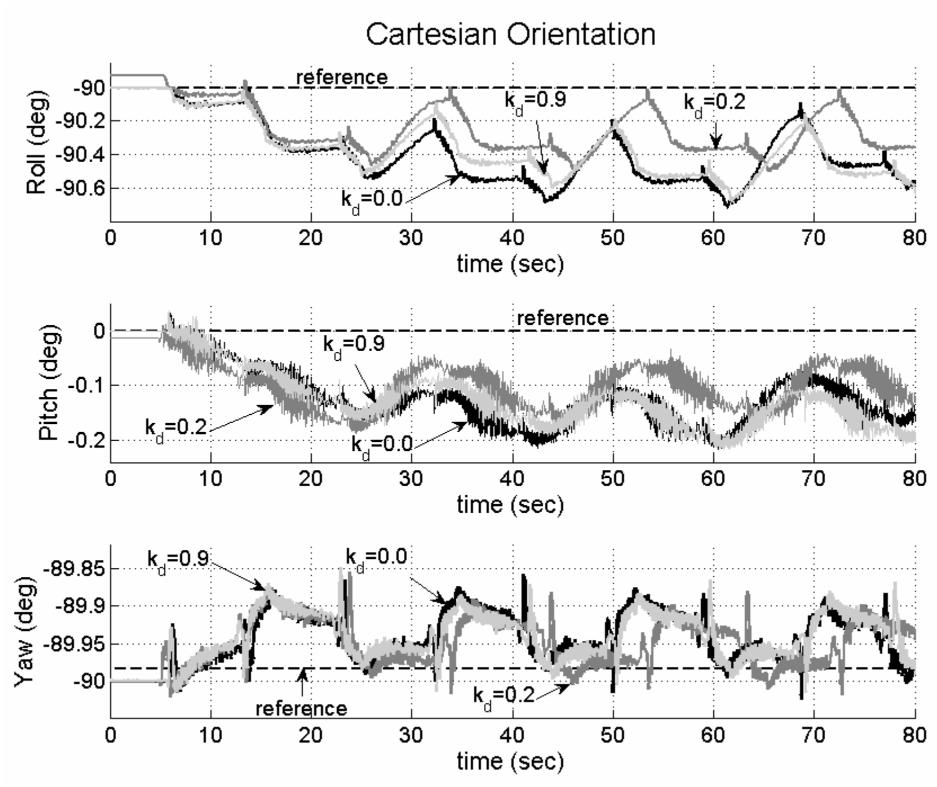


Figure 5.15. Influence of gain k_d on manual error compensation of end-effector orientation while moving it along Y-axis w.r.t. world coordinates.

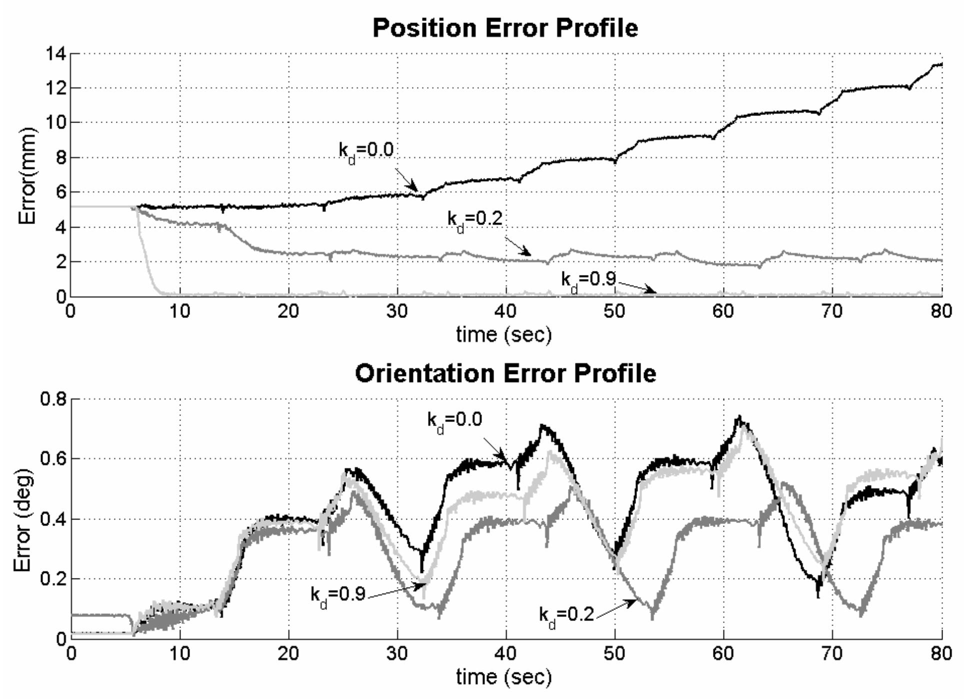


Figure 5.16. Influence of gain k_d on the error norm of manual compensation while moving the end-effector along Y-axis w.r.t. world coordinates.

5.9.2 Manual compensation vs. autonomous compensation

The following two experiments compare the behavior of both manual and autonomous compensation. In the first case, a translational VF is setup while the second case concerns a rotational VF.

5.9.2.1 Translational case

The setup of this experiment is similar to that explained above, the only difference being that the initial TCP position is equal the target position, i.e. $\boldsymbol{T}_{TCP} = \boldsymbol{T}_{TAR}$. The objective is to analyze the efficiency of manual and autonomous controllers (expressed in equations (5.20) and (5.22), respectively) to keep the error at minimum along the non-preferred directions while moving along a preferred one. Therefore a translational VF is defined as follows: $S^l = \{\boldsymbol{R}_{TCP}{}^T \mathbf{l}_1\}$ and $S^\alpha = \{0\}$, where $\mathbf{l}_1 = [0 \quad 1 \quad 0]^T$. The gain values of the controllers are $k_d = 0.9$ for the manual controller and $k_v = 5$ for the autonomous one. Additionally, an attempt with no error compensation is included to provide an additional benchmark for the results comparison.

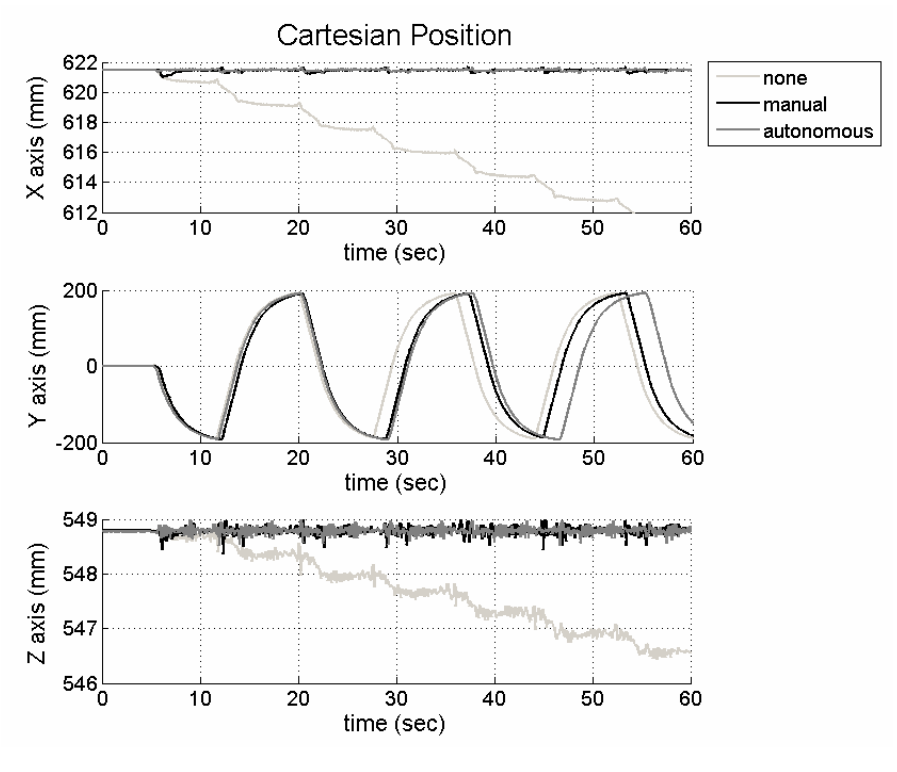


Figure 5.17. End-effector position while moving along the Y-axis w.r.t. world coordinates.

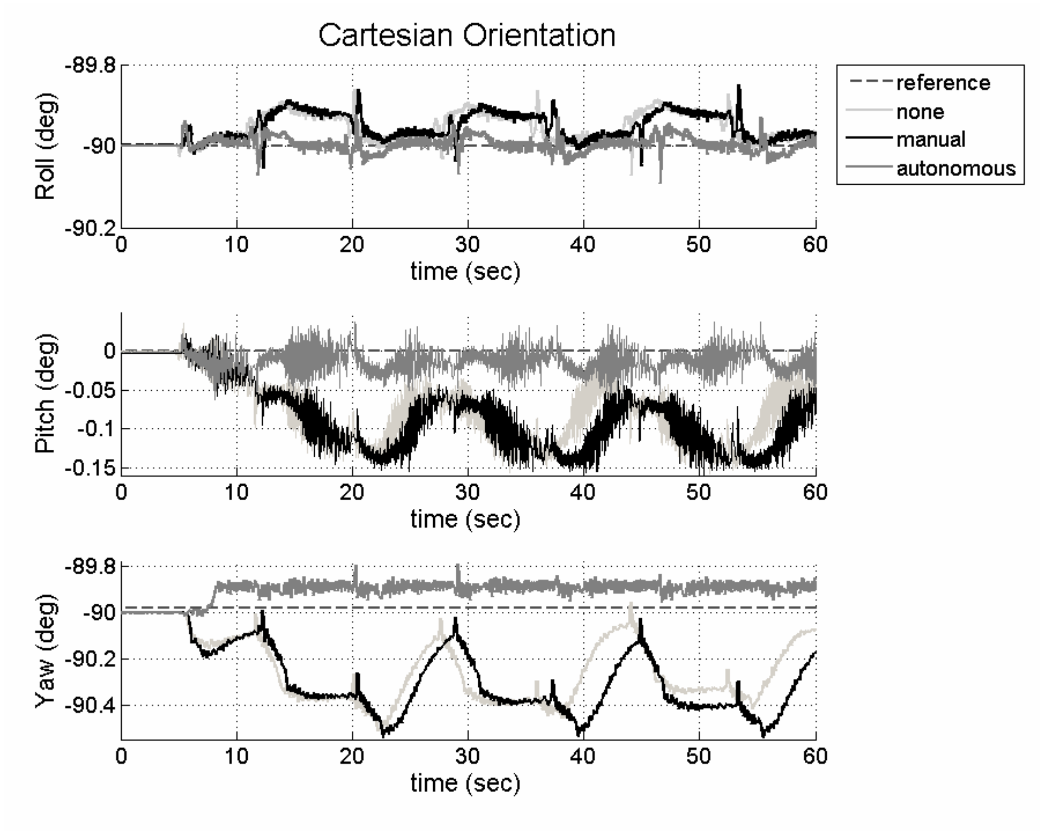


Figure 5.18 End-effector orientation while moving along the Y-axis w.r.t. world coordinates.

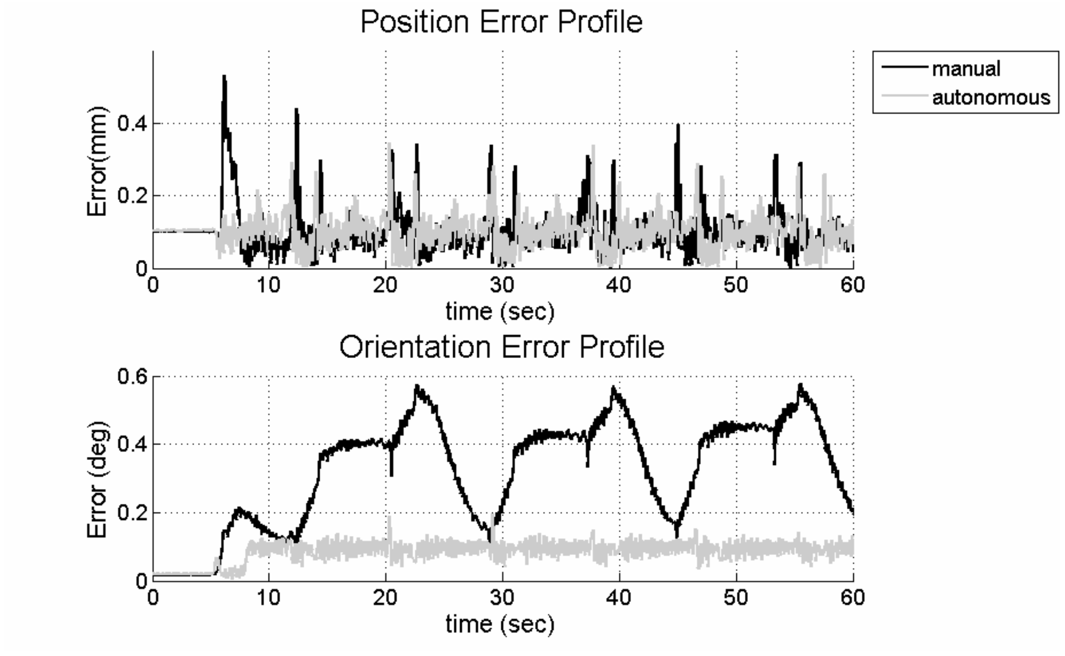


Figure 5.19. Error profile of manual and autonomous error compensation while moving along the Y-axis w.r.t. world coordinates.

The position behavior, orientation behavior and instantaneous quadratic error norm are presented in Figure 5.17, Figure 5.18 and Figure 5.19, respectively. Both controllers present a similar behavior concerning the position, both having a position error of the same order. However, it is in the orientation error where a great difference arises. While the orientation error behavior of the manual compensation looks very similar to the case where no compensation at all occurs, the error is strongly reduced when applying autonomous error compensation. Table 5.1 shows the mean position and orientation error for the three cases: no compensation, manual compensation and autonomous compensation.

Table 5.1. Mean value of position and orientation error of translational VF

Control	Mean Position error (mm)	Mean Orientation error (deg)
None	6.4582	0.3044
Manual	0.1006	0.3492
Autonomous	0.1012	0.0831

5.9.2.2 Rotational case

This experiment evaluates the error compensation when a rotational VF is applied. Three responses are compared: no compensation, manual compensation (equations (5.20)) and autonomous compensation (equation (5.22)). The rotational VF consists of a pivot rotation of the end-effector 360° back and forth about a rotation axis parallel to the Z-axis of the robot base frame (see Figure 5.20) and positioned at a specific point in the space, while keeping a constant inclination ($\beta = 45°$)of the tool with respect to the rotation axis. In this case, the TCP is not more equal to end-effector position. The transformation ${}^{EE}\boldsymbol{T}_{TCP}$ from TCP to end-effector is:

$$
{}^{EE}\boldsymbol{T}_{TCP} = \begin{bmatrix} 0.998 & -0.06 & -0.016 & -3 \\ 0.017 & 0.011 & 1 & 219 \\ -0.06 & -0.998 & 0.012 & 129 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The initial pose of TCP, also used as reference target pose $\boldsymbol{T}_{TAR}$ is:

$$\boldsymbol{T}_{TAR} = \boldsymbol{T}_{0TCP} = \begin{bmatrix} -0.0476 & -0.5816 & -0.8122 & -486.4 \\ -0.998 & 0.0601 & 0.015 & 3.2 \\ 0.04 & -0.8111 & -0.5835 & 260.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A VF was created in order to rotate the tool around Z-axis with respect to the base frame while keeping the relative angle relationship β constant, i.e. $S^l = \{0\}$ and $S^\alpha = \{\boldsymbol{R}_{TAR}^T \boldsymbol{\alpha}_1\}$, where $\boldsymbol{\alpha}_1 = [0 \quad 0 \quad 1]^T$. Contrary to the experiment of sections 5.9.1 and 5.9.2.1, $\boldsymbol{R}_{TAR}$ is used instead of $\boldsymbol{R}_{TCP}$ to define the rotational VF. The latter rotation matrix would mean that the VF would stay constant with respect to the tool coordinates, while this experiment requires a VF that stays constant with respect to the base coordinates. Depending on the application it may be desired the virtual fixture to come along with the tool or to define it with respect to the tool and afterward keep it constant. In the first case $\boldsymbol{T}_{TCP}$ is used while $\boldsymbol{T}_{TAR}$ is applied to the second one.

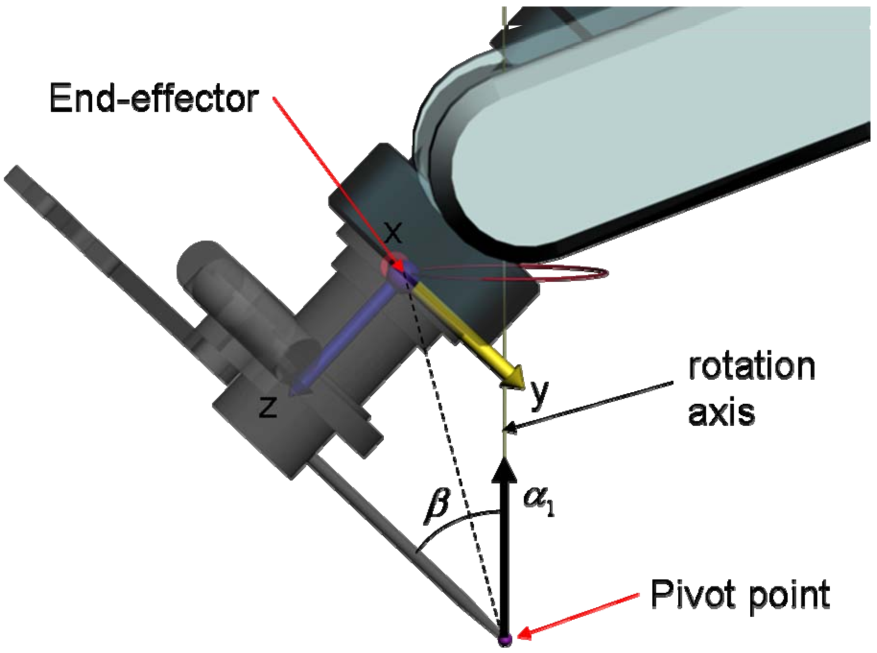


Figure 5.20. Experimental setup for rotational case

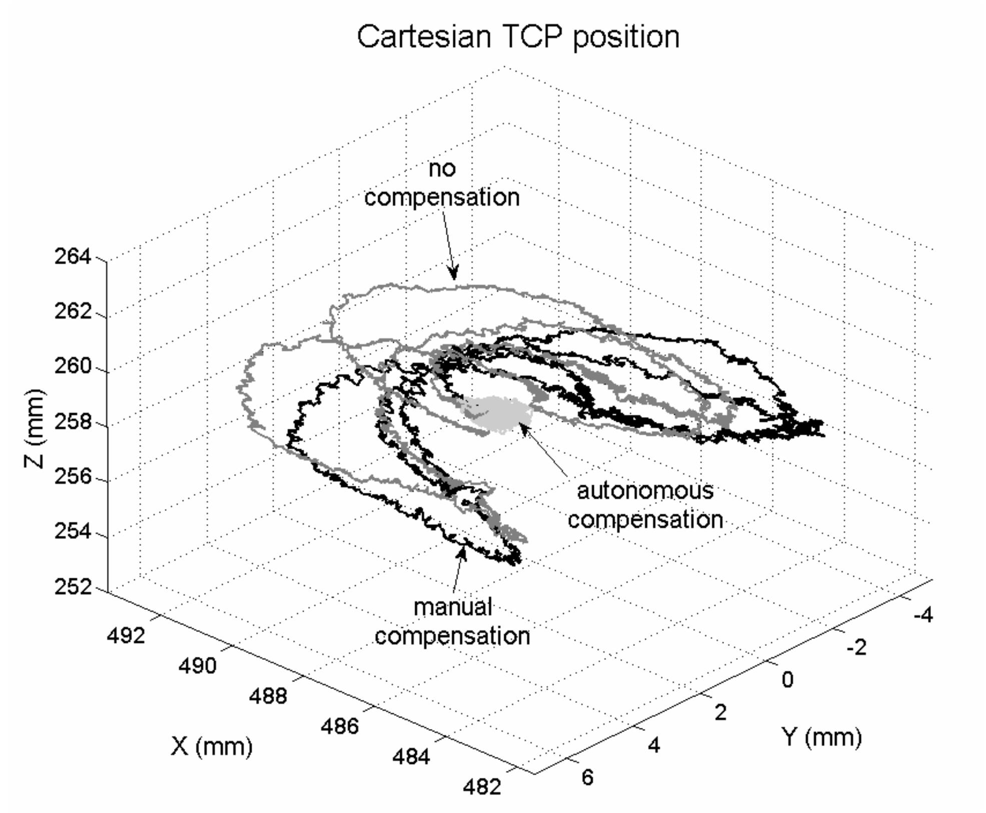


Figure 5.21. Tool tip position in 3D while pivot rotation above Z axis

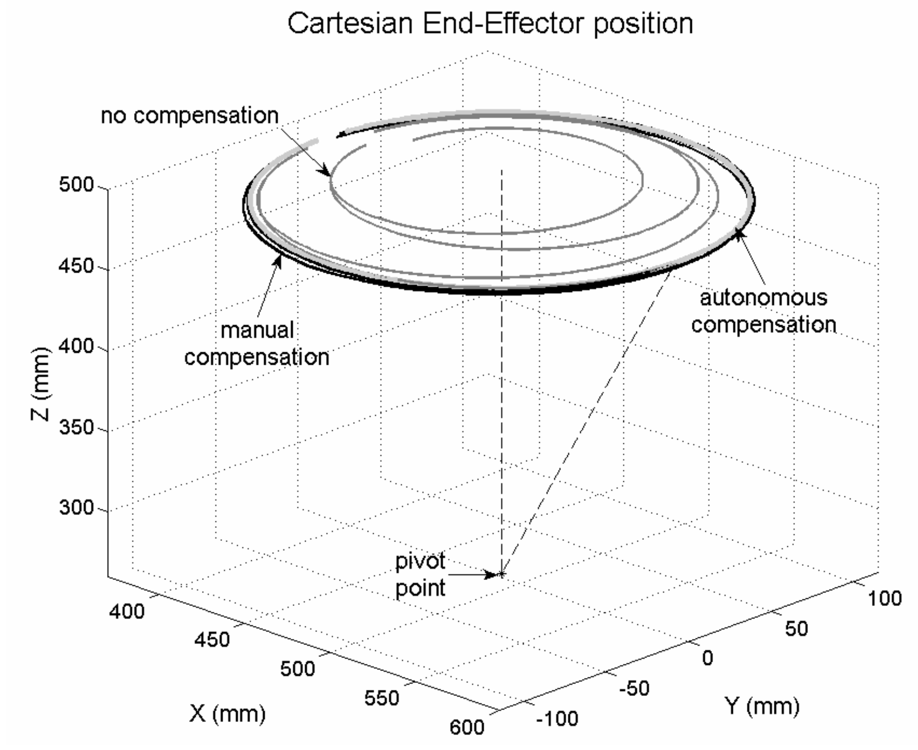


Figure 5.22. Tool tip position in 3D while pivot rotation above Z axis

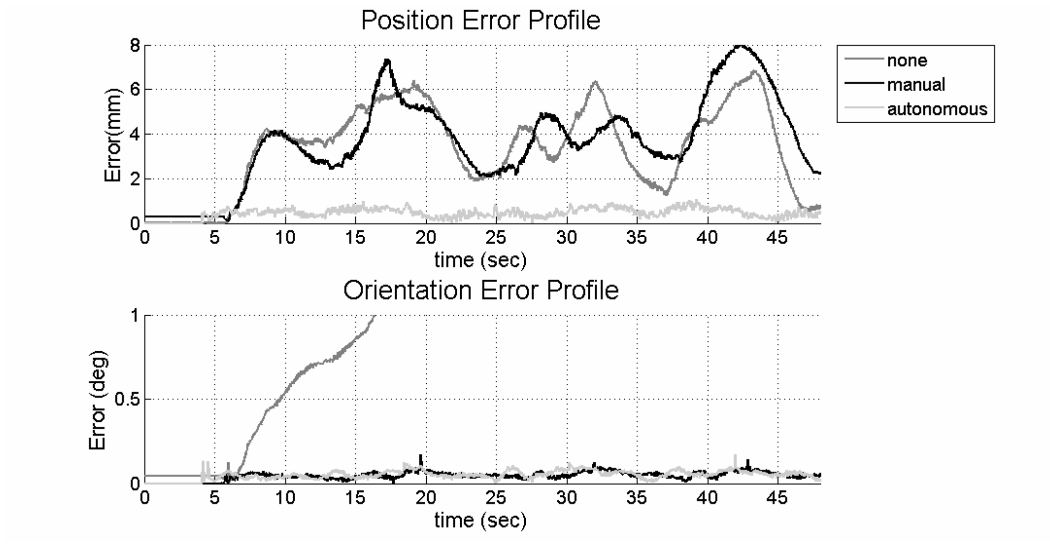


Figure 5.23. End-effector error profile while pivot rotation above Z axis

The results of this experiment reveal that the decoupling of position and orientation during manual compensation occurs in a similar way as in the translational case. Now, orientation movements are executed and the position should be kept constant. Figure 5.21 shows that the position of the TCP presents strong deviations from the reference pivot point in the cases of no compensation and manual compensation, having both of them the same patron. Autonomous compensation, on the contrary, reduces the position error. The orientation error is reduced in a similar manner with both manual and autonomous compensation, while the error continuously increases when no compensation is executed (see Figure 5.22). The deviations on the end-effector trajectory during manual compensation, easily observed in Figure 5.22, are more because of position deviation rather than orientation deviation. Finally, the quadratic error norm plotted in Figure 5.23 corroborates the behavior just explained above. The corresponding mean errors are presented in Table 5.2.

Table 5.2. Mean value of position and orientation error of rotational VF

Control	Mean Position error (mm)	Mean Orientation error (deg)
None	3.1758	2.4977
Manual	3.4864	0.0440
Autonomous	0.435	0.0495

5.9.3 Moving along a trajectory

The previous experiments evaluate performance of manual and autonomous error compensation in a virtual constrained subspace which was defined with respect to a static $\boldsymbol{T}_{TAR}$. In the following experiments a parametric function is used to specify the reference target curve along which the VF is to be applied. The $\boldsymbol{T}_{TAR}$ is calculated on-line as the closest point of the curve to the TCP (see Appendix B). The virtual fixture is a translational virtual unit pointing always along the tangential direction of the curve while the orientation is maintained constant. Both manual and autonomous error compensation were tested with two different trajectories: (a) a sinusoidal trajectory on the ZX-plane with respect to the base frame, and (b) a circular trajectory on the XY-plane also with respect to the base frame. During the experiments the user has to follow the reference paths two times back and forth until the boundaries are reached.

Figure 5.24 and Figure 5.26 show the TCP Cartesian position during sinus and circle experiments, respectively. Notice that the plots are not homogenous scaled, the axis along which end-effector position remains constant has a very small scale in comparison with the other two axes, so that the error difference between manual and autonomous compensation can be observed. The norm of the error during sinusoidal trajectory and circular trajectory are presented in Figure 5.25 and Figure 5.27, respectively. Finally, the resulting mean error during both trajectories can be consulted in Table 5.3. In both trajectories the error is smaller during autonomous compensation.

Table 5.3. Mean error while following a reference trajectory

	Sinus		**Circle**	
Control	**Mean Position error (mm)**	**Mean Orientation error (deg)**	**Mean Position error (mm)**	**Mean Orientation error (deg)**
Manual	0.6123	0.1186	0.8978	0.0719
Autonomous	0.2618	0.0478	0.2998	0.0477

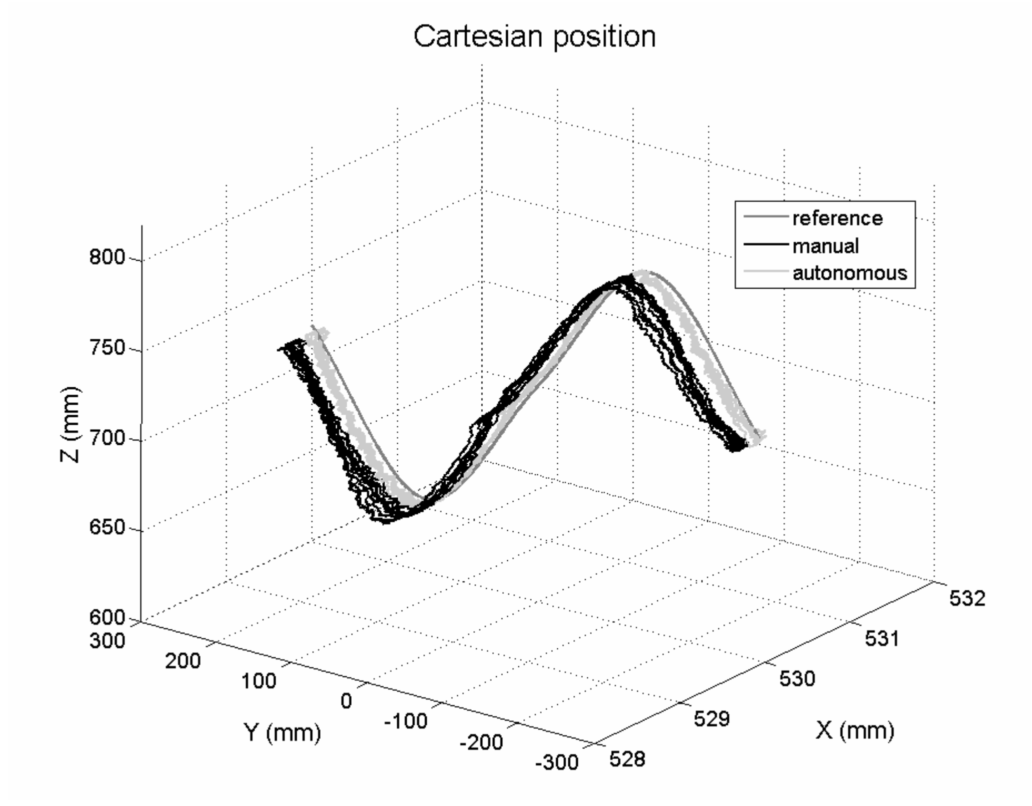


Figure 5.24. End-effector position while following a sinusoidal trajectory

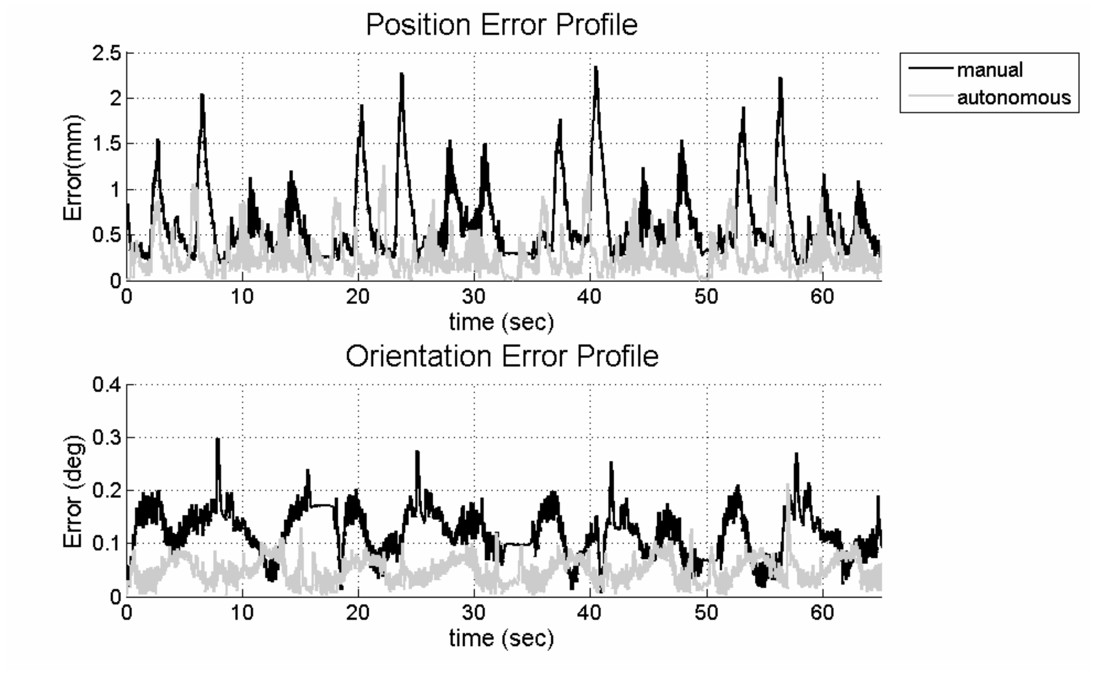


Figure 5.25. End-effector error profile while following a sinusoidal trajectory

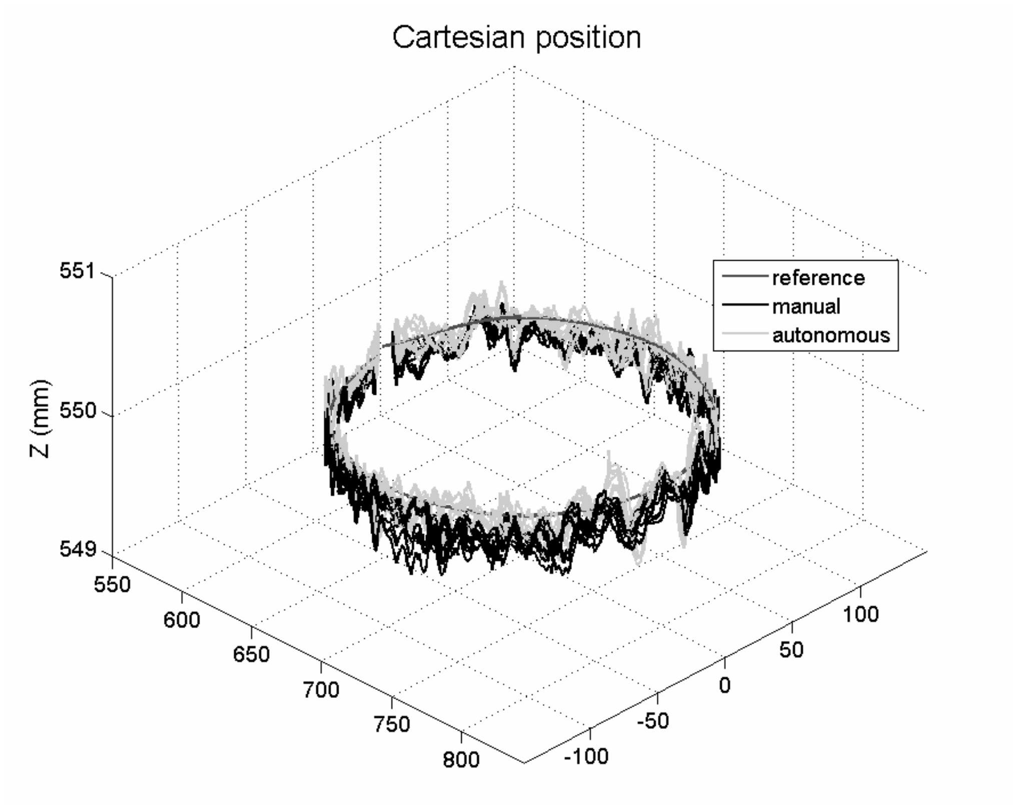


Figure 5.26. End-effector position while following a circular trajectory

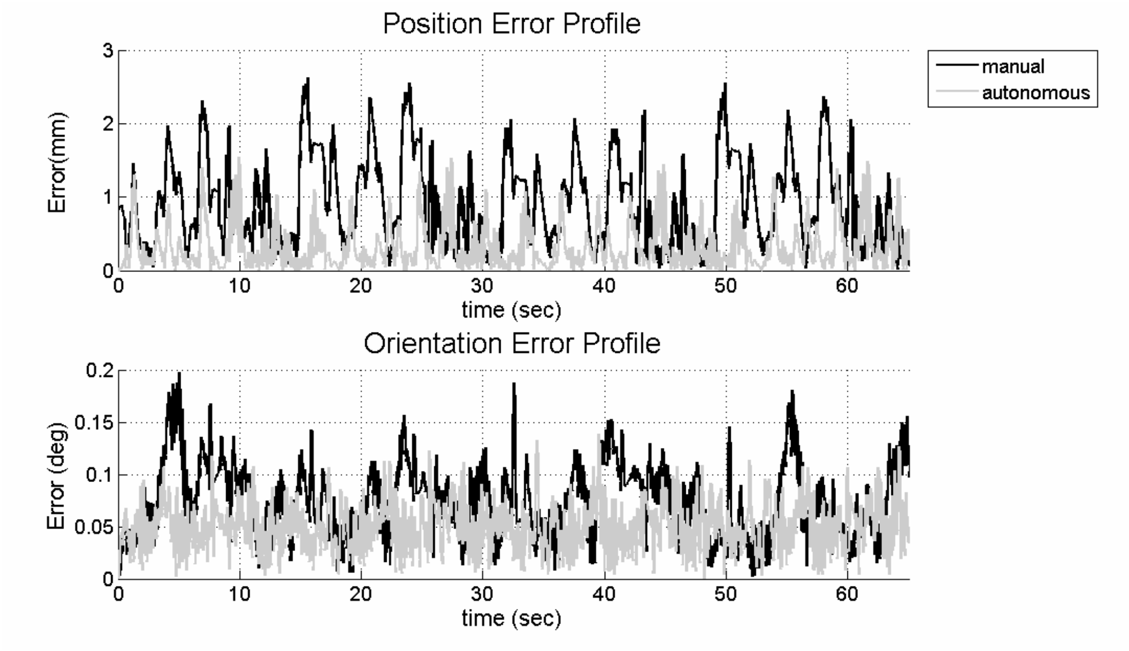


Figure 5.27. End-effector error profile while following a circular trajectory

5.10 Discussion

The two methods presented in this section were compared for error compensation during cooperative manipulation of the tool along virtually constrained subspaces: the manual compensation and autonomous compensation. The philosophy behind manual compensation states that the user is the only one been able to generate any kind of motion, while the robotic system is more like a passive system with the sole job of constraining the possible movements into an allowed subspace. This is done by means of the so called virtual fixtures. In the presence of a deviation error, the manual controller redefines such VFs to include the direction needed in order to compensate for such an error, having then one new direction that guarantees the error compensation. The experiments presented above have shown that the compensation takes place as long as an input force induced by the user is applied. Unfortunately, the translational and rotational movements are not directly coupled between each other. This means that when performing one of these two types of movements, any deviation error appearing on the other type of movement may not be necessarily compensated. The reason is that, due to the nature of the movement, despite the error being detected by the controller, and a new VF being redefined to compensate for the error, the user may not realize that the generation of such motions is required. Thus, if no movement is induced by the user in such direction, the compensation does not occur.

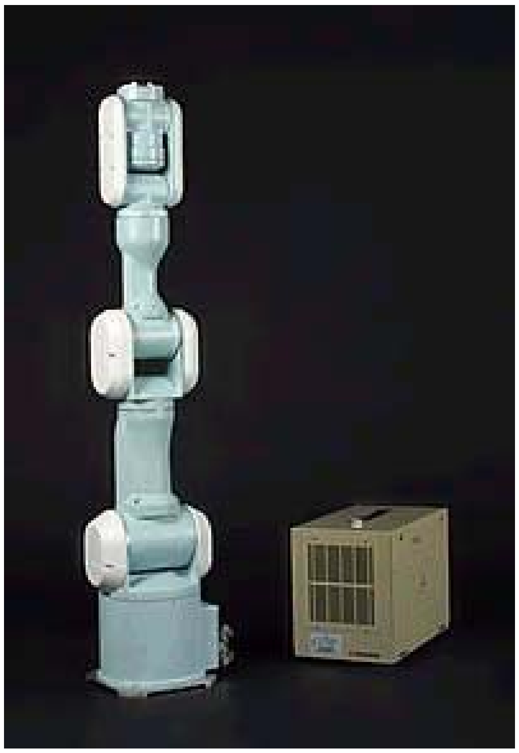
The concept of virtual fixtures with autonomous error compensation is then proposed to deal with this drawback of manual compensation. The main idea is to give the robotic system the responsibility of error compensation while the user keeps complete control inside the allowed subspace. This has the disadvantage that the robot itself is able to generate motion which may be undesired for the sake of safeness. For instance, suppose that a virtual fixture is defined by mistake on a target pose $\boldsymbol{T}_{TAR}$ which is far away from the current position of the end-effector and the autonomous compensation is active. At the moment that the user activates the cooperative mode, the robot would automatically begin to compensate the error, producing an unexpected and even more undesirable movement which could lead serious consequences. This is not the case if manual compensation is active. In such a case, the controller redefines the VF and the robot waits until the user compensates the error by his/her own, which is intrinsically safer than with the autonomous controller. The combination of both controllers is proposed as a solution to this safety issue by establishing an error threshold above which the manual compensator becomes active while autonomous compensation can only run below this value. This means that the main objective of autonomous compensation is to keep the TCP inside the preferred subspace rather than getting the TCP into it. Once the preferred subspace is reached, i.e. the error is below the threshold, autonomous compensation becomes active.

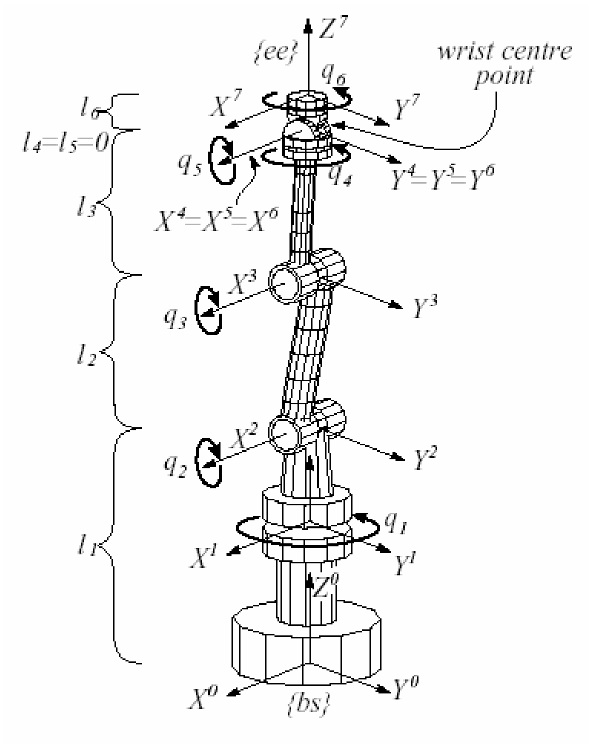
The main advantage of autonomous compensation in comparison to manual compensation is that the decoupling nature of translation and rotation is not more a problem since while doing movements of the one type, possible deviation error of the other type is automatically compensated.

6. Singularity robustness

6.1 PA10 kinematics

The kinematic description of the PA10-6C robot arm used in this work is based on the Featherstone convention [38]. Its frames are shown in Figure 6.1, having only four parameters, i.e. link lengths l_1, l_2 l_3 and l_6, while the wrist link lengths l_4 and l_5 are zero. A detailed description of the Featherstone convention is out of the scope of this work and only final results of the symbolic calculation of the forward position and velocity kinematics are presented, since these are imperative for further implementation of the singularity robust strategies treated here. For further details, the reader can consult the following literature [20], [38], [81].





(a) (b).

Figure 6.1. (a) PA10-6C robot arm (courtesy of Mitsubishi Heavy Industries), (b) Kinematic description of PA10-6C based on Featherstone [15]

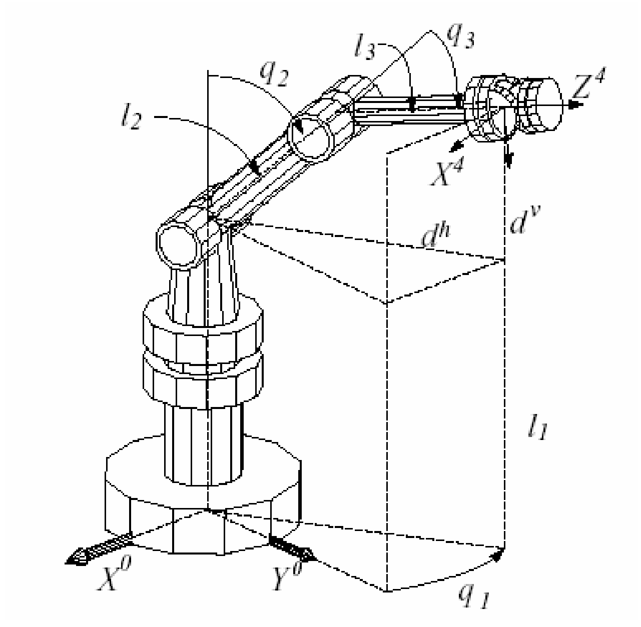


Figure 6.2. Kinematics of first three joints of the 321 manipulator [15]

6.1.1 Forward position kinematics

The forward position kinematics (FPK) is intended to find the end-effector pose from a given set of joints position $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_6 \end{bmatrix}^T$. The solution is always unique, that means, one given joint position vector always corresponds to only one single end-effector pose.

The forward mapping to obtain the relative orientation of the end-effector frame $\{EE\}$ with respect to the frame $\{4\}$ is obtained by the calculation of the relative orientation of $\{EE\}$ with respect to $\{6\}$, $\{6\}$ with respect to $\{5\}$, and $\{5\}$ with respect to $\{4\}$ as follows:

$$
\begin{aligned}
{}^{3}\boldsymbol{R}_{6} &= \boldsymbol{R}(Z,\theta_4)\,\boldsymbol{R}(-X,\theta_5)\,\boldsymbol{R}(Z,\theta_6) \qquad (6.1)\\
&= \begin{bmatrix} c_4 & -s_4 & 0 \\ s_4 & c_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_5 & s_5 \\ 0 & -s_5 & c_5 \end{bmatrix}
\begin{bmatrix} c_6 & -s_6 & 0 \\ s_6 & c_6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_6c_4 - s_6c_5s_4 & -s_6c_4 - c_6c_5s_4 & -s_5s_4 \\ c_6s_4 + s_6c_5c_4 & -s_6s_4 + c_6c_5c_4 & s_5c_4 \\ -s_6s_5 & -c_6s_5 & c_5 \end{bmatrix}
\end{aligned}
$$

where c_4 stands for $\cos(\theta_4)$ and so on. The resulting homogeneous transformation matrix from {6} to {4} is presented below:

$$ {}^{3}\boldsymbol{T}_6 = \begin{bmatrix} {}^{3}\boldsymbol{R}_6 & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \tag{6.2}$$

The determination of the pose of the wrist reference frame {4} with respect to the base reference frame {0} = { bs } of the robot is:

$$ {}^{0}\boldsymbol{R}_3 = {}^{0}\boldsymbol{R}_1\,{}^{1}\boldsymbol{R}_3 = \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{bmatrix} = \begin{bmatrix} c_1 & -s_1c_{23} & -s_1s_{23} \\ s_1 & c_1c_{23} & c_1s_{23} \\ 0 & -s_{23} & c_{23} \end{bmatrix} \tag{6.3}$$

for the orientation of the wrist with respect to the base frame, and

$$ \mathbf{p}_{wr} = {}^{0}\mathbf{p}_{wr} = \begin{bmatrix} -s_1 d^h \\ c_1 d^h \\ l_1 + d^v \end{bmatrix}, \tag{6.4}$$

with

$$ \begin{aligned} d^v &= c_2 l_2 + c_{23} l_3 \\ d^h &= s_2 l_2 + s_{23} l_3 \end{aligned}, \tag{6.5}$$

for the position of the wrist with respect to the base.

Finally, the pose of the end-effector reference frame {7} = {EE} with respect to the last wrist reference frame {6} (i.e., ${}^{6}\boldsymbol{T}_7$) corresponds to a translation along Z^6 over a distance l_6:

$$ {}^{6}\boldsymbol{T}_7 = \begin{bmatrix} {}^{6}\boldsymbol{R}_7 & \begin{bmatrix} 0 & 0 & l_6 \end{bmatrix}^T \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

Hence, the total orientation ${}^{Base}\boldsymbol{R}_{EE}$ follows from equations (6.1), (6.3), and (6.6):

$$ \boldsymbol{R}_{EE} = {}^{0}\boldsymbol{R}_7 = {}^{0}\boldsymbol{R}_3\,{}^{3}\boldsymbol{R}_6\,{}^{6}\boldsymbol{R}_7 \tag{6.7}$$

The position of the wrist centre (i.e., the origin of {4}) with respect to the base {0} is

$$\mathbf{p}_{wr} = {}^{0}\mathbf{p}_{wr} = \begin{bmatrix} -s_1 d^h \\ c_1 d^h \\ l_1 + d^v \end{bmatrix}, \tag{6.8}$$

and the position of the end-effector (i.e., the origin of {*EE*} with respect to the base {0}) is

$$\mathbf{p}_{EE} = \mathbf{p}_{wr} + \boldsymbol{R}_{EE} \begin{bmatrix} 0 & 0 & l_6 \end{bmatrix}^T . \tag{6.9}$$

Finally, equations (6.7) and (6.9) yield into the homogeneous transformation matrix from the end-effector frame {*EE*} to the base frame {*Base*}:

$$\boldsymbol{T}_{EE} = \begin{bmatrix} \boldsymbol{R}_{EE} & \mathbf{p}_{EE} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}. \tag{6.10}$$

with

$$\begin{aligned}
r_{11} &= -s_1 c_{23}(c_4 c_5 s_6 + s_4 c_6) - c_1(s_4 c_5 s_6 - c_4 c_6) + s_1 s_{23} s_5 s_6 \\
r_{21} &= c_1 c_{23}(c_4 c_5 s_6 + s_4 c_6) - s_1(s_4 c_5 s_6 - c_4 c_6) - c_1 s_{23} s_5 s_6 \\
r_{31} &= -s_{23}(c_4 c_5 s_6 + s_4 c_6) - c_{23} s_5 s_6 \\
r_{12} &= -s_1 c_{23}(c_4 c_5 c_6 - s_4 s_6) - c_1(s_4 c_5 c_6 + c_4 s_6) + s_1 s_{23} s_5 c_6 \\
r_{22} &= c_1 c_{23}(c_4 c_5 c_6 - s_4 s_6) - s_1(s_4 c_5 c_6 + c_4 s_6) - c_1 s_{23} s_5 c_6 \\
r_{32} &= -s_{23}(c_4 c_5 c_6 - s_4 s_6) - c_{23} s_5 c_6 \\
r_{13} &= -s_1 c_{23} c_4 s_5 - c_1 s_4 s_5 - s_1 s_{23} c_5 \\
r_{23} &= c_1 c_{23} c_4 s_5 - s_1 s_4 s_5 + c_1 s_{23} c_5 \\
r_{33} &= -s_{23} c_4 s_5 + c_{23} c_5
\end{aligned}$$

and

$$\mathbf{p}_{EE} = \begin{bmatrix} r_{13} \cdot l_6 - s_1 d^h \\ r_{23} \cdot l_6 + c_1 d^h \\ r_{33} \cdot l_6 + l_1 + d^v \end{bmatrix}$$

where r_{ij} denotes the element of the *i*th row and *j*th column of $\boldsymbol{R}_{EE}$. The homogeneous transformation matrix $\boldsymbol{T}_{EE}$ express the position and orientation of the end-effector with respect to the base {*Base*} in function of the joint variables $\boldsymbol{\theta}$ [15].

6.1.2 Forward velocity kinematics

The forward velocity kinematics (FVK) calculates the resulting end-effector linear velocity $\dot{\mathbf{p}}$ and angular velocity $\boldsymbol{\omega}$. given the joint position $\boldsymbol{\theta}$ and joint velocity $\dot{\boldsymbol{\theta}}$. The FVK is always unique and the relationship between the joint velocities and the linear and angular velocity of the end effector is linear, i.e., if joint velocity is incremented by a factor of two, the end-effector velocity will increment by a factor of two too. This velocity relationship is then determined by means of the *Jacobian* matrix as follows:

$$\dot{\mathbf{x}} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}, \tag{6.11}$$

where $\boldsymbol{J}(\boldsymbol{\theta})$ is the manipulator Jacobian matrix with dimensions $m \times n$, which relates joint velocities $\dot{\boldsymbol{\theta}}$ to the end-effector velocity $\dot{\mathbf{x}}$. For non-redundant manipulators, $m = n$ while for redundant manipulators $m < n$. In the case of PA10-6C, which is a non-redundant robot, $\boldsymbol{J}$ is a 6×6 square matrix. As a physical interpretation, the ith column of $\boldsymbol{J}$ can be thought of as the end-effector linear and angular velocities generated by a unit velocity applied at the ith joint, and zero velocities at the others. Notice that the matrix itself depends non-linearly on the joint position vector $\boldsymbol{\theta}$.

In general terms, the methodology for the calculation of the end-effector velocities, suggest the sum of the joint velocities successively starting at the base frame {*Base*}. But by taking advantage of the robot kinematic structure and using the results of the FPK, this procedure is made more efficient.

First of all, the angular velocity of the end-effector with respect to the wrist expressed in the frame {4} is deduced by inspection of Figure 6.1. This yields into:

$$\begin{bmatrix} {}^{wr}\omega_{xEE} \\ {}^{wr}\omega_{yEE} \\ {}^{wr}\omega_{zEE} \end{bmatrix} = \begin{bmatrix} 0 & -c_4 & -s_4 s_5 \\ 0 & -s_4 & c_4 s_5 \\ 1 & 0 & c_5 \end{bmatrix} \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{6.12}$$

The angular velocities of the remaining joints are then added. Again, by inspection it is easily inferable that the angular velocity of end-effector with respect to link 1 expressed in the frame {1} looks like

$$
{}^{1}\boldsymbol{\omega} = \begin{bmatrix} {}^{wr}\omega_{xEE} - \dot{\theta}_2 - \dot{\theta}_3 \\ {}^{wr}\omega_{yEE} c_{23} + {}^{wr}\omega_{zEE} s_{23} \\ {}^{wr}\omega_{zEE} c_{23} - {}^{wr}\omega_{yEE} s_{23} \end{bmatrix} \tag{6.13}
$$

It is only matter of adding the last pendent joint angular velocity $\dot{\theta}_1$ and express the whole expression with respect to the base frame $\{bs\}$:

$$
\boldsymbol{\omega}_{EE} = \begin{bmatrix} {}^{1}\omega_x c_1 - {}^{1}\omega_y s_1 \\ {}^{1}\omega_y c_1 - {}^{1}\omega_x s_1 \\ {}^{1}\omega_z + \dot{\theta}_1 \end{bmatrix} \tag{6.14}
$$

Now, the linear velocity can be calculated, conveniently, as the sum of the linear velocity of the end-effector with respect to the wrist ${}^{wr}\dot{\mathbf{p}}_{EE}$ and the linear velocity of the wrist with respect to the base frame $\dot{\mathbf{p}}_{wr}$. Thereto, both velocities must be represented in the same coordinate system, in this case the base frame $\{bs\}$.

$$
\dot{\mathbf{p}} = \dot{\mathbf{p}}_{EE} + \dot{\mathbf{p}}_{wr} \tag{6.15}
$$

where

$$
\dot{\mathbf{p}}_{EE} = \boldsymbol{\omega} \times \boldsymbol{R}_{EE} \begin{bmatrix} 0 & 0 & l_6 \end{bmatrix}^T, \tag{6.16}
$$

and

$$
\dot{\mathbf{p}}_{wr} = \begin{bmatrix} c_1({}^{1}\dot{p}_{xwr}) - s_1({}_{1}\dot{p}_{ywr}) \\ s_1({}^{1}\dot{p}_{xwr}) + c_1({}_{1}\dot{p}_{ywr}) \\ {}^{1}\dot{p}_{zwr} \end{bmatrix}, \tag{6.17}
$$

with

$$
{}^{1}\dot{\mathbf{p}}_{wr} = \begin{bmatrix} -d^h\dot{\theta}_1 \\ d^v\dot{\theta}_2 + l_3 c_{23}\dot{\theta}_3 \\ -d^h\dot{\theta}_2 - l_3 s_{23}\dot{\theta}_3 \end{bmatrix}. \tag{6.18}
$$

Since the angular velocity $\boldsymbol{\omega}$ and the linear velocity $\dot{p}$ are calculated in function of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$, the Jacobian can be factored out. Resulting the final expression

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & j_{13} & j_{14} & j_{15} & j_{16} \\ j_{21} & j_{22} & j_{23} & j_{24} & j_{25} & j_{26} \\ j_{31} & j_{32} & j_{33} & j_{34} & j_{35} & j_{36} \\ j_{41} & j_{42} & j_{43} & j_{44} & j_{45} & j_{46} \\ j_{51} & j_{52} & j_{53} & j_{54} & j_{55} & j_{56} \\ j_{61} & j_{62} & j_{63} & j_{64} & j_{65} & j_{66} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{6.19}$$

with

$$\begin{array}{lll}
j_{11} = -c_1 d^h - l_{6y} & j_{21} = s_1 d^h + l_{6x} & j_{31} = 0 \\
j_{12} = -s_1 d^v - s_1 l_{6z} & j_{22} = c_1 d^v + c_1 l_{6z} & j_{32} = -d^h - c_1 l_{6y} + s_1 l_{6x} \\
j_{13} = -s_1 c_{23} l_3 - s_1 l_{6z} & j_{23} = c_1 c_{23} l_3 + c_1 l_{6z} & j_{33} = -s_{23} l_3 - c_1 l_{6y} + s_1 l_{6x} \\
j_{14} = j_{54} l_{6z} - j_{64} l_{6y} & j_{24} = j_{64} l_{6x} - j_{44} l_{6z} & j_{34} = j_{44} l_{6y} - j_{54} l_{6x} \\
j_{15} = j_{55} l_{6z} - j_{65} l_{6y} & j_{25} = j_{65} l_{6x} - j_{45} l_{6z} & j_{35} = j_{45} l_{6y} - j_{55} l_{6x} \\
j_{16} = j_{56} l_{6z} - j_{66} l_{6y} & j_{26} = j_{66} l_{6x} - j_{46} l_{6z} & j_{36} = j_{46} l_{6y} - j_{56} l_{6x}
\end{array}$$

$$\begin{array}{lll}
j_{41} = 0 & j_{51} = 0 & j_{61} = 1 \\
j_{42} = -c_1 & j_{52} = -s_1 & j_{62} = 0 \\
j_{43} = -c_1 & j_{53} = -s_1 & j_{63} = 0 \\
j_{44} = -s_1 s_{23} & j_{54} = c_1 s_{23} & j_{64} = c_{23} \\
j_{45} = s_1 c_{23} s_4 - c_1 c_4 & j_{55} = -s_1 c_4 - c_1 c_{23} s_4 & j_{65} = s_{23} s_4 \\
j_{46} = -c_1 s_4 s_5 - s_1 c_{23} c_4 s_5 - s_1 s_{23} c_5 & j_{56} = s_1 s_4 s_5 + c_1 c_{23} c_4 s_5 + c_1 s_{23} c_5 & j_{66} = c_{23} c_5 - s_{23} c_4 s_5
\end{array}$$

where j_{pk} denotes the element of the pth row and kth column of $\boldsymbol{J}$. Notice that some elements of $\boldsymbol{J}$ dependent on some of the others, due to the influence of the angular velocity with respect to the wrist over the whole linear velocity. Hence, the order of calculation must be taken in consideration.

6.1.3 Singularities of PA10-6C

The inverse kinematic relationship, mapping the end-effector velocities $\dot{\mathbf{x}}$ given in the working space (Cartesian space) into the corresponding joint velocities $\dot{\boldsymbol{\theta}}$, is written as:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^{-1} \dot{\mathbf{x}} \tag{6.20}$$

A singular configuration, also known as singularity, is a configuration of the robot's joints at which the end-effector mobility – defined as the rank of the Jacobian matrix – locally decreases, i.e., it is then not possible to move or exert force in certain directions in the Cartesian space. In the neighborhood of singularity, small velocities in the operational space may cause excessive high velocities in the joint space, thus producing an acute behavior of the robot. In the PA10-6C robot manipulator, three types of singularities are possible (see Figure 6.3):

- *Arm-extended singularity* (θ_3=0): The robot reaches the end of its regional workspace, i.e., the position that the wrist centre point can reach by moving the first three joints. As the name suggests, this occurs when the elbow of the robot is fully extended.
- *Wrist-extended singularity* (θ_5=0): Joints θ_4 and θ_6 are collinear, so they span the same motion freedom. Hence, the angular velocity about the common normal of the three wrist joints is lost.
- *Wrist-above-shoulder singularity* (d^h=0): The wrist centre point intersects Z^1 axis. Infinitely solutions of $\theta_1$exist for the inverse kinematics.

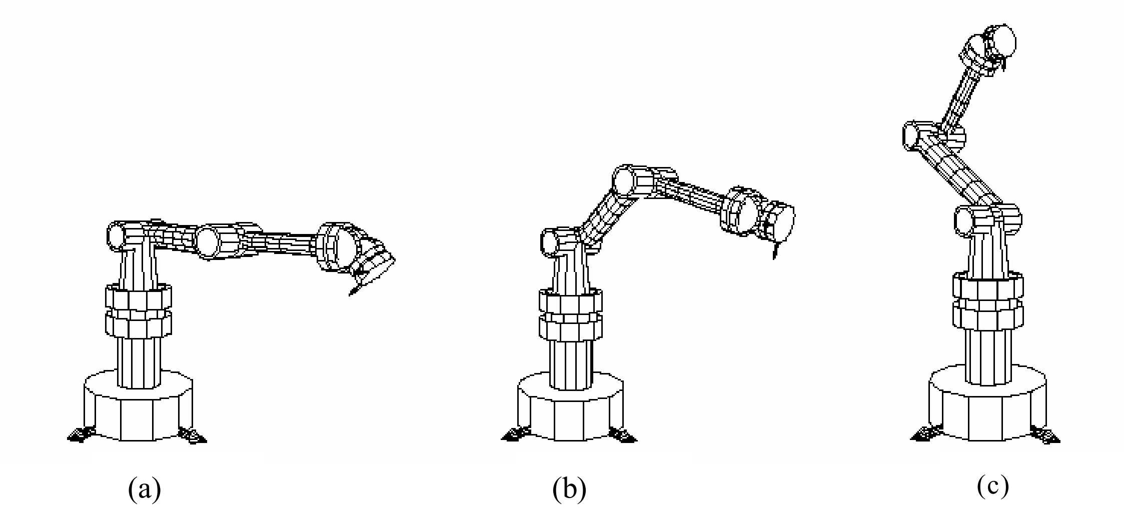


Figure 6.3.Singular configurations: (a) arm-extended, (b) wrist-extended, (c) wrist-above-shoulder [15]

A *singular configuration* is easily detectable, since in such a case $\det \boldsymbol{J} = 0$. The decoupling condition of the kinematic structure of the PA10-6C robot can be taken in advantage the determination of singularities. From the fact that the determinant of the Jacobian is independent of the reference frame with respect to which it is calculated, it results convenient to calculate the Jacobian with respect to wrist centre, frame {4}. When expressed in this frame, the spherical wrist does not generate translational components. Consequently, the lower right most block of the Jacobian corresponding to the linear velocity influence of the last three joints is equal to zero. Getting a Jacobian matrix of the form:

$$
{}^{4}\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} \\ \boldsymbol{J}_{21} & \boldsymbol{0}_{3\times 3} \end{bmatrix} \tag{6.21}
$$

This situation make possible to decouple the singularity problem into two simpler problems: the *wrist singularity* problem ($\det\boldsymbol{J}_{12} = 0$), and the *arm singularity* problem, ($\det\boldsymbol{J}_{21} = 0$). Hence, the PA10-6C robot arm presents any singular configurations if and only if:

$$
\det \boldsymbol{J} = \det \boldsymbol{J}_{12} \det \boldsymbol{J}_{21} \tag{6.22}
$$

From equation (6.12), the angular velocity generated by the last three joints with respect to the wrist centre, frame {4}, is already known. The angular velocity of the first three joints with respect to the frame {4} is easily deduced by inspection of Figure 6.2, and the linear velocity of the wrist with respect to the base expressed in the frame {4} is obtained by premultiplying ${}^{3}\boldsymbol{R}_{1}$ to the equation (6.18). Finally, the expression for the ${}_{4}\boldsymbol{J}$ states as follows:

$$
{}^{4}J = \begin{bmatrix} 0 & 1 & 1 & 0 & -c_4 & -s_5 s_4 \\ -s_{23} & 0 & 0 & 0 & -s_4 & s_5 c_4 \\ c_{23} & 0 & 0 & 1 & 0 & c_5 \\ -d^h & 0 & 0 & 0 & 0 & 0 \\ 0 & l_2 c_3 + l_3 & l_3 & 0 & 0 & 0 \\ 0 & l_2 s_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.23}
$$

Hence,

$$\det {}^4J = \det\left(\begin{bmatrix} -d^h & 0 & 0 \\ 0 & l_2c_3 + l_3 & l_3 \\ 0 & l_2s_3 & 0 \end{bmatrix}\right)\det\left(\begin{bmatrix} 0 & -c_4 & -s_5s_4 \\ 0 & -s_4 & s_5c_4 \\ 1 & 0 & c_5 \end{bmatrix}\right) \tag{6.24}$$
$$= -d^h l_2 l_3 s_3 s_5$$

It can be concluded from equation (6.24) that the geometry of the PA10-6C robot brings a lot of simplicity in the determination of singularities.

It is important to notice that at singular configuration, there exists a singular direction in which movement of the end-effector becomes unfeasible. The singular direction at wrist singularity is illustrated in Figure 6.4. Any attempt to move the end-effector along this direction is physically impossible. On the contrary, the end-effector can freely move along the plane orthogonal to it. Thus, in order to escape from singularity in a specific direction, the singular direction must be orthogonal to it. If not the case, this can be forced by considering the robot as a redundant mechanism in the subspace orthogonal to the singular direction of the end-effector and creating a null space motion- movement of some joints of the robot such that the position and orientation of the end-effector is not affected (see Figure 6.4).

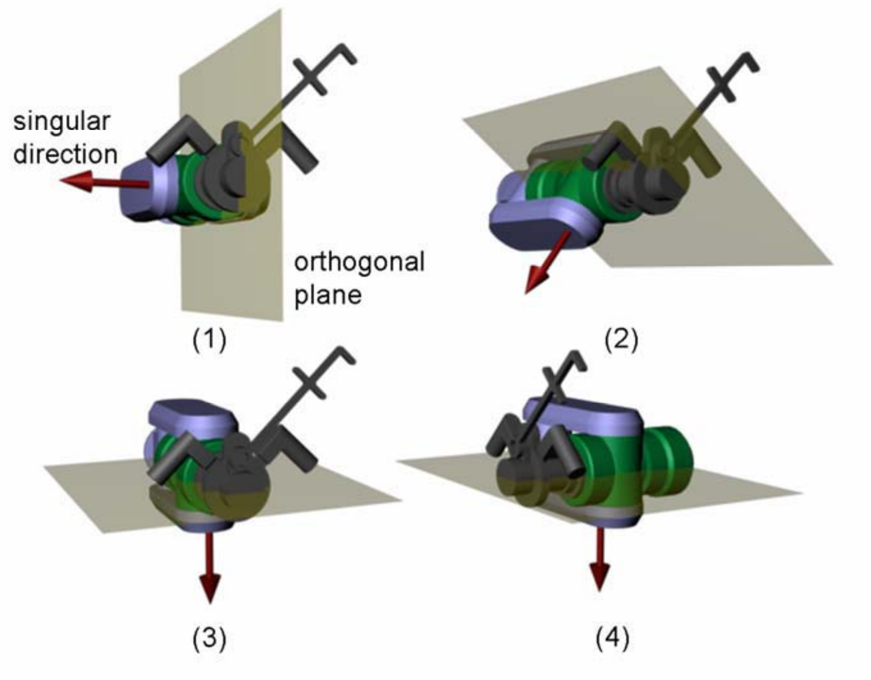


Figure 6.4. Singular direction and its orthogonal plane at wrist singularity

6.2 Differential kinematics inversion

For most manipulators, a closed-form inverse kinematic function does not exist at the position level. As a result, inverse kinematics is usually carried out at the velocity or acceleration level. The remaining of this chapter presents two different strategies to solve the differential kinematics inversion which are robust at singular configuration: the Damped Least Squares and the adjoint Jacobian inversion approach. The two methods are compared between each other. Finally, their utilization within the modiCAS system for cooperative tasks is discussed based on the experimental results.

6.3 Damped least squares approach

The most common method for handling singularities is the Damped-Least-Squares (DLS) method, proposed independently in [92] and [140]. This method is a local optimization method that makes a trade-off between the accuracy and feasibility of the inverse kinematics solution to prevent excessively high joint velocities by using a damping factor. However, at singular configuration and its neighborhood the accuracy of the inverses kinematic solution has to be sacrificed in order to achieve feasibility. [34], especially when the command velocity vector points along the singular direction. Concerning the cooperative mode, this would mean that if a force is applied in the singular direction, the corresponding joint motion would degenerate having a deviation error in the end-effector movement.

This method uses an instantaneous trade-off between the accuracy and feasibility of the inverse kinematic solution to prevent the joint velocities from becoming excessively high. The trade-off is quantified by a factor known as the damping factor. The DLS method can be theoretically justified as follows [18]. Instead of just finding the minimum vector $\dot{\boldsymbol{\theta}}$ that gives the best solution, the DLS find the value of $\dot{\boldsymbol{\theta}}$ that minimizes the expression

$$\left\| \boldsymbol{J}\dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right\|^2 + \lambda \left\| \dot{\boldsymbol{\theta}} \right\|^2 \tag{6.25}$$

where $\lambda \in \Re$ is a non-zero damping constant. This can be equivalent rewritten as:

$$\left(\boldsymbol{J}^T \boldsymbol{J} + \lambda \boldsymbol{I} \right) \dot{\boldsymbol{\theta}} = \boldsymbol{J}^T \dot{\mathbf{x}} \tag{6.26}$$

where $\left(\boldsymbol{J}^T\boldsymbol{J}+\lambda\boldsymbol{I}\right)$ is non-singular. Finally, the damped least squares solution states as:

$$\dot{\boldsymbol{\theta}}=\boldsymbol{J}^*\dot{\mathbf{x}}=(\boldsymbol{J}^T\boldsymbol{J}+\lambda\boldsymbol{I})^{-1}\boldsymbol{J}^T\dot{\mathbf{x}} \tag{6.27}$$

where $\boldsymbol{J}^*$ is denote as the SR-inverse of $\boldsymbol{J}$. The damping factor λ renders the inversion better conditioned from a numerical viewpoint. Note that $\left(\boldsymbol{J}^T\boldsymbol{J}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{J}^T=\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T+\lambda\boldsymbol{I}\right)^{-1}$. Thus, expression (6.27) can be reformulated as:

$$\dot{\boldsymbol{\theta}}=\boldsymbol{J}^*\dot{\mathbf{x}}=\boldsymbol{J}^T(\boldsymbol{J}^T\boldsymbol{J}+\lambda\boldsymbol{I})^{-1}\dot{\mathbf{x}} \tag{6.28}$$

Expression (6.29) has the advantage over equation (6.28) that its computation is less expensive, since $\left(\boldsymbol{J}^T\boldsymbol{J}+\lambda\boldsymbol{I}\right)^{-1}\in\Re^{n\times n}$ while $\left(\boldsymbol{J}\boldsymbol{J}^T+\lambda\boldsymbol{I}\right)^{-1}\in\Re^{m\times m}$, and $m\leq n$. Also note in equation (6.28) that when rank of $\boldsymbol{J}$ is m,

$$\lambda=0\rightarrow\boldsymbol{J}^*=\boldsymbol{J}^T(\boldsymbol{J}^T\boldsymbol{J})^{-1}=\boldsymbol{J}^\# \tag{6.29}$$

i.e. when there is no damping, the DLS reduces to the pseudoinverse.

6.3.1 Singular value decomposition of the damped least squares

The singular value decomposition (SVD) can be used to analyses the DLS solution (see Appendix D). From equation (6.28), the matrix $\boldsymbol{J}\boldsymbol{J}^T+\lambda\boldsymbol{I}$ expressed with SVD leads:

$$\begin{aligned}\boldsymbol{J}\boldsymbol{J}^T+\lambda\boldsymbol{I}&=(\boldsymbol{U\Sigma V}^T)(\boldsymbol{V\Sigma}^T\boldsymbol{U}^T)+\lambda\boldsymbol{I}\\&=\boldsymbol{U}(\boldsymbol{\Sigma\Sigma}^T+\lambda\boldsymbol{I})\boldsymbol{U}^T\end{aligned} \tag{6.30}$$

where $\boldsymbol{\Sigma\Sigma}^T+\lambda\boldsymbol{I}$ is a non-singular diagonal matrix with its diagonal entries as $\rho_i^2+\lambda$, where ρ_i denotes the singular values if $\boldsymbol{J}$. Then the SR-inverse $\boldsymbol{J}^*$ expressed using SVD is equal to

$$\begin{aligned}\boldsymbol{J}^*&=\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T+\lambda\boldsymbol{I})\\&=(\boldsymbol{V\Sigma}^T(\boldsymbol{\Sigma\Sigma}^T+\lambda\boldsymbol{I})^{-1}\boldsymbol{U}^T)\\&=\boldsymbol{V\Sigma}^*\boldsymbol{U}^T\end{aligned} \tag{6.31}$$

where

$$\boldsymbol{\Sigma}^* = \begin{bmatrix} \frac{\rho_1}{\rho_1^{\ 2}+\lambda} & & & 0 \\ & \frac{\rho_2}{\rho_2^{\ 2}+\lambda} & & \\ & & \ddots & \\ 0 & & & \frac{\rho_n}{\rho_n^{\ 2}+\lambda} \end{bmatrix} \tag{6.32}$$

6.3.2 The damping factor

The DLS presents a continuous and feasible solution at singularities and their neighborhood. However, away from the singularity an exact solution is still desired i.e. $\lambda \to 0$. A number of methods to determine the damping factor have been proposed in the literature [34], which compute the damping factor λ based on some Jacobian-dependent measure such as the manipulability measure, condition number or the minimum singular value which indicates the closeness of the manipulator to a singular configuration. Nakamura [93] suggested adjusting the damping factor according to the value of the manipulability measure *w*:

$$w = \sqrt{\det(JJ^T)} \tag{6.33}$$

Using a threshold value w_t. *w* is a nonnegative measure which becomes zero at a singular configuration. Thus, λ is computed as [93]:

$$\lambda = \begin{cases} \lambda_0 \left(1 - \frac{w}{w_t}\right)^2, & w < w_t \\ 0, & \textit{otherwise} \end{cases} \tag{6.34}$$

where λ_0 is the scale factor at singular points. So, no damping is applied when the value of *w* is greater than w_t until it reaches its maximum value λ_0 at *w*=0.

6.4 Adjoint Jacobian approach

The adjoint Jacobian approach permits to move the robot at and in the neighborhood of singularities without any position or orientation error. The deterioration of motion ability at the singularity reflects on velocity only. Let us denote the end-effector as

$$\dot{\mathbf{x}} = v\mathbf{u}\,, \tag{6.35}$$

where the unit vector $\mathbf{u} \in \mathbb{R}^{n}$ denotes the end-effector instantaneous motion direction, while the scalar variable v ($v \geq 0$) stands for end-effector velocity. Substitution of equation (6.35) into (6.20) yields into

$$\dot{\boldsymbol{\theta}} = vJ^{-1}\mathbf{u} \tag{6.36}$$

Furthermore, the inverse Jacobian can be calculated by use of the adjoint Jacobian:

$$J^{-1} = \frac{1}{\det \boldsymbol{J}} adj\boldsymbol{J} \tag{6.37}$$

where det$\boldsymbol{J}$ and $adj\boldsymbol{J}$ denote the determinant and the adjoint matrix of the Jacobian, respectively.

$$adj(J) = \begin{bmatrix} \Upsilon_{11} & \cdots & \Upsilon_{1n} \\ \vdots & \Upsilon_{ij} & \cdots \\ \Upsilon_{n1} & \cdots & \Upsilon_{nn} \end{bmatrix}^{T} \tag{6.38}$$

$$\Upsilon_{ij} = (-1)^{i+j} \det \boldsymbol{J}_{ij}$$

The term J_{ij} represents the Jacobian sub-matrix obtained by deleting the i^th^ row and j^th^ column of J. By combining expression (6.36) and (6.37), the following expression is obtained:

$$\dot{\boldsymbol{\theta}} = \frac{v}{\det \boldsymbol{J}} (adj\boldsymbol{J})\mathbf{u} \tag{6.39}$$

Notice that in equation (6.39), the determinant of the Jacobian represents a scalar factor related to the magnitude of motion in the joint space, while $(adj\boldsymbol{J})v\mathbf{u}$ determines the velocity relationship between the individual joints. At singularity, $\boldsymbol{J}^{1}$ does not exist, since det$\boldsymbol{J}$=0. Moreover, in the neighborhood of singularity the determinant is almost zero, which considerably influences yielding into excessive joint velocities.

The adjoint Jacobian approach consists in modifying equation (6.39) as follows:

$$\dot{\boldsymbol{\theta}} = \sigma b\left(adj\boldsymbol{J}\right)\mathbf{u} \tag{6.40}$$

In equation (6.40) the system is decoupled in terms of velocity, represented by the scalar variable b ($b \geq 0$), and direction of motion, represented by $\sigma(adj\boldsymbol{J})\mathbf{u}$, where σ is a sign variable ($\sigma = \pm 1$). With a proper design of σ and b, it is possible to control the non-redundant robot arm at and around a singularity, without any error in the direction, and with feasible joint velocity.

6.4.1 Relationship between adjoint Jacobian and the null vector

The velocity equation (6.36) can be rewritten as

$$J\dot{\boldsymbol{\theta}} - v\mathbf{u} = 0\,, \tag{6.41}$$

which can be considered as an instantaneous-motion closure equation for the kinematic chain. A compact notation is obtained by augmenting the joint space with the trajectory variable **u**. Denote by

$$\boldsymbol{\Phi} = \left(\boldsymbol{\theta}^T, v\right)^T \tag{6.42}$$

any point in the n+1 dimensional augmented joint space. Equation (6.41) can be then rewritten as:

$$\boldsymbol{H}\left(\boldsymbol{\Phi}\right)\dot{\boldsymbol{\Phi}} = 0\,, \tag{6.43}$$

where $\boldsymbol{H} = \begin{bmatrix} J & -\mathbf{u} \end{bmatrix} \in \Re^{n\times(n+1)}$ is called the column augmented Jacobian. The vector **u** is in accordance with the forces/moments at the end-effector applied by the user. Then, one can in general assume that vector **u** is an n-dimensional parameter. The augmented Jacobian $\boldsymbol{H}$ is regarded as a nonlinear function of the joint variables $\boldsymbol{\theta}$ and the n-dimensional parameter **u** and equation (6.43) represents a nonlinear parameterized system of autonomous differential equations [44]. A general solution to equation (6.43) can be written as:

$$\dot{\boldsymbol{\Phi}} = b\overline{\mathbf{n}}_H(\boldsymbol{\Phi}) \tag{6.44}$$

where b has the same meaning as in equation (6.40) and $\overline{\mathbf{n}}_H$ represents a vector existing in the null space of $\boldsymbol{H}$. Instead of derive it as a function of the pseudoinverse $\boldsymbol{H}^+$, e.g. as a vector from the null-space projection $(\boldsymbol{I}-\boldsymbol{H}^{+}\boldsymbol{H})$, the null vector is directly determined based on Bedrossian's methodology [11]:

$$\bar{\mathbf{n}}_H(\Phi)=\begin{bmatrix}C_1 & C_2 & \cdots & C_{n+1}\end{bmatrix}^T \tag{6.45}$$
$$C_p=(-1)^{p+1}\det \boldsymbol{H}_p,$$
$$(p=1,2,\ldots,n+1),$$

where, $\boldsymbol{H}_{\mathrm{p}}$ stands for matrix $\boldsymbol{H}$ with column p removed. The determinant of a matrix can be expanded in cofactors of one of the columns. Thus, C_{p} can be expanded in cofactors of column **u:**

$$C_p=(-1)^n(u_1a_{1p}+u_2a_{2p}+\ldots+u_na_{np}) \tag{6.46}$$
$$(p=1,2,\ldots,n)$$
$$C_{n+1}=(-1)^n\det \boldsymbol{J}$$

Notice that $\bar{\mathbf{n}}_H$ is a (n+1)-dimensional vector. It can be expressed in the following form:

$$\bar{\mathbf{n}}_H(\boldsymbol{\Phi})=\begin{bmatrix}\mathbf{n}_H(\Phi)\\ \det \boldsymbol{J}\end{bmatrix} \tag{6.47}$$

where

$$\mathbf{n}_H(\boldsymbol{\Phi})=\begin{bmatrix}C_1 & C_2 & \cdots & C_n\end{bmatrix}^T \tag{6.48}$$
$$=(-1)^n\begin{bmatrix}a_{11} & a_{11} & \cdots & a_{11}\\ a_{11} & a_{11} & \cdots & a_{11}\\ & & \vdots & \\ a_{11} & a_{11} & \cdots & a_{11}\end{bmatrix}\begin{bmatrix}u_1\\ u_1\\ \vdots\\ u_n\end{bmatrix}$$
$$=(-1)^n(adj\boldsymbol{J})\mathbf{u}$$

The vector $\mathbf{n}_H$ can be regarded as a map:

$$\mathbf{n}_H(\boldsymbol{\Phi}):\Re^{n+1}\rightarrow\Re^n:\boldsymbol{\Phi}\mapsto\boldsymbol{\theta} \tag{6.49}$$

Equation (6.44) can be split into two parts to obtain joint differential motion

$$\dot{\boldsymbol{\theta}}=b\mathbf{n}_H(\boldsymbol{\Phi}), \tag{6.50}$$

and path parameter differential

$$v = b\det \boldsymbol{J} \tag{6.51}$$

Equation (6.50) shows that the adjoint Jacobian approach expressed in equation (6.40) and the null vector yield the same result in terms of direction of motion in joint space.

6.4.2 Velocity relations at singularity

The kinematic singularities can be associated with an important type of solutions of the autonomous type differential equations known as *equilibria* or fixed points, where the null space function $\mathbf{n}_H$ vanishes: $\{\boldsymbol{\theta}, \mathbf{u} : \mathbf{n}_H(\boldsymbol{\Phi}) = 0\}$ [44]. There are generally two types of equilibria [97]:

- Equilibrium I: $\exists \mathbf{u} = \bar{\mathbf{u}} : \mathbf{n}_H(\boldsymbol{\Phi}) = 0$
- Equilibrium II: $\forall \mathbf{u} : \mathbf{n}_H(\boldsymbol{\Phi}) = 0$

Equilibrium I may occur at a codimension one kinematic singularity ($rank(\boldsymbol{J}) = n-1$), while Equilibrium II occurs with a codimension larger than one ($rank(\boldsymbol{J}) < n-1$), where self-motion vanishes for all velocity vectors **u**. Further discussion concentrates on the type Equilibrium I. This means that two singularities (e.g. shoulder and elbow singularities) occurring simultaneously are not considered within the scope of this analysis and non solution to this special case is not given. However, for robots where their position and orientation sub-chains can be regarded as separate mechanisms, the singularities can be treated separately. Moreover, Tsumaki *et al.* [130] have shown that the adjoint Jacobian approach can be applied even to a 6 DOF robot arm with non-spherical wrist.

Nenchev *et al.* [99] distinguish between two types of velocity relations at kinematic singularity depending on the direction **u**. In terms of the adjoint formulation they are:

- Type-A: $\{\det\boldsymbol{J} = 0, (adj\boldsymbol{J})\mathbf{u} \neq 0)\}$
- Type-B: $\{adj\boldsymbol{J})\mathbf{u} = 0\}$

Velocity relations of type A are not equilibria, some joint angles are being affected by the Cartesian space velocity vector **u**. This represents the self-motion condition where the end-effector velocity is zero while some of the intermittent links are moving. On the other hand, in Type-B relationship, all components of $adj\boldsymbol{J}$ vanish and motion would stop entirely. This relation represents equilibrium of the first type of equation (6.40). Notice that the velocity relation can generally be of either type-A or type-B, depending on the direction of the velocity vector **u**, and the codimension of the singularity. Since in this work only codimension one singularity is assumed, the type of relation will depend entirely of the velocity vector **u**. Distinguishing between type-A and type-B velocity relations can be easily achieved. The det***J*** is used to detect the type-A velocity relation, whereas the norm of the null space function (6.48) is used to detect the type-B relation.

A thoroughly analysis of kinematic singularities in the context of the relationship between differentials of motion and kinematic singularities can be found in [67], [11], [10], [96] and [114].

6.4.3 Selection of scalar variable *b*

At singularity, the determinant of the Jacobian become zero, and for the calculation of joint velocities using equation (6.40), $adj\boldsymbol{J}$ is divided by zero. Also near singularities, division by an almost zero number results in excessive joint velocities. To overcome this problem, the joint velocity can be restricted by using the norm of the joint velocity vector together with the following condition:

$$b = \begin{cases} \dfrac{v}{|\det \boldsymbol{J}|} & \|\dot{\boldsymbol{\theta}}\| \le \dot{\theta}_{\max} \dfrac{v}{v_{\max}} \\ \dfrac{\dot{\theta}_{\max}}{v_{\max}} \dfrac{v}{\|(adj\boldsymbol{J})\mathbf{u}\|} & otherwise \end{cases}, \tag{6.52}$$

where $\dot{\theta}_{\max}$ and $v_{\max}$ are user defined restriction on the joint velocity and the end-effector velocity, respectively. This condition is used to smoothly change between two possible values of *b* from equation (6.40). Notices that when the joint velocity norm is smaller as the condition threshold, the following expression is obtained:

$$\begin{aligned} \dot{\boldsymbol{\theta}} &= \sigma \frac{v}{\det \boldsymbol{J}} (adj\boldsymbol{J})\mathbf{u} \\ &= \boldsymbol{J}^{-1}\dot{\mathbf{x}} \end{aligned} \tag{6.53}$$

Equation (6.53) is the same solution as the one expressed in equation (6.39). On the other hand, if the norm value is larger than the boundary condition, equation (6.40) becomes:

$$\dot{\boldsymbol{\theta}} = \sigma v \frac{\dot{\theta}_{\max}}{v_{\max}} \frac{(adj\boldsymbol{J})\mathbf{u}}{\left\|(adj\boldsymbol{J})\mathbf{u}\right\|} \tag{6.54}$$

Calculating the scalar variable b in this way is useful when approaching any singularity, and when having velocity relation of type A at singularity, i.e. when doing self-motion at singularity, since $(adj\boldsymbol{J})\mathbf{u} \neq 0$ even if $\det \boldsymbol{J} \to \boldsymbol{0}$. However, with velocity relation of type B, $\det \boldsymbol{J} \to \boldsymbol{0}$ and $(adj\boldsymbol{J})\mathbf{u} \to 0$, i.e. equation (6.40) becomes indefinite at the singularity. Thus, this particular case must be treated separately. An important assumption for this case is that the determinant of the Jacobian is factorized. Thus, it can be expressed as a product of terms,

$$\det \boldsymbol{J} = f_1 \cdot f_2 \cdots f_k, \tag{6.55}$$

where each term corresponds to one of the different singularities of the robot. The subscript k is the number of possible singularities. Using expression (6.55), the joint velocity equation (6.39) can be represented as

$$\dot{\boldsymbol{\theta}} = \frac{f_i (adj_i \boldsymbol{J})\dot{\mathbf{x}} + (\overline{adj_i}\boldsymbol{J})\dot{\mathbf{x}}}{f_1 \cdots f_i \cdots f_k}, \tag{6.56}$$

where $adj_i\boldsymbol{J}$ contains all the elements of the adjoint Jacobian that are affected by f_i, and the rest of it entries are zero. The $\overline{adj_i}\boldsymbol{J}$ contains all terms that are not included into $adj_i\boldsymbol{J}$. Notice that in case of velocity type B relation, $\overline{adj_i}\boldsymbol{J}$ becomes zero. Then, the inverse Jacobian may be reformulated as follows:

$$\begin{aligned} \dot{\boldsymbol{\theta}} &= \frac{f_i (adj_i \boldsymbol{J})\dot{\mathbf{x}}}{f_1 \cdots f_i \cdots f_k} \\ &= \frac{(adj_i \boldsymbol{J}) v \mathbf{u}}{f_1 \cdots f_k} \end{aligned} \tag{6.57}$$

Equation (6.57) can be rewritten in the same context as of equation (6.40) as:

$$\dot{\boldsymbol{\theta}} = \sigma b(adj_i\boldsymbol{J})\mathbf{u} \tag{6.58}$$

$$b = \frac{v}{|f_1 \cdots f_k|}$$

In equation (6.58), the singular component is factorized out from the numerator and canceled out with the one from denominator, thus no division by zero will occur and there will be no effect of the singular configuration. The robot can then move out of the singular configuration without any position or orientation error.

6.4.4 Selecting sign variable σ

The sign variable σ will affect the final direction of the joint velocity command. Out of singularities, σ agrees with the sign of the determinant of the Jacobian:

$$\sigma = \operatorname{sgn}(\det \boldsymbol{J}) \tag{6.59}$$

Note that when moving through a singularity, the sign of the determinant changes. Furthermore, at singularity, equation (6.59) is undefined, since det$\boldsymbol{J}$=0, i.e. there is no possibility to directly determinate the value of σ. Therefore, special care must be taken in order to preserve the correct joint velocity direction and agree with the direction of the applied forces at the hands-on interface.

6.4.5 Wrist singularity

The wrist singularity is of our main concern under the assumption that the virtual constraints limit the working area. Thus, elbow and overhead are practically eluded. On the other hand, the wrist singularity can occur at any time within the working area.

The wrist singularity can be simple analyzed by regarding the orientation kinematic sub-chain as an independent subsystem. Therefore, the differential kinematic relationship stated in equation (6.12) is used:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 & -c_4 & -s_4 s_5 \\ 0 & -s_4 & c_4 s_5 \\ 1 & 0 & c_5 \end{bmatrix} \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{6.60}$$

Notice that the Jacobian matrix of equation (6.60) is equal to $\boldsymbol{J}_{12}$ (see section 6.1.3), which is the one degenerating at wrist singularity. The determinant and the adjoint Jacobian are derived as

$$\det \boldsymbol{J}_{12} = -s_5, \tag{6.61}$$

and

$$adj\boldsymbol{J}_{12} = \begin{bmatrix} -s_4c_5 & c_4c_5 & -s_5 \\ c_4s_5 & s_4s_5 & 0 \\ s_4 & -c_4 & 0 \end{bmatrix}, \tag{6.62}$$

respectively. At wrist singularity $s_5 = 0$. The above expression can be split according to expression (6.56) as:

$$f_i adj_i \boldsymbol{J}_{12} = s_5 \begin{bmatrix} 0 & 0 & -1 \\ c_4 & s_4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.63}$$

and

$$\overline{adj}_i \boldsymbol{J}_{12} = \begin{bmatrix} -s_4c_5 & c_4c_5 & 0 \\ 0 & 0 & 0 \\ s_4 & -c_4 & 0 \end{bmatrix} \tag{6.64}$$

Substituting equation (6.62) into equation (6.40) and assuming that $adj\boldsymbol{J}_{12} \neq 0$ and $s_5 = 0$, yields into:

$$\begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = \sigma b \begin{bmatrix} -c_5(s_4u_x - c_4u_y) \\ 0 \\ s_4u_x - c_4u_y \end{bmatrix}, \tag{6.65}$$

$$b = \frac{v}{\left|\det \boldsymbol{J}_{12}\right|} = \frac{v}{|s_5|}$$

which corresponds to the behavior of the system at wrist singularity with velocity relation of type-A. The above expression produces the so called self-motion.

During a velocity relation of type-B, $adj\boldsymbol{J}_{12} = 0$, which implies that $\overline{adj_i}\boldsymbol{J}_{12} = 0$. If the determinant factor f_i is canceled out and using $adj_i\boldsymbol{J}_{12}$ only, as stated in equation (6.58), the final expression becomes:

$$\begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = \sigma b \cdot s_5 \begin{bmatrix} -u_z \\ c_4 u_x + s_4 u_y \\ 0 \end{bmatrix} \tag{6.66}$$

$$b = \frac{v}{\left|\det \boldsymbol{J}_{12}\right|} = \frac{v}{\left|s_5\right|}$$

Two types of motions can be observed in equation (6.66) depending on **u**: Escape/through motion ($u_z = 0$), and boundary motion ($u_x = 0, u_y = 0$). The former will take the wrist out of singularity, while the latter rotates the whole wrist maintaining the singularity configuration.

Calculation of the sign variable σ when crossing singularity with velocity relation of type-B is done by comparing the direction of the applied force in relation with the direction orthogonal to singular direction.

6.5 Experimental results

The main objective of this section is to analyze the behavior of both DLS and adjoint Jacobian approach in the context of human-robot cooperation using virtual fixtures when dealing with singular configurations. Only the wrist singularity is studied, since, as it has already been pointed out, contrary to shoulder and boundary singularities, the wrist singularity may appear inside a virtual constrained working space during cooperative operation. Three different situations are distinguished: (a) Passing through singularity, (b) Escaping from singularity along singular direction, and (c) Passing near singularities. During the experiments, the path is virtual constrained to allow applied forces only in the desired direction. The evaluation criterion is basically based on the Cartesian deviation from the subspace of preferred directions *U*.

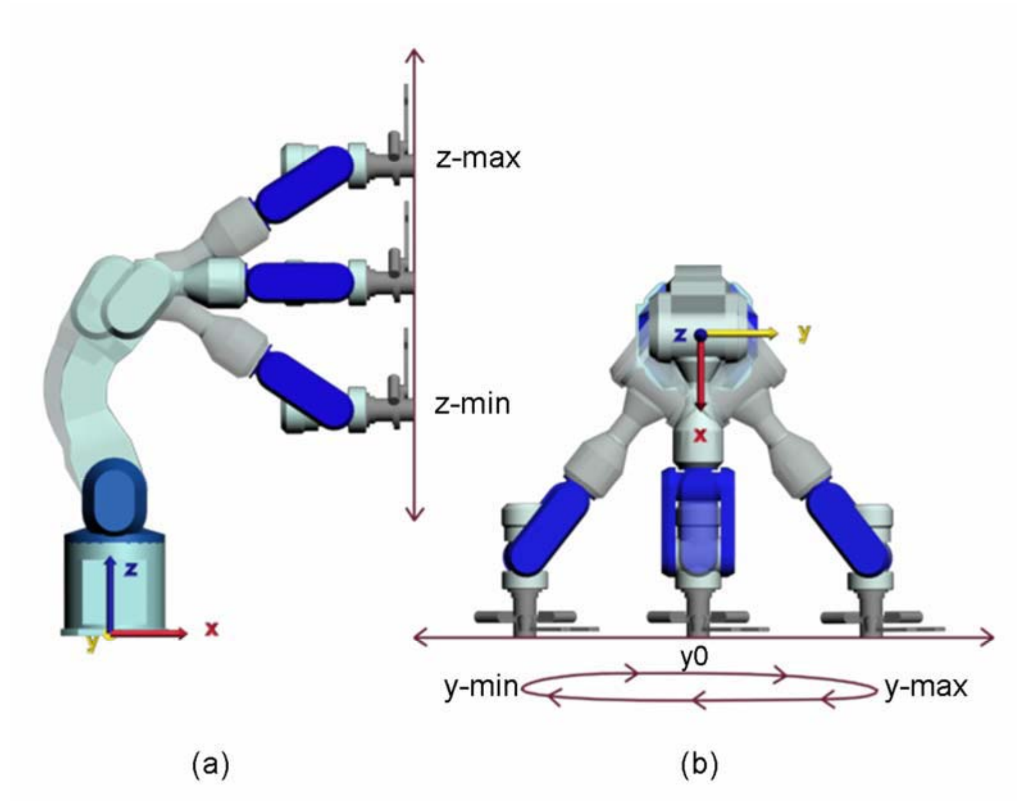


Figure 6.5. End-effector movement along (a) direction orthogonal to singular direction, and (b) singular direction

6.5.1 Passing through singularity

The first experiment consists of movements of the end-effector along Z-axis with respect to the base reference frame in such a way that passing through singularity occurs. Therefore, the robot initial position, defined in the joint space, equal to $\boldsymbol{\theta}_0 = \{0, 15, 100, 0, -25, 0\}$ (deg) is used. The corresponding Cartesian pose is defined as $\boldsymbol{T}_{TAR}$. Then, a translational VF along Z-axis is created, i.e. $S^l = \{\boldsymbol{R}_{TCP}{}^T \mathbf{l}_1\}$ and $S^\alpha = \{0\}$, where $\mathbf{l}_1 = [0 \quad 0 \quad 1]^T$. The end-effector is commanded by hand up and down so long as necessary to pass through singular configuration, repeating this movement several times.

Even though, passing through singularity presents not complications singularity for any of both approaches, DLS and adjoint Jacobian, plots of the actual Cartesian position and orientation (Figure 6.6 and Figure 6.7, respectively) reveal that the DLS produces larger deviations from the constrained path than the adjoint Jacobian.

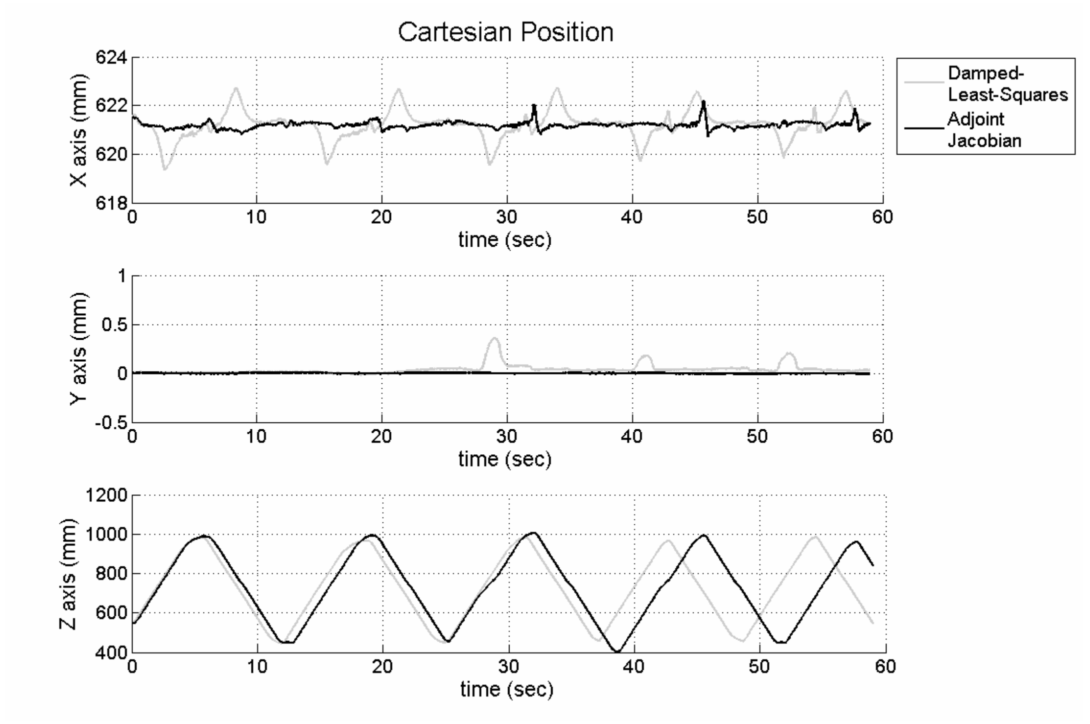


Figure 6.6. End-effector position while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame

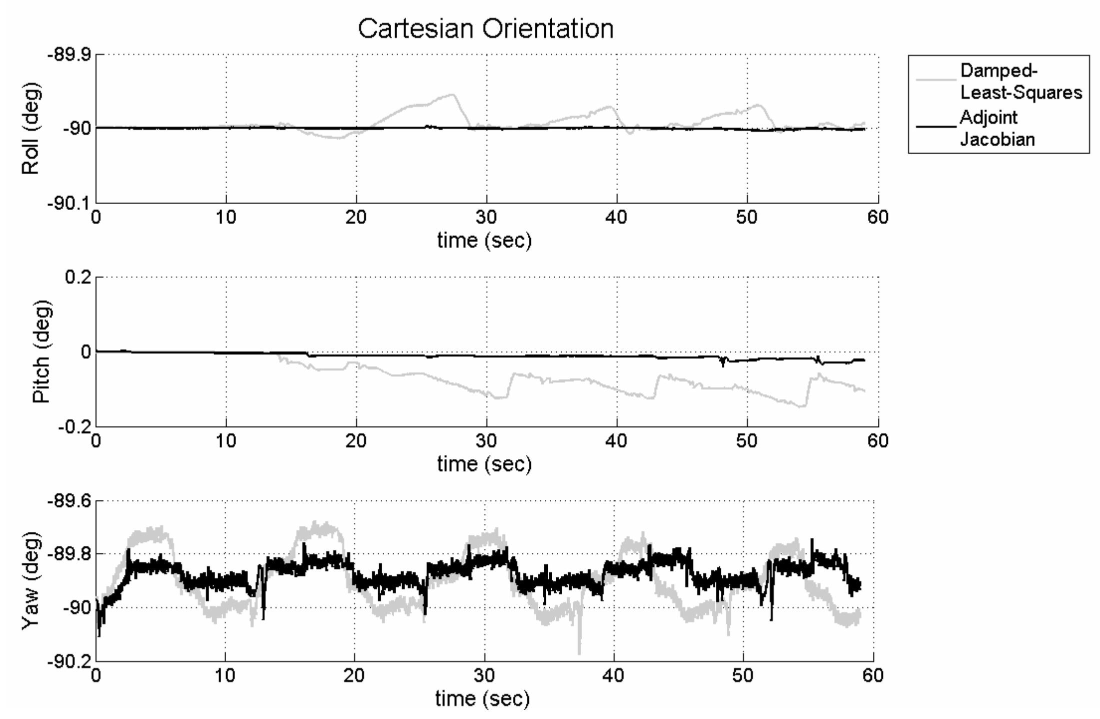


Figure 6.7. End-effector orientation while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame

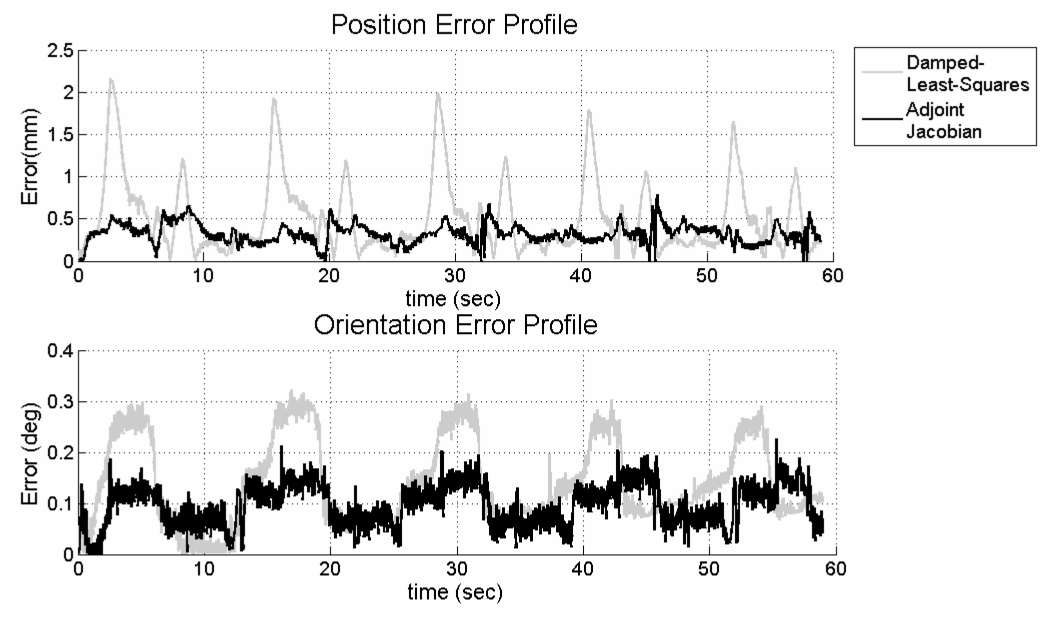


Figure 6.8. End-effector error profile while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame

Figure 6.8 shows the instantaneous norm of the error for both position and orientation in the Cartesian space. In both approach, crossing through singularity represents a critical point affecting the error behavior. Nevertheless, the adjoint Jacobian reduces the produced error when compared with the DLS. Table 6.1 presents maximal and mean error produced during the experiments.

Table 6.1. Maximum and mean error when passing through wrist singularity

Singularity Robust approach	Maximum Position error (mm)	Mean Position error (mm)	Maximum Orientation error (deg)	Mean Orientation error (deg)
DLS	2.1651	0.5029	0.3226	0.1386
adjoint Jacobian	0.7864	0.3189	0.2263	0.0957

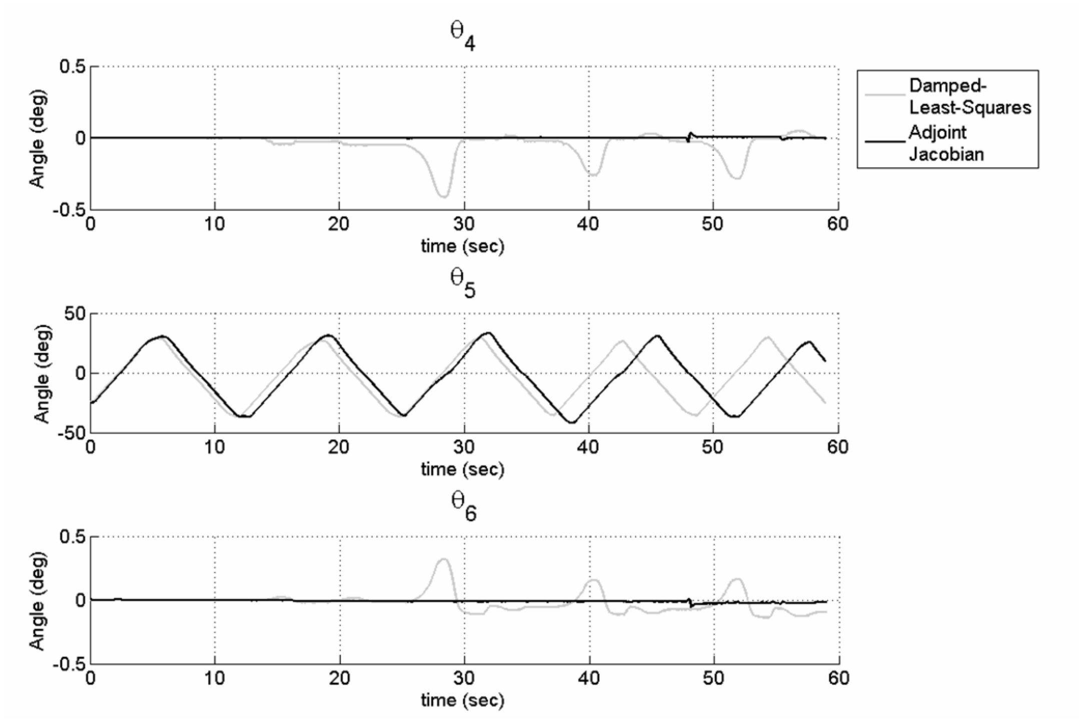


Figure 6.9. Position of joints θ_4, θ_5 and θ_6 while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame

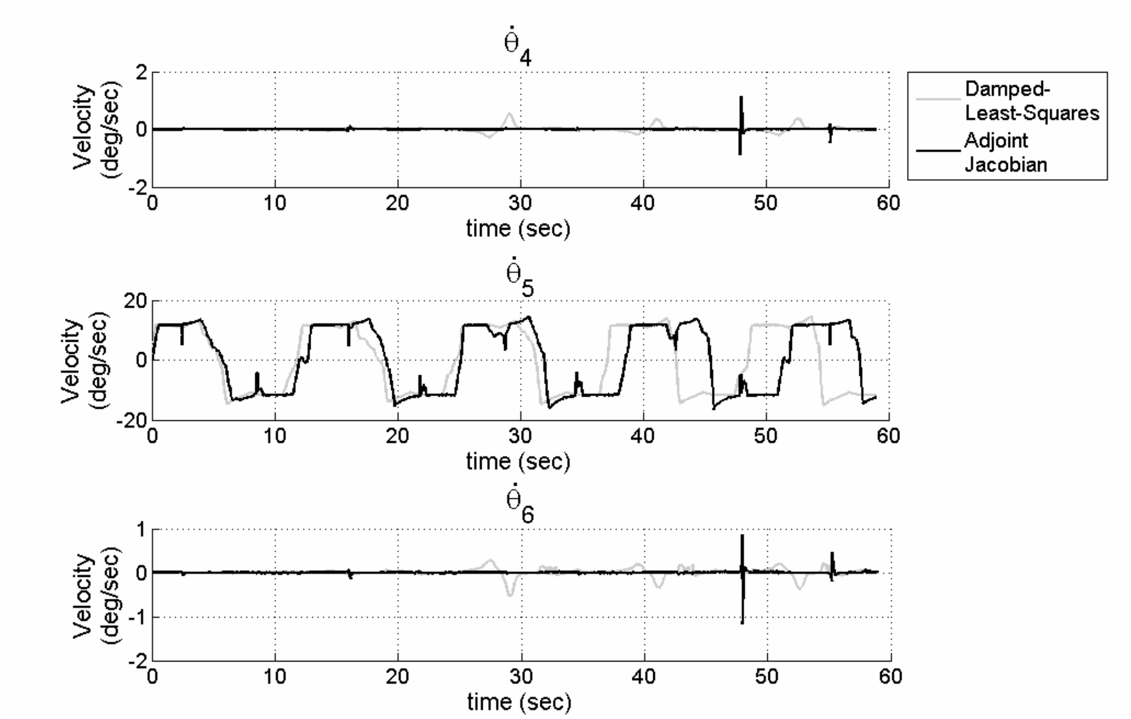


Figure 6.10. Velocity of θ_4, θ_5 and θ_6 while passing through singularity with virtual fixture along Z-axis w.r.t. the base frame

It is worth to point out the behavior of the robot on the joint space under the influence of both strategies. Figure 6.9 and Figure 6.10 show the position and velocity, respectively, of the last three joints (θ_4, θ_5 and θ_6) which are the critical ones when passing through wrist singularity. The joint perturbation at singularity is notably stronger under DLS as under adjoint Jacobian. On the other side, joint velocities present a discontinuity at singularity (see Figure 6.10). This occurs from switching between control strategy for velocity relations of Type-A and Type-B. Although the discontinuity can be notice by the user during cooperative operation, such discontinuity does not produce significant deviation in the Cartesian space, since this occurs in the null space.

6.5.2 Escaping singularity along singular direction

It has already been pointed out that the singular direction at singularity is actually an unfeasible direction along which it is physically impossible for the robot to exert a movement. One possible solution is the so called null space motion to relocate the robot joints in such a way that the singular direction becomes orthogonal to the desired movement. The main objective of this experiment is to analyze the behavior of both strategies when trying to escape singularity along singular direction. The initial joint position is $\boldsymbol{\theta}_0 = \{0,10,80,0,0,0\}$, which corresponds to the y_0 of Figure 6.5.b, where the singular direction is parallel to the Y-axis with respect to the base frame. A VF is defined with $\boldsymbol{T}_{TAR}$ equal to the Cartesian pose corresponding to $\boldsymbol{\theta}_0$ and a translational VF along Y-axis, i.e. $S^l = \{\boldsymbol{R}_{TCP}{}^T \mathbf{l}_1\}$ and $S^\alpha = \{0\}$, where $\mathbf{l}_1 = [0 \quad 1 \quad 0]^T$. Thus, movements back and fort along the virtual constrained path require, first of all, that the robot escapes from singularity and afterwards passes through singularity in a similar way as in the last experiment.

The behavior of the end-effector in the Cartesian space during the experiment is plotted in Figure 6.11 and Figure 6.12 for position and orientation, respectively. It can be observed at the very beginning of the plots that the DLS presents a very large deviation error in the attempt of escape from singularity. In Figure 6.13 the instantaneous error is shown: $\|\mathbf{e}_p\|_{\max} = 7.9\,(\text{mm})$ and $\|\mathbf{e}_r\|_{\max} = 24\,(\text{deg})$. On the contrary, the response of the adjoint Jacobian approach presents no significant deviations while escaping from singularity: $\|\mathbf{e}_p\|_{\max} = 0.86\,(\text{mm})$ and $\|\mathbf{e}_r\|_{\max} = 0.36\,(\text{deg})$. Notice too, that on the subsequent attempts of passing through singularity the adjoint Jacobian slows its motion while the DLS continuous with the same velocity profile.

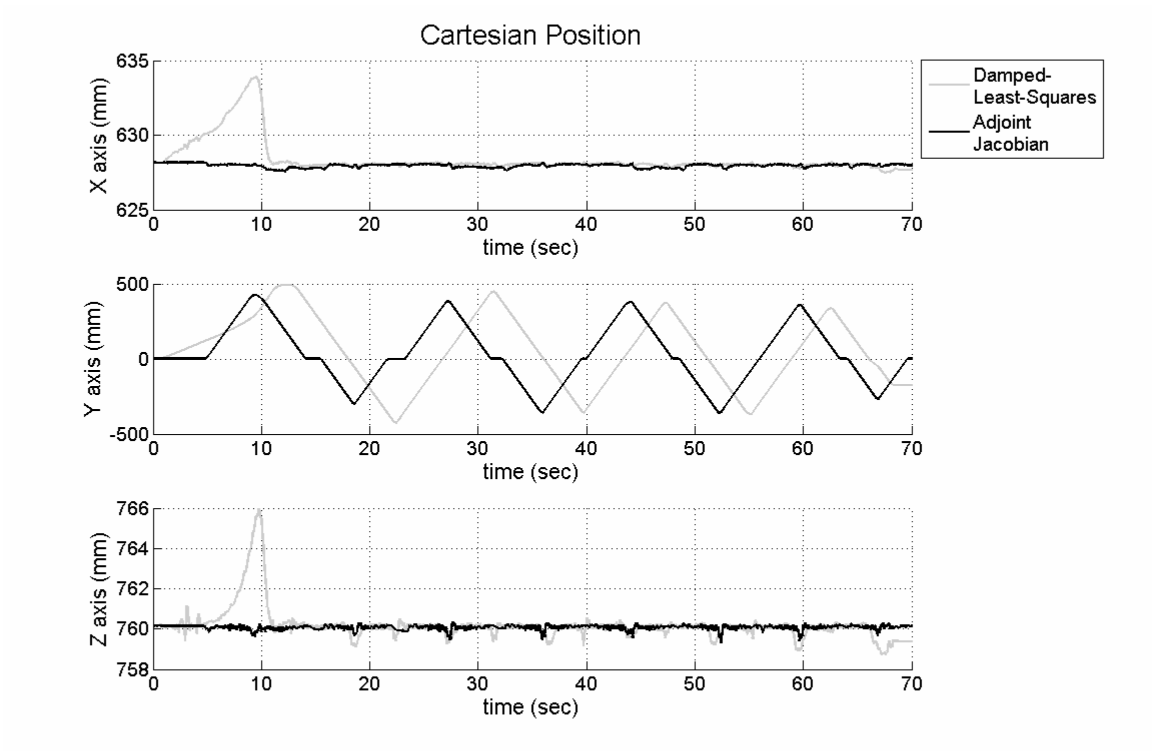


Figure 6.11. End-effector position while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

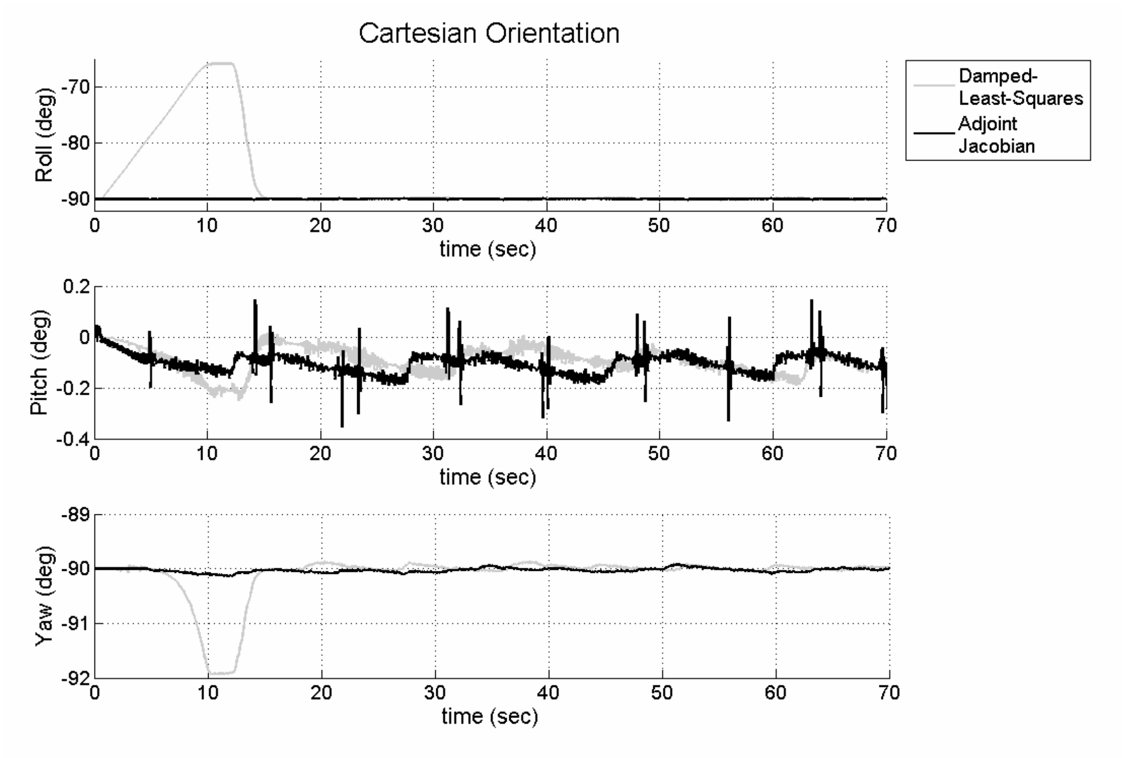


Figure 6.12. End-effector orientation while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

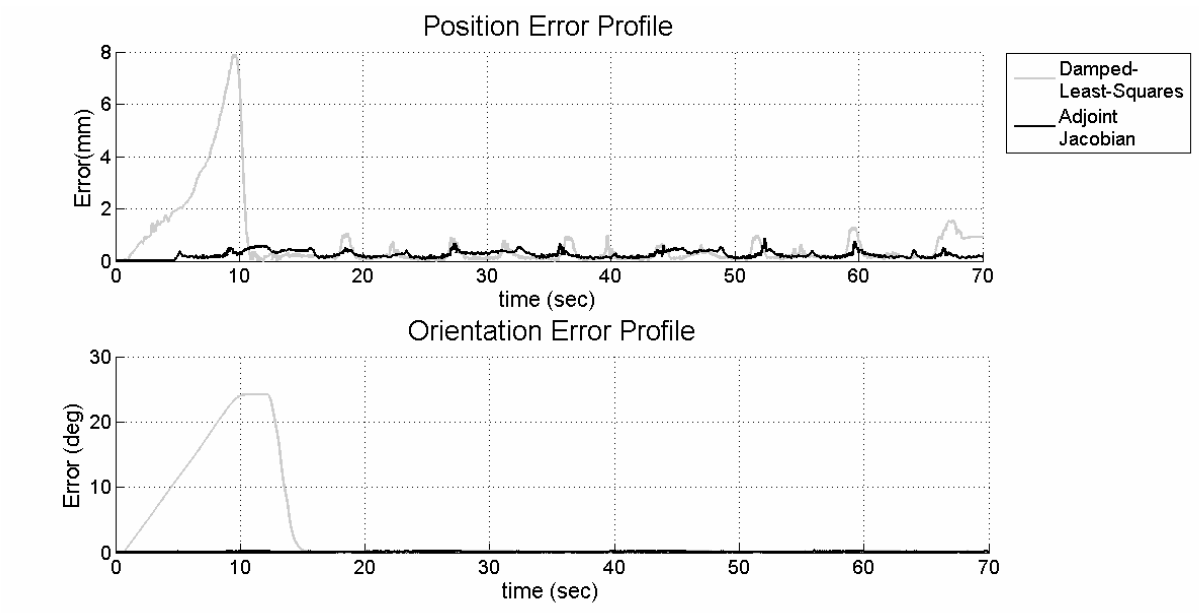


Figure 6.13. End-effector error profile while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

The results observed in the Cartesian space become clearer when analyzing the robot behavior in the joint space. Figure 6.14 shows the position of all joints. Notice that the DLS does not attempt any motion of the critical joints (θ_4, θ_5 and θ_6) when trying to escape singularity, instead, the first joint θ_1 moves until it becomes possible for the other joints to start any kind of movement, producing of course a large error in the Cartesian space. On the other hand, the adjoint Jacobian approach moves first joints θ_4 and θ_5 simultaneously and in opposite direction keeping the other joint constant in the meanwhile. This movement represents the null space motion that rotates the robot joints so that the singular directions becomes orthogonal to escaping direction, which occurs when $\theta_4 = -90$ and $\theta_5 = 90$. The null motion is the result of a velocity relation of Type-A at singularity, and only when the singular direction becomes orthogonal to the desired motion, the velocity relation of Type-B occurs, which corresponds to the escape/through motion described in section 6.4.5. In this way, the robot escapes singularity without producing any position deviation.

Now, the reduction of velocity at singularity produced with the adjoint Jacobian approach occurs due to the behavior of the robot in the joint space which is clearly observed in Figure 6.15 and Figure 6.16. A velocity discontinuity occurs just like in the last experiment, but this time the discontinuity is bigger, which can be explained from the fact that the configuration of the robot when approaching to singularity is not as straightforward as in the last experiment. Notice that joints θ_4 and θ_6 attempt to move just before getting into the singularity in the same way it does when passing near singularity, since a velocity relation of Type-A is present , but at singularity the attempt of motion is stopped and the joint are driven back until the velocity relation becomes of Type-B in order to pass through singularity. Joints θ_4, θ_5 and θ_6 are plotted once again on Figure 6.16 in order to have a closer look to observe the effect of velocity on the joint positions.

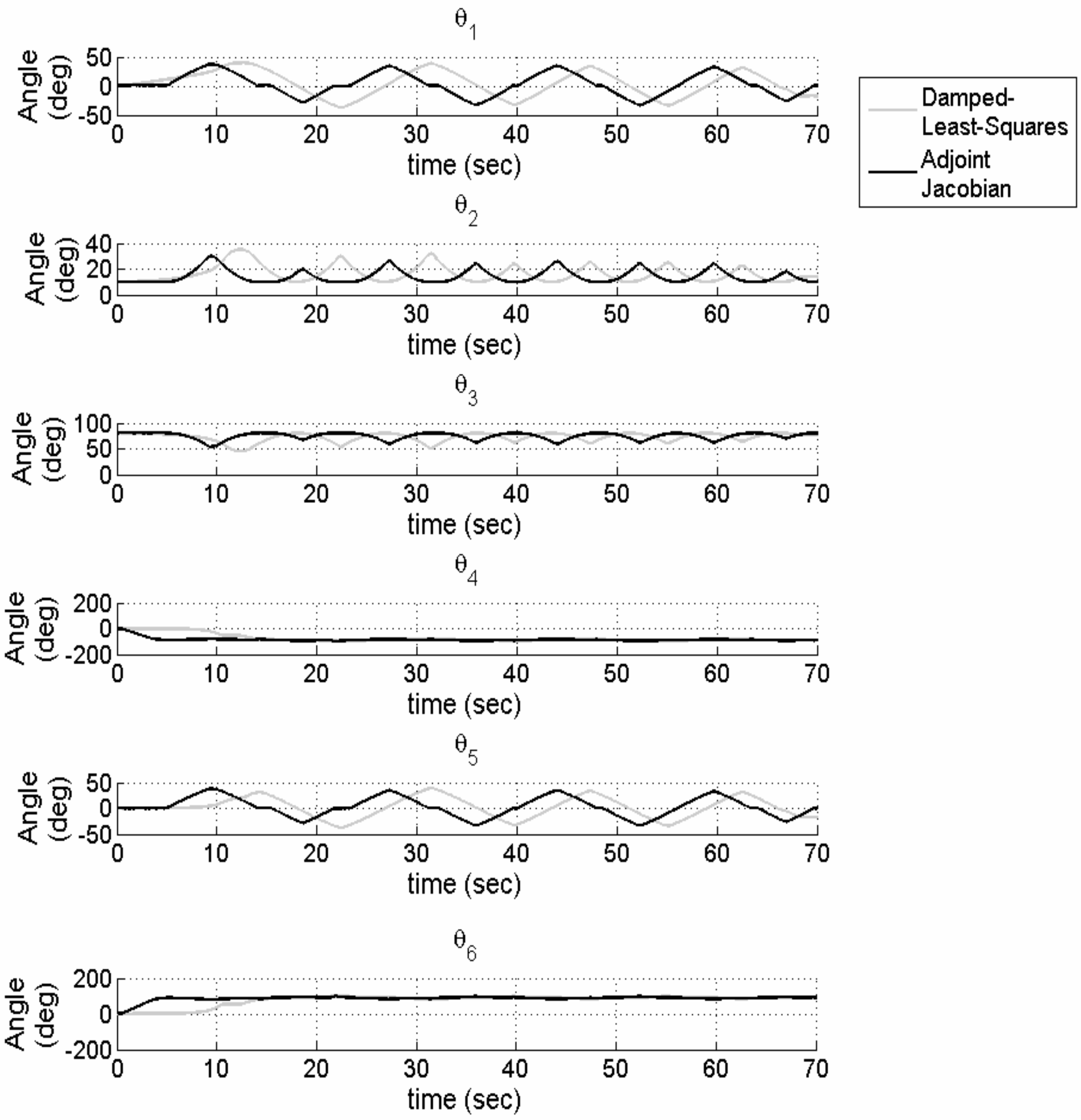


Figure 6.14. Position of robot joints while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

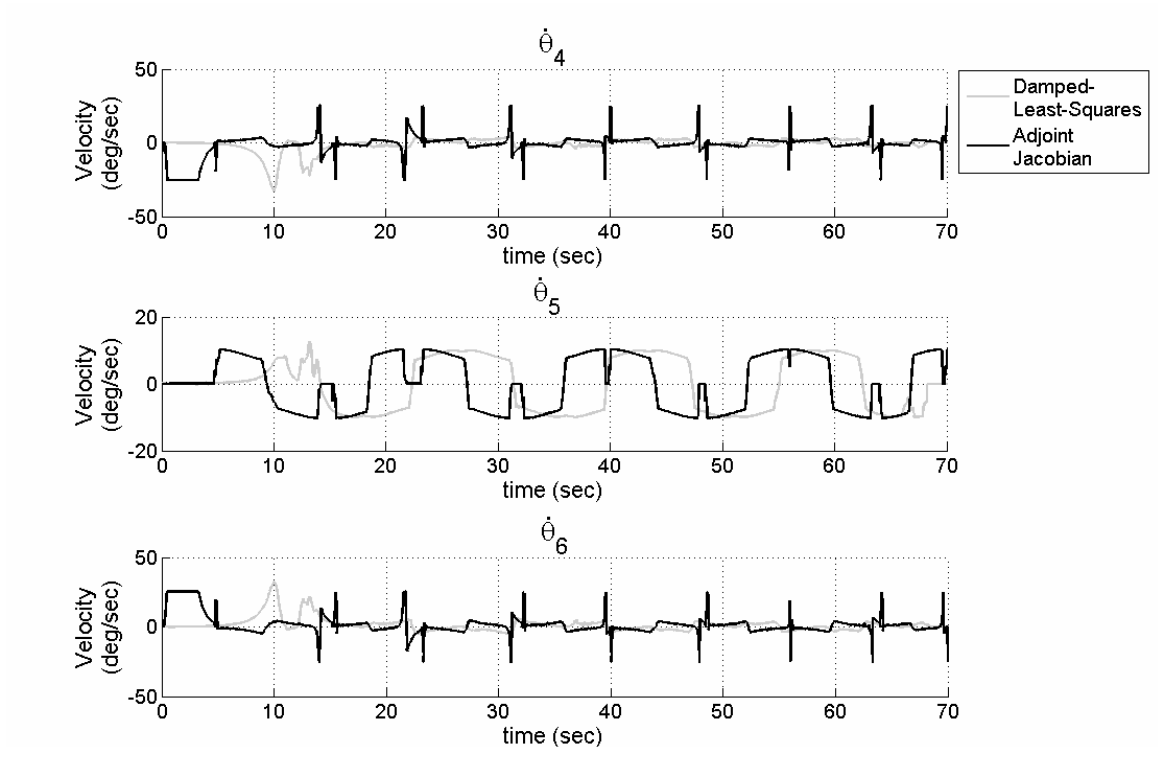


Figure 6.15. Velocity of joints θ_4, θ_5 and θ_6 while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

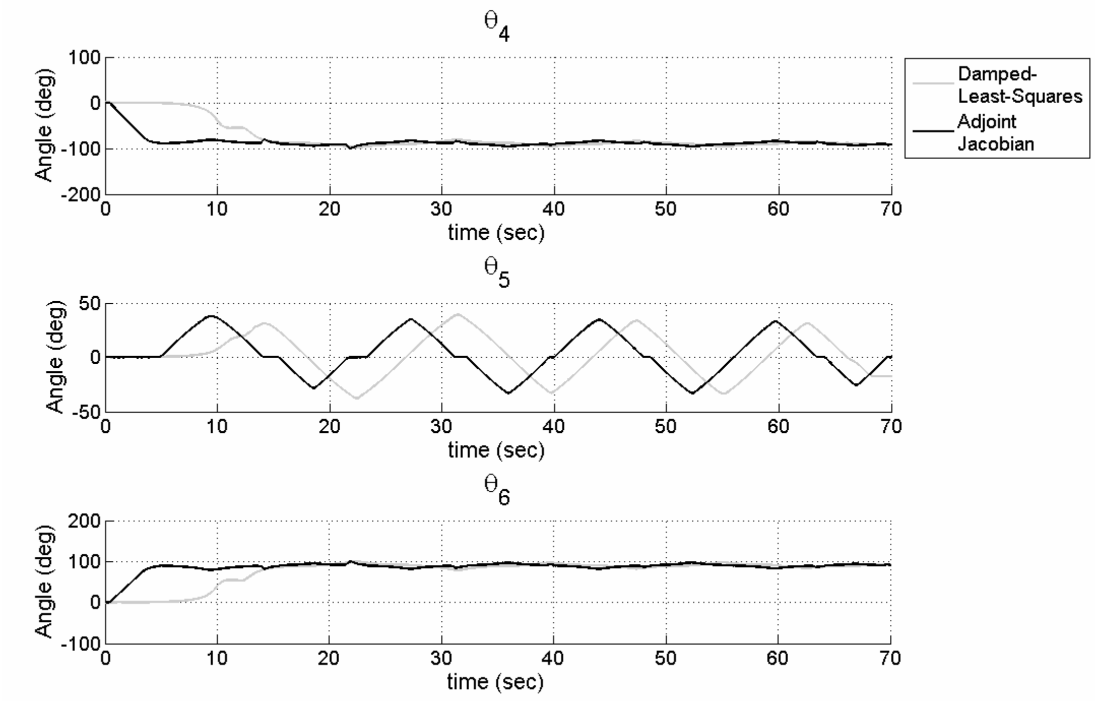


Figure 6.16. Position of joints θ_4, θ_5 and θ_6 while escaping from singularity with virtual fixture along Y-axis w.r.t. the base frame

6.5.3 Passing in the neighborhood of singularity

An exact solution to the inverse kinematics in the neighborhood of wrist singularity yields into considerably high velocity of joints θ_4 and θ_6. The closer to the singularity, the higher is the resulting velocity. Both DLS and adjoint Jacobian reduce such velocities, the former by applying the variable damping factor (equation (6.34)), while the latter by velocity restriction using the norm of the joint velocity (equation (6.52)).

The purpose of this experiment is to analyze the behavior of both approaches when passing in the neighborhood of singularity. Therefore, the initial joint position $\boldsymbol{\theta}_0 = \{0, 11, 81, 0, -2, 0\}$ is used, which is almost the same as the one used in the last experiment (escaping from singularity), but with a slight difference of the joint position so that the robot configuration does not lay exactly at singularity. A VF is defined with $\boldsymbol{T}_{TAR}$ at this position and a translational movement along Y-axis, i.e. $S^l = \{\boldsymbol{R}_{TCP}{}^T \mathbf{l}_1\}$ and $S^\alpha = \{0\}$, where $\mathbf{l}_1 = [0 \quad 1 \quad 0]^T$. Back and fort movements of the end-effector are then executed.

Figure 6.17 and Figure 6.18 show the behavior of end-effector position and orientation, respectively, while the error profile is presented in Figure 6.19. Considerable strong deviations for both position and orientation is observed for the case of DLS when compared with the adjoint Jacobian approach, On the other hand observe that the adjoint Jacobian approach reduces its motion rate when passing near singularity, therefore remaining closer to the virtual constrained path. Maximum error and mean error are given in Table 6.2.

Table 6.2. Maximum and mean error when passing in the neighborhood of wrist singularity

Singularity Robust approach	Maximum Position error (mm)	Mean Position error (mm)	Maximum Orientation error (deg)	Mean Orientation error (deg)
DLS	10.38	1.3835	9.4944	2.9962
adjoint Jacobian	0.6467	0.2412	1.101	0.5775

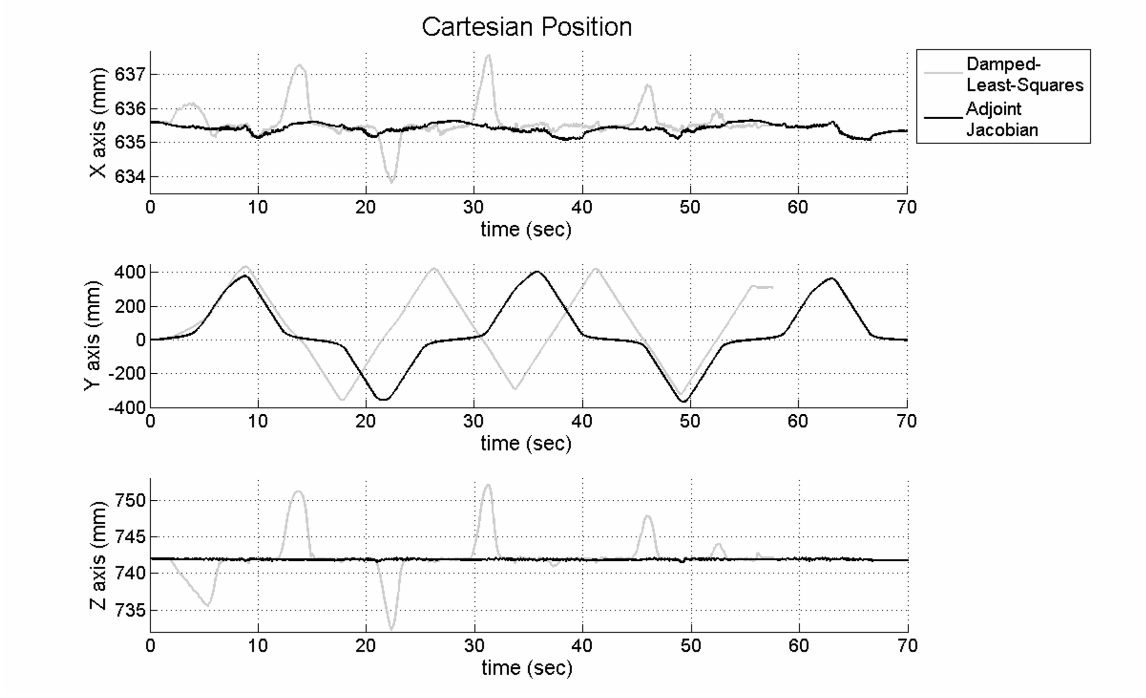


Figure 6.17. End-effector position while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame

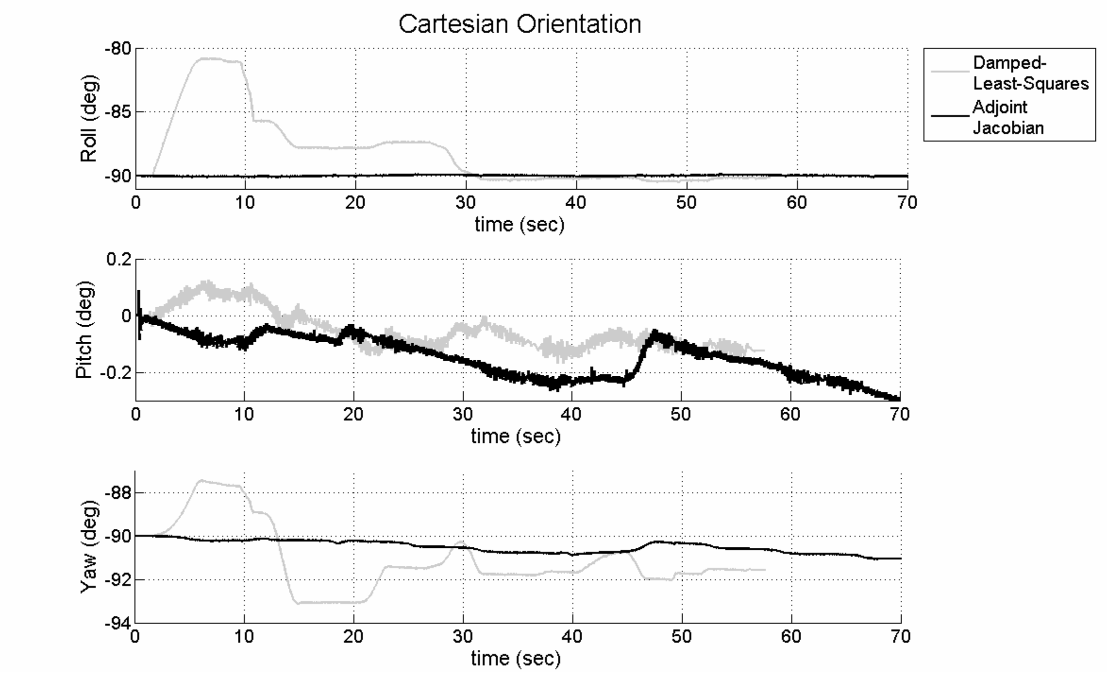


Figure 6.18. End-effector orientation while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame

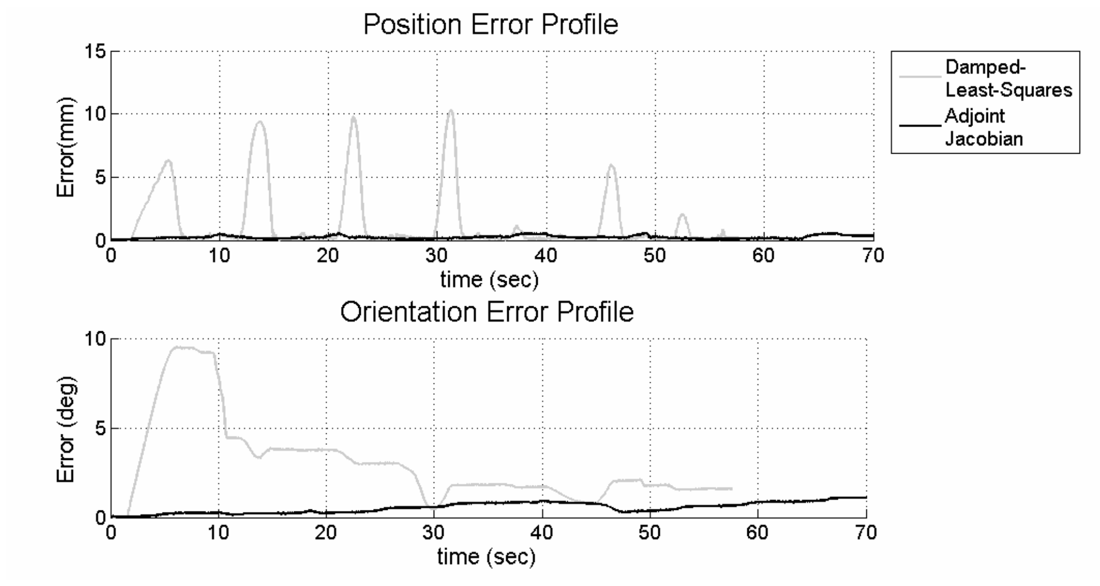


Figure 6.19. End-effector error profile while passing in the neighborhood of wrist singularity with virtual fixture along Y-axis w.r.t. the base frame

The response of each approach in the neighborhood of singularities defers considerably from each other and it can be corroborated when looking at the behavior of the robot in the joint space (Figure 6.20 and Figure 6.21). While the adjoint Jacobian approach reduces joint velocities in the neighborhood of singularity to avoid reconfiguration[7] of the robot, the DLS forces the robot to pass through singularity by damping the velocities. (see Figure 6.20). In adjoint Jacobian method, θ_4 and θ_6 are constantly rotating in such a way that θ_5 does not cross through singular position, while in the case of DLS, θ_4 and θ_6 are kept almost constant and it is rather joint θ_5, which continuously rotates passing through wrist singularity. Figure 6.21 shows the velocity response of both approaches. The adjoint Jacobian reduces smoothly reduces joint velocities when approaching singularities without producing any kind of discontinuity.

[7] In the contest of singularities, reconfiguration means passing frome one side of the singular position to the other.

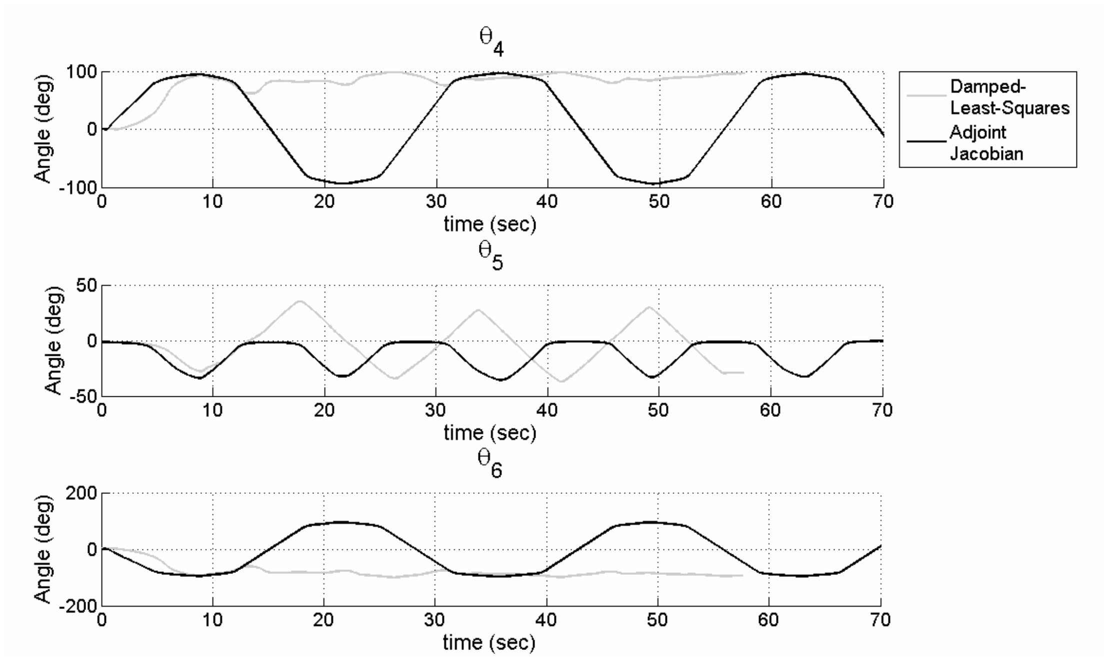


Figure 6.20. Position of joints θ_4, θ_5 and θ_6 while passing near singularity with virtual fixture along Y-axis w.r.t. the base frame

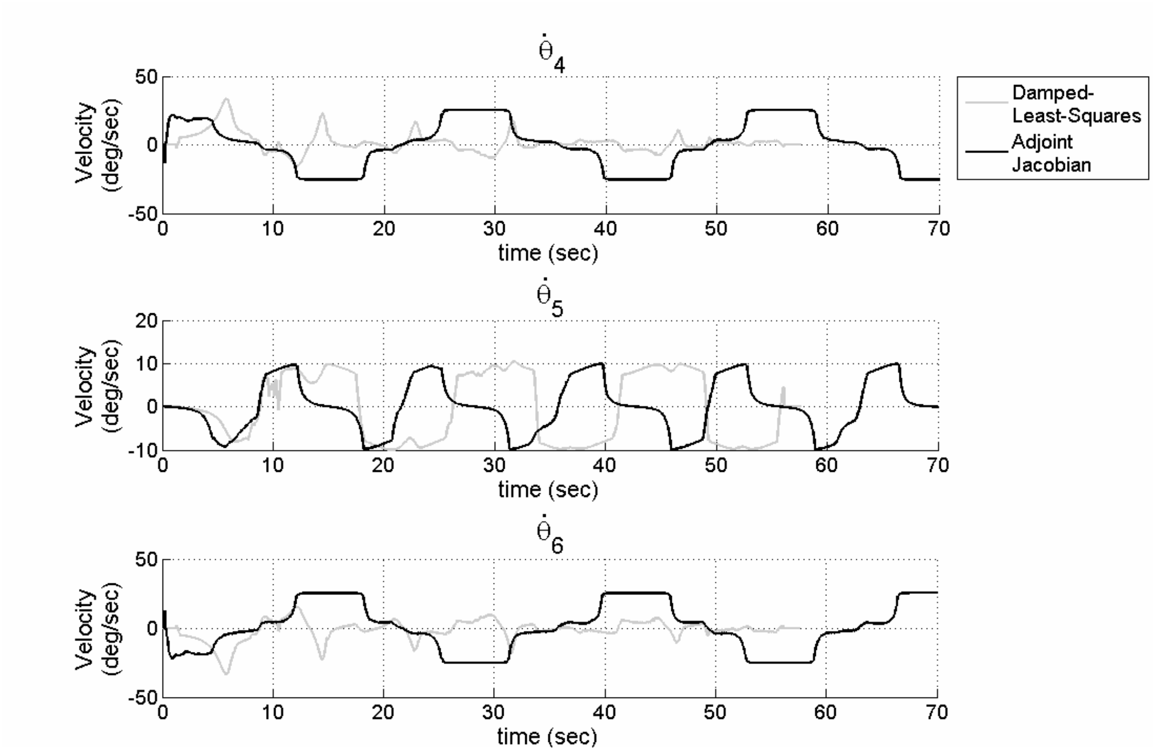


Figure 6.21. Velocity of joints θ_4, θ_5 and θ_6 while passing near singularity with virtual fixture along Y-axis w.r.t. the base frame

6.6 Discussion

The problem of high joint velocities at wrist singularity is solved by both DLS and adjoint Jacobian approaches. The response of DLS is smooth along the subspace orthogonal to the singular direction, what turn out to be very comfortable when commanding the end-effector by hand. Approaching to singularity with forces applied in other directions would degenerate motion causing position deviations, while the velocity behavior remains unchanged. One important objective of this work is to incorporate the robotic system in the OR in such a way that its usage result as intuitive as possible. Moving the robot freely in the space in a unconstrained fashion without been necessary to care about the exactness of the movement may happen, for instance when moving the end-effector apart from the operation scenario to change tool or simply to have more available space for the surgeon to perform other tasks. In such cases, the priority is the surgeon to comfortably move the robot. The DLS approach provides a convenient solution for such scenarios since it avoids high joint velocities and allow crossing singularity smoothly.

A virtual constrained environment, on the other hand, demands the end-effector position to remain always within the allowed subspace. Any deviation out of the permitted subspace could result in any kind of damage during operation, which is unacceptable, especially considering that the robot has direct contact with human beings. Therefore, the DLS cannot be considered for cooperative tasks where position accuracy is demanded, since robot accuracy decreases in the neighborhood of singularity especially along singular direction. The usage of the adjoint Jacobian approach to solve the inverse kinematics of the robot is proposed as alternative for constrained cooperative tasks, since its performance does not affect end-effector position accuracy, degeneration of motion reflects rather in velocity. The null space motion feature of the adjoint Jacobian inversion provides robustness against singular direction. Exactly at singular position velocity discontinuities occur in the joint space. Nevertheless, these can be reduced by reducing the applied force when crossing singularity, which turns out not to be as comfortable for the user as the response of the DLS but assures position accuracy.

7. Conclusions

A cooperative system for robotic assisted surgery is introduced. The combination of a navigation system and a hands-on robotic arm form an integral solution for surgical applications. A new command-based architecture is presented which provides a solid foundation for building complex, robust and scalable applications. A clear modularization of the different tasks as well as a strategic distribution of them along the system framework, depending of their roll within the system, give enough flexibility to cope various applications. Basic functions have been implemented within the new proposed framework to cope different fundamental demands directly related to the robotic arm, such as point to point motion in the joint space, linear motion in the Cartesian space, velocity commanded motion in both joint and Cartesian space, and cooperative motion through a hands-on interface mounted at the robot's end-effector.

Special attention is focused on the interaction of the robotic system with the surgeon, where a cooperative approach appears to be a good candidate to achieve a good integration of the robot within surgical interventions. However such cooperation implies extra safety measurements because direct contact with the human being takes place. The concept of virtual constraints is used to assure safeness during operation by limiting the allowed working space. This is realized in the form of virtual fixtures which guide the tool along a predefined directions or paths. Previous work related to virtual fixtures applies admittance control to create the virtually constrained subspace. This controller relies on the user applied force to generate motion of the end-effector, where even deviation error compensation depends on such applied forces (here known as manual error compensation). In such approach, when deviation error occurs, the virtual preferred directions are redefined to consider such error, creating a new virtual fixture that makes it possible to compensate it. However, it has been shown that manual compensation does not necessary compensate for all deviations, especially when the virtual fixture is translational and the deviation error appears at orientation, or vice versa. In order to solve this problem, the present work proposes another admittance controller with autonomous error compensation, which has a clear division of responsibilities between user and robotic system during cooperative tasks. While the user keeps complete control on the movements along the preferred directions, the robotic system takes care of the error compensation independent of the applied forces. Such approach shows considerable minimization of the error when compared to the manual compensation. Additionally, the problem of robot singularities during cooperative tasks is treated, in particular, the case of wrist singularity, which may appear at any moment within a virtual constrained working area. The other two singularities are practically avoided. Two solutions are compared: the Damped Least Square and the adjoint Jacobian. The first one introduces a damping factor to the Jacobian inversion to prevent high velocities in the neighborhood of singularities and at singularity. Although, smooth motion becomes possible with this approach, position deviations of the end-effector appear, especially when the commanded velocity points in the singular direction. In a virtual constrained environment such deviations may mean getting out of the boundaries, which cannot be permitted. An alternative solution is then considered to enhance position accuracy; this is the so called adjoint Jacobian approach. Degeneration of the motion appears only on velocity while the end-effector position is kept. During cooperative tasks, rather slow motions are executed, and further reduction of the velocity of motion in order to keep a correct position does not represent a problem. The adjoint Jacobian approach produces null space motion to escape from singular directions avoiding the degeneration of the movement in the Cartesian space. In the context of cooperative manipulation, each of both methods can be favorable on particular scenarios. On the one hand, the damped least-squares method can be used for unconstrained motions, where position deviation is not critical, to provide a smooth transition through singularity which may be comfortable for the user. On the other hand, the adjoint Jacobian approach can be applied to virtual constrained motions, where the accuracy of the end-effector position is relevant.

The presented methods significantly contribute in making manually guided robot movements during cooperative tasks safer and more accurate. They increase the assistive functionality of robotic systems and the level of integration within surgical interventions. The interaction between surgeon and robot becomes more intuitive and friendlier. Moreover, the concept of virtual fixtures improves safety measurements during such cooperative tasks, while the surgeon maintains full control over the operation procedure.

References

[1]. Aarno, D., Ekvall, S. and Kragic, D.: Adaptive Virtual Fixtures for Machine-Assisted Teleoperation Tasks. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1139-1144, April 2005.

[2]. Abbott, J. J., and Okamura, A. M.: Analysis of Virtual Fixture contact Stability for Telemanipulation. Proceedings of the IEEE International conference on Intelligent Robots and Systems, pp. 2699-2706, October 2003.

[3]. Abbot, J. J., Hager, G. D., and Okamura, A. M.: Steady-Hand Teleoperation with Virtual Fixtures. Proceedings of the 12th IEEE Workshop on Robot and Human Interactive Communication, pp. 145-151, 2003.

[4]. Abbott, J. J., Marayong, P. and Okamura, A. M.: Haptic Virtual Fixtures for Robot-Assisted Manipulation. In S. Thrun, R. Brooks, and H. Durrant-Whyte, editors, Robotics Research: Results of the 12th International Symposium ISRR, pp. 49-64. Springer, 2007.

[5]. Adams, R. J. and Hannaford, B.: Stable Haptic Interaction with Virtual Environments. IEEE Transactions on Robotics and automation, Vol. 15, No. 3, pp. 465-474, June 1999.

[6]. Adler Jr., J. R., Murphy, M.J., Chang, S.D., Hancock, S.L.: Image-Guided Robotic Radiosurgery. Neurosurgery, 44(6), p. 1299-306, June 1999.

[7]. Armstrong, J.: Catmull-Rom Splines, TechNote TN-06-001, January 2006, http://www.algorithmist.net/media/catmullrom.pdf , last visited: 15th of April, 2007.

[8]. Armstrong, J.: Arc-Length Parameterization Part I, TechNote TN-06-004, May 2006, http://www.algorithmist.net/arclengthparam.html , last visited: 15th of April, 2007.

[9]. Ardence/Venturcom, Inc., Phar Lap ETS.

[10]. Bedrossian, N.S.: Classification of Singular Configuration for Redundant Manipulators, in Proceedings of the IEEE International conference on Robotics and Automation, Vol. 2, pp. 818-823, 1990.

[11]. Bedrossian, N.S., Flueckiger, K.: Characterizing Spatial Redundant Manipulator Singularities, in Proceedings of the IEEE International conference on robotics and automation, pp. 714-719, April 1991.

[12]. Berkelman, P. and Ma, J.: A Compact, Modular, Teleoperated Robotic Minimally Invasive Surgery System. International Conference on Biomedical Robots and Mechatronics, Pisa, Italy, February 2006.

[13]. Bettini, A., Lang, S., Okamura, A., and Hager, G.: Vision Assisted Control for Manipulation Using Virtual Fixtures: Experiments at Macro and Micro Scales. Proceedings of the IEEE International Conference on Robotics and automation, pp. 3354-3361, May 2002.

[14]. Bettini, A., *et al.*: Vision-Assisted Control for Manipulation Using Virtual Fixtures. IEEE Transactions on Robotics, Vol. 20, No. 6, pp. 953-966, December 2004.

[15]. Bruyninckx H. and De Schutter J.: Introduction to Intelligent Robotics. 7th edition, K. U. Leuven, Belgium, 2001.

[16]. Burckhardt, C. W., Flury, P., Glauser, D.: Stereotactic Brain surgery, Integrated MINERVA system meets demanding robotic requirements. In IEEE Engineering in Medicine and Biology, Vol. 14, p. 314-317, 1995.

[17]. Burghart C., Krempien R., Redlich T., Pernozzoli A., Grabowski H.A., Münchenberg J., Albers J., Hassfeld S., Vahl C.: Robot assisted craniofacial surgery; first clinical evaluation, Computer Assisted Radiology and Surgery, 828-833, Paris 1999.

[18]. Buss, S.R.: Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods, Department of Mathematics University of California, San Diego La Jolla, CA 92093-0112, 2004.

[19]. Cai, M., Tianmiao, W., Wusheng; C., Yuru, Z.: A Neurosurgical Robotic System under Image-Guidance. Proceedings of IEEE International conference on Industrial Informatics, p.1245-1250, August, 2006.

[20]. Castillo-Cruces, R.A.: Development of efficient kinematics algorithms for Cartesian control of a Mitsubishi PA10 robot, Master Thesis, Center for Sensorsystems, University of Siegen, Siegen, Germany, 2004.

[21]. Cenk Cavosoglu, M. Tendick, F. Cohn, M. And Shankar Sastry, S.: A Laparoscopic Telesurgical Workstation. IEEE Transactions on Robotics and Automation, Vol. 15, No. 4, pp. 728-739, August 1999.

[22]. Chang, K.S. and Khatib, O.: Manipulator Control at Kinematic Singularities: A Dynamically consistent Strategy, in Proceedings of IEEE International Conference on Intelligent Robots and Systems, Vol. 3, pp. 84-88, 1995.

[23]. Chang, K.S. and Khatib, O.: Operational Space Dynamics: Efficient algorithms for Modeling and control of Branching Mechanisms, in Proceedings of the IEEE International Conference on Robotics and automation, pp. 850-856, April 2000.

[24]. Cheng, F.T., Hour, T.L., Sun, Y.Y. and Chen, T.H.: Study and Resolution of Singularities for a 6-DOF PUMA Manipulator, in IEEE Transactions on Systems, Man, and Cybernetics- Part B, Vol. 27, No. 2, pp. 332-343, April 1997.

[25]. Chiaverini, S. and Egeland, O.: A solution to the singularity problem for six-joint manipulators, in Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 644-649, May 1990.

[26]. Chiaverini, S., Siciliano, B., and Egeland, O.: Review of the Damped Least-Squares Inverse Kinematics with Experiments on an Industrial Robot Manipulator, in IEEE Transactions on Control, Systems Technology, Vol. 2, No. 2, pp. 123-134, June 1994.

[27]. Cleary, K., *et al.*: Robotically Assisted Spine Needle Placement: Program Plan and Cadaver Study. IEEE Symposium on Computer-Based Medical Systems, p. 339-342, 2001.

[28]. Cleary, K., Nguyen, C.: State of the Art in Surgical Robotics: Clinical Applications and Technology Challenges. Computer Aided Surgery, 6:312-328, November 2001.

[29]. Craig, J.: Introduction to Robotics: Mechanics and Control, ISBN 0201095289, Addison-Wesley Publishing, Inc., 1989.

[30]. Dam, E. B., Koch, M., Lillholm, M..: Quaternions, Interpolation and Animation. Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, Denmark, July 17, 1998.

[31]. Dario, P., Guglielmelli, E., Allotta, B. Carrozza, M. C.: Robotics for Medical Applications, in IEEE Robotics & Automation Magazine, 3(3), p. 44-56, 1995.

[32]. Dario, P., Hannaford, B., Menciassi, A: Smart Surgical tools and Augmenting Devices. IEEE Transactions on Robotics and Automation, Vol. 19, No. 5, pp. 782-792, October 2003.

[33]. Davies, B. L., *et al.*: Hands-On Robotic Surgery: Is This the Future?. G. Z. Yang and T. Jiang Eds.: MIAR 2004, LNCS 3150, pp. 27-37, Springer-Verlag Berlin Heidelberg 2004.

[34]. Deo, A.S., Walker, I.D.: Overview of Damped Least-Squares Methods for Inverse Kinematics of Robot Manipulators, in Journal of Intelligent and Robotic Systems, Vol 14, pp 43-68, 1995.

[35]. Dombre, E.: Introduction to Surgical Robotics, 2nd Summer School in Surgical robotics, Montpellier, September 7-14, 2005.

[36]. Edward Colgate, J. Wannasuphoprasit, W. and Peshkin, M. A.: Cobots: Robots for collaboration with human operators. Proceedings of the International Mechanical Engineering Congress and Exhibition, Vol. 58, pp. 433-439, 1996.

[37]. Faulring, E. L., Edward Colgate, J. and Peshkin, M. A.: A High Performance 6-DOF Haptic Cobot. Proceedings of IEEE International Conference on Robotics and Automation, pp. 1980-1985, New Orleans, April 2004.

[38]. Featherstone, R.: Position and Velocity transformations between robot end-effector coordinates and joint angles. International Journal on Robotics Research, 2(2):35-45, 1983.

[39]. Federspil, P. A., Stallkamp, J. and Plinkert, P. K.: Robotic. Eine neue Dimension in der HNO-Heilkunde?. HNO 2001, 49, pp. 505-5132001.

[40]. Fjellstad, Ola-Erik and Fossen, Thor O. (1994). Quaternion Feedback Regulation of Underwater Vehicles, in Proceedings of the 3rd IEEE Conference on control applications, Glasgow, August, p. 24-26, 1994.

[41]. Funda, J., *et al.*: Constrained Cartesian Motion Control for Teleoperated Surgical Robots. IEEE Transactions on Robotics and Automation, Vol. 12, No. 3, pp. 453-465, June 1996.

[42]. Ghdoussi, M., Butner, S. E., Wang, Y.: Robotic Surgery- The Transatlantic Case. Proceedings of the IEEE International Conference on Robotics & Automation, Washington, DC, pp. 1882-1888, May 2002.

[43]. Groß, I: Entwicklung eines Softwaresystems zur universellen Planung chirurgischer Eingriffe an 2D- und 3D-Bildmodalitäten. Doctoral Thesis, Center for Sensorsystems, University of Siegen, Siegen, Germany, 2005.

[44]. Guckenheimer, J. and Holmes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields, ISBN 0387908196, Springer Verlag, 1983.

[45]. Guthart, G. S., Salisbury Jr., J. K.: The IntuitiveTM Telesurgery System: Overview and Application. IEEE International Conference on Robotics & Automation, p. 618-621, San Francisco, CA, USA, April 2000.

[46]. Harris, S. *et al.*: Interactive Pre-operative selection of Cutting Constraints, and Interactive Force Controlled Knee Surgery by a Surgical Robot. W. M. Wells *et al.* Eds.: MICCAI 1998, LNCS 1496, pp. 996-1006, Springer-Verlag Berlin Heidelberg 1998.

[47]. Hashtrudi-Zaad, K., and Salcudean, S. E.: Analysis of control Architectures for Teleoperation Systems with Impedance/Admittance Master and Slave Manipulators. The International Journal of Robotics Research, Vol. 20, No. 6, pp. 419-445, June 2001.

[48]. Heindl, J.: Statische Eigengewichtskompensation von Werkzeugen an der sensorbestückten Roboterhand; Lehrgang R2.02 Roboter mit Sensor-Rückführung; CCG Carl-Cranz-Gesellschaft e.V.; Oberpfaffenhofen; 1988.

[49]. Heuel, S.: Planung und Generierung von Trajektorien für Roboter-Mensch Iteraktion mit Hilfe von Spline Funktionen, Studienarbeit, University of Siegen, 2008.

[50]. Ho, S. C., Hibberd, R. D. and Davies, B. L.: Robot Assisted Knee Surgery. IEEE Engineering in Medicine and Biology Magazine, Vol. 14, Issue 3, pp. 292-300, May/June 1995.

[51]. Holt, D., Zaidi, A., Abramson, J., and Somogyi, R.: Telesurgery: Advances and Trends. Technology Review, University of Toronto Medical Journal, pp. 52-54, 2004.

[52]. Howe, R. D. and Matsuoka, Y.: Robotics for Surgery. Annual Reviews Biomed. Eng., pp. 211-240, 1999.

[53]. Hsu, P., Hauser, J. and Sastry, S.: Dynamic control of Redundant Manipulators, Proceedings of the IEEE International conference on Robotics and Automation, Vol. 1, pp. 183-187, April 1988.

[54]. Hu, T., Castellanos, A. E., Tholey, G. and Desai, J. P.: Real-Time Haptic Feedback in Laparoscopic Tools for Use in Gastro-Intestinal Surgery. T. Dohi and R. Kikinis (Eds.): MICCAI 2002, LNCS 2488, pp. 66-74, Springer-Verlag Berlin Heidelberg 2002.

[55]. Ikeura, R., Monden, H. and Inooka, H.: Cooperative Motion Control of a Robot and a Human. Proceedings of the 3rd IEEE International Workshop on Robot and Human Communication, pp. 112-117, 1994.

[56]. Itoh, T., Kosuge, K., and Fukuda, T.: Human-Machine Cooperative Telemanipulation with Motion and Force Scaling Using Task-Oriented Virtual Tool Dynamics. IEEE Transactions on Robotics and Automation, Vol. 16, No. 5, pp. 505-516, October 2000.

[57]. Jakopec, M., Rodriguez y Baena, F., Harris, S. J.: The Hands-On Orthopaedic Robot "Arobot": Early clinical Trials of Total Knee Replacement surgery, in IEEE Transactions on Robotics and Automation, Vol. 19, No. 5, p. 902-911, October 2003.

[58]. Joly, L. D. and andriot, C.: Imposing Motion Constraints to a Force Reflecting Telerobot through Real-Time Simulation of a Virtual Mechanism. Proceedings of the IEEE International Conference on Robotics and Automation, part 1, pp. 357-362, 1995.

[59]. Kaiser, W.A.; Fischer, H.; Vagner, J. and Selig, M.: Robotic system for biopsy and therapy of breast lesions in a high-field whole -body magnetic resonance tomography unit, Invest Radiol,vol. 35, pp. 513-9., 2000.

[60]. Kapoor, A., Li, M., Taylor, R. H.: Spatial Motion Constraints for Robot Assisted Suturing Using Virtual Fixtures. IEEE International Conference on Robotics & Automation, pp. 1954-1959, Taiwan, September , 2003.

[61]. Kapoor, A., Li, M. and Taylor, R. H.: Spatial Motion Constraints for Robot Assisted Suturing using Virtual Fixtures. Medical Image Computing and Computer Assisted Intervention – MICCAI (2) 2005, pp. 89-96, Palm springs, USA, October 2005.

[62]. Kazerooni, H.: Human/Robot Interaction via the Transfer of Power and Information Signals Part I: Dynamics and Control Analysis. Proceedings of the Annual International Conference of the IEEE Engineering I Medicine and Biology society, Vol. 3, pp. 908-909, November 1986.

[63]. Kazerooni, H.: Human-Robot Interaction via the Transfer of Power and Information Signals. IEEE Transactions on Systems, Man, and Cybernetics, Vol 20, No. 2., pp. 450-463, March/April 1990.

[64]. Kennedy, C. W., Hu, T. And Desai, J. P.: Combining Haptic and Visual Servoing for Cardiothoracic Surgery. Proceedings of the IEEE International Conference on Robotics & Automation, pp. 2106-2111, May 2002.

[65]. Khalil, W. and Dombre, E.: Modeling, Identification & Control of Robots, Hermes Penton Science, ISBN 1903996139, Great Britain, 2002.

[66]. Kieffer, J.: Manipulatior Inverse Kinematics for Unitmed End-Effector Trajectories With Ordinary Singularities, in International Journal of Robotics Research, Vol. 11, No. 3, pp. 225-237, June 1992.

[67]. Kieffer, J.: Differential Analysis of Bifurcations and Isolated Singularities for Robots and Mechanisms, in IEEE Transactions on Robotics and Automation, Vol. 10, No. 1, February 1994.

[68]. Kircanski, M., Kircanski, N., Lekovic, D., Vukobratovic, M.: An experimental study of resolved acceleration control in singularities: damped least-squares approach, in IEEE International Conference on Robotics and Automation, Vol. 4, pp. 2686-2691, 1994.

[69]. Khatib,O.: A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation, in IEEE Journal of Robotics and Automation, Vol. RA-3, No. 1, pp. 43-53, February 1987.

[70]. Knappe, P., Gross, I., Pieck, S., Wahrburg, J.; Kuenzler, S.; Kerschbaumer; F.: Position control of a surgical robot by a navigation system. International Conference on Intelligent Robots and Systems, (IROS 2003). Proceedings. IEEE/RSJ, Vol 4, Issue 27-31, p. 3350-3354, Las Vegas, Nevada, October 2003.

[71]. Kosuge, K., Fujisawa, Y., and Fukunda T.: control of robot directly maneuvered by operator. Proceedings of the IEEE International conference on Intelligent Robots and Systems, pp. 49-54, July 1993.

[72]. Kosuge, K., Yoshida, H., Fukuda, T.: Dynamic Control for Robot-Human Collaboration. Proceedings of 2nd IEEE International Workshop on Robot and Human Communication, pp. 398-399, 1993.

[73]. Kosuge, K. and Kayamura, N.: Control of a Robot Handling an Object in Cooperation with a Human. Proceedings of the 6th IEEE International Workshop on Robot and Human Communication, pp. 142-147, 1997.

[74]. Kumar, R., Jensen, P., Taylor R. H.: Experiments with a Steady Hand Robot in Constrained Compliant Motion and Path Following. Proceedings of the 8th IEEE International Workshop on Robot and Human Interaction (RO-MAN), Pisa, Italy, 1999.

[75]. Kwoh, Y. S., Hou, J., Jonckheere, E. A., Hayati, S.,: A robot with Improved Absolute Positioning, in IEEE Transactions on Biomedical Engineering, vol. 35, No. 2, p. 153-160, February, 1988.

[76]. LabVIEW Intermediate I Successful Development Practices Course Manual. National Instruments Corporation, Part Number 323756B-01, October 2006.

[77]. LabVIEW Real-Time Application Development Course Manual. National Instruments Corporation, Part Number 323843B-01, March 2006.

[78]. Lanfranco, A. R. *et al.*: Robotic Surgery A Current Perspective. Annals of Surgery, Vol. 239, No. 1, pp. 14-21, January 2004.

[79]. Lavallee, S., *et al.*: Image Guided Operating Robot: a Clinical Application in Stereotactic Neurosurgery, in Proceedings of the 1992 IEEE International Conference on Robotics and Automation, p. 618-624, Nice, France, May 1992.

[80]. Li, M., Kapoor, A. and Taylor, R. H.: A Constrained Optimization Approach to Virtual Fixtures. Proceedings of the International Conference on Intelligent Robots and Systems, pp. 2924-2929, Edmonton, Canada, August, 2005.

[81]. Low, K.H. and Dubey, R.N.: A comparative study of generalized coordinates for solving the inverse kinematics problem of a 6R robot manipulator. International Journal on Robotics Research, 5(4):69-88, 1986.

[82]. Luenberger, D.: Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.

[83]. Lueth, T.C.et al: A Surgical Robot System for Maxillofacial Surgery. IEEE Int. Conf. on Industrial Electronics, Control, and Instrumentation (IECON), p. 2470-2475, Aachen, Germany, September 1998.

[84]. Maciejewski, A.A. and Klein, C.A.: Numerical filtering for the operation of robotic manipulators through kinematically singular configurations, J. Robot. Sys., 5, p. 527-552, 1988.

[85]. Marayong, P., Bettini, A. and Okamura, A.: Effect of Virtual Fixture Compliance on Human-Machine Cooperative Manipulation. Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems EPFL, pp. 1089-1095, Lausanne, Switzerland, October 2002.

[86]. Mayer, H., Nagy, I. and Knoll, A.: Skill Transfer and Learning by Demonstration in a Realistic Scenario of Laparoscopic Surgery. Humanoids, Munich, 2003.

[87]. Mayorga, R.V. and Wong, A.K.C.: A singularities avoidance approach for the optimal local path generation of redundant manipulators, in Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 49-54, 1988.

[88]. Micaelli, A., Bidard, C., and Andriot, C.: Decoupling control Based on Virtual Mechanisms for Telemanipulation. Proceedings of the IEEE International Conference on Robotics & Automation, pp. 1924-1931, May 1998.

[89]. Mukundan, R.: Quaternions: From Classical Mechanics to Computer Graphics, and Beyond, In Proceedings of the 7th Asian Technology conference in Mathematics, pp. 97-106, 2002.

[90]. Nagy, I., Mayer, H. Knoll, M.: The Endo[PA]R System for minimally Invasive Robotic Surgery. TUM-I0320 Institute für Informatik der Technische Universität München, Dezember 2003.

[91]. Nakamura R, *et al.*: Development of a sterilizable MRI-compatible manipulator for stereotactic neurosurgery; Proc of the 13th International Congress and Exhibition of Computer Assisted Radiology and Surgery(CARS'99), p. 1019, Paris, France, 1999.

[92]. Nakamura, Y., and Hanafusa, H.: Inverse kinematic solutions with singularity robustness for robot manipulator control. ASME J. Dyn. Sys. Meas. Control 108(2): 163–171, 1986.

[93]. Nakamura, Y.: Advanced Robotics: Redundancy and Optimization, Addison-Wesley Publishing Company, Inc., 1991, ISBN 0-201-15198-7, 1991.

[94]. Nathoo, N., *et al.* In Touch with Robotics:Neurosurgery for the Future. Neurosurgery, Vol. 56, No. 3, pp. 421-433, March 2005.

[95]. Nenchev, D.N.: Tracking manipulator trajectories with ordinary singularities: A null space based approach, IEE Control’94, The University of Warwick, Coventry,UK, pp. 1145-1147,March 1994.

[96]. Nenchev, D. N., Tsumaki, Y., Uchiyama, M., Senft, V., Hirzinger, G.: Two Approaches to Singularity-Consistent Motion of Nonredundant Robotic Mechanisms, Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 2, pp. 1883-1890, USA, 1996.

[97]. Nenchev, D. N., Tsumaki, Y., and Uchiyama, M.: Singularity-Consistent Behavior of Telerobots: Theory and Experiments, The International Journal of Robotics Research, Vol. 17, No. 2, pp. 138-152, 1998.

[98]. Nenchev, D.N., Tsumaki, Y., Uchiyama, M.: Real-Time Motion Control in the Neighborhood of Singularities: A Comparative Study Between the SC and the DLS Methods, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 506-511, Detroit, Michigan, USA, 1999.

[99]. Nenchev, D.N.: Singularity-Consistent Parameterization of Robot Motion and Control, International Journal of Robotics Research, Vol. 19, No. 2, pp. 159-182, 2000.

[100]. Oetomo, D., Ang, M. Jr. and Lim, S.Y.: Singularity Handling on Puma in Operational Space Formulation, Lecture Notes in Control and Information Sciences, Eds. Daniela Rus and Sanjiv Singh, Eds., Vol. 271, pp. 491-500, Springer Verlag 2001.

[101]. Okamura, A. M., *et al.*: Human-Machine Collaborative Systems for Microsurgical Applications. International Journal of Robotics Research, Vol. 24, Issue 9, pp. 731-741, September 2005.

[102]. Operation Lindbergh, A World First in TeleSurgery: The Surgical Act Crosses the Atlantic! New York - Strasbourg, Press Conference, Espace Multimedia, Paris France, September 2001.

[103]. Park, S., Howe, R: D., Torchiana, D. F.: Virtual Fixtures for Robotic Cardiac Surgery. In W. Niessen and M. Viergever (Eds.): MICCAI 2001, LNCS 2208, pp. 1419-1420, 2001.

[104]. Payandeh, S., and Stanisic, Z.: On application of virtual Fixtures as an aid for Telemanipulation and Training. Proceedings of Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 18-23, 2002.

[105]. Petermann J, Kober R, Heinze R, Frölich JJ, Heeckt PF,Gotzen L.: Computer-assisted planning and robot-assisted surgery in anterior cruciate ligament reconstruction. Op Tech Orthop, Vol. 10, pp. 50–55, 2000.

[106]. Pieck, S: Leistungsanalyse ines optischen 3D/6D Lokalisiersystems und dessen Integration in ein chirurgisches Assistenzsystem. Doctoral Thesis, Center for Sensorsystems, University of Siegen, Siegen, Germany, 2006.

[107]. Rininsland, H.: ARTEMIS. A telemanipulator for cardiac surgery. European Journal of Cario-thoracic Surgery, Vol. 16, Supplement 2, pp. 106-111, November 1999.

[108]. Rizun, P. R., McBeth, P. B., Louw D. F. and Sutherland, G. R.: Robot-Assisted Neurosurgery, Surgical Innovation, Vol. 11, No. 2, pp. 99-106, 2004.

[109]. Rosenberg, L. B.: Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation. Proceedings of the IEEE Virtual Reality Annual International Symposium, pp. 76-82, 1993.

[110]. Ruurda, J. P., *et al.*: Feasibility of Laparoscopic Surgery Assisted by a Robotic Telemanipulation System. W. Niessen and M. Viergever (Eds.): MICCAI 2001, LNCS 2208, pp. 1304-1305, Springer-Verlag Berlin Heidelberg 2001.

[111]. Sayers, C.: Remote Control Robotics. ISBN 0387985972, Springer-Verlag, New York, 1999.

[112]. Schneider, O., Troccaz, J., Chavanon and Blin, D.: PADyC: a Synergistic Robot for Cardiac Puncturing. Proceedings of the IEEE International Conference on Robotics & Automation, pp. 2883-2888, April 2000.

[113]. Schneider, O., Troccaz, J.: A Six-Degree-of-Freedom Passive Arm with Dynamic constraints (PADyC) for Cardiac Surgery Application: Preliminary Experiments. Computer aided surgery, Vol. 6, No. 6, pp. 340-351, 2001.

[114]. Senft, V., and Hirzinger, G.: Redundant Motions of Non Redundant Robots- A New Approach to singularity Treatment. IEEE International conference on Robotics and Automation, Vol. 2, pp. 1553-1559, Nagoya, Japan, May 1995.

[115]. Sciavicco, L. and Siciliano, B.: Modelling and Control of Robot Manipulators. Advanced Textbooks in Control and Signal Processing, 2nd. Edition, ISBN 1852332212, Springer 2005.

[116]. Strang, G., editor: Linear algebra and its Applications. Academic Press, New York 1980.

[117]. Su, L. M., *et al.*: Robotic Percutaneous Access to the Kidney: Comparison with Standard Manual Access, in Journal of Endourology, Vol, 16, No. 7, pp. 471-475, September 2002.

[118]. Tabaie, H. A. *et al.*: Endoscopic Coronary Artery Bypass Graft (ECABG) Procedure with Robotic Assistance. Heart Surgery Forum 1999-0552, 2 (4), pp. 310-317, 1999.

[119]. Taylor, R.H.: Planning and Execution of Straight Line Manipulator Trajectories, MIT Press, pp. 265-286, Cambrige, Mass, 1982.

[120]. Taylor, R. H. Mittelstadt, B. D., Williamson, B., Musits, B. L., Glassman E., Bargar, W. L.: An Image-Directed Robotic System for Precise Orthopaedic surgery, in IEEE Transactions on Robotics and Automation, Vol. 10, No. 3, pp. 261-275, 1994.

[121]. Taylor, R. H., *et al.*: A Telerobotic Assistant for Laparoscopic Surgery. IEEE Engineering in Medicine and Biology Magazine, Vol. 14, Issue 3, pp. 279-288, 1995.

[122]. Taylor, R. H.: Robots as surgical assistants: Where we are, wither we are tending, and how to get there, in AIME 97, Sixth ed., pp. 3-11, Genoble, France, 1997.

[123]. Taylor, R. H., *et al.*: A Steady-Hand Robotic System for Microsurgical Augmentation. International Journal of Robotic Research, Vol. 18, No. 2, pp. 1201-1210, December 1999.

[124]. Taylor, R. H. and Stoianovici, D.: Medical Robotics in Computer-Integrated Surgery. IEEE Transactions on Robotics and Automation, Vol. 19, No. 5, pp. 765-781, October 2003.

[125]. Taylor, H.: A perspective on Medical robotics. Proceedings of the IEEE, Vol. 94, No. 9, pp. 1652-1664, September 2006.

[126]. Trevelyan, J.P., Kovesi, P.D., Ong, M. and Elford, D.: ET: A Wrist Mechanism without Singular Positions, The International Journal of Robotics Research, No. 4, pp. 71-85, 1986.

[127]. Troccaz, J., Peshkin, M., Daies, B.: The Use of Localizers, Robots and Synergistic Devices in CAS. Proceedings of the First Joint Conference on Computer Vision, Virtual Reality, and Robotics in Medicine and Medical Robotics and Computer-Assisted surgery, pp. 727, Grenoble, France, 1997.

[128]. Troccaz, J.: Introduction to medical robotics. Summer European University on Surgical Robotics, Montpellier, September. 2003.

[129]. Troccaz, J., *et al.*: Medical Image Computing and Computer Aided Medical Interventions applied to soft tissues: Work in Progress in Urology. Proceedings of the IEEE, Vol. 94, Issue 9, pp. 1665-1677, Sept. 2006.

[130]. Tsumaki, Y., Kotera, S., Nenchev, D. N. and Uchiyama, M.: Singularity-Consistent Inverse Kinematics of a 6 D.O.F. Manipulator with a Non-Spherical Wrist, in Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, USA, pp. 2980-2985, 1997.

[131]. Tsumaki, Y., Nenchev, D. N., Kotera, S. and Uchiyama, M.: Teleoperation based on the adjoint Jacobian approach, in IEEE Control Systems Magazine, Vol. 17, No. 1, pp. 53-62, February 1997.

[132]. Tsumugiwa, T., Yokogawa, R. and Hara, K.: Variable Impedance Control with Virtual stiffness for Human-Robot Cooperative Task. Proceedings of the 41st SICE Annual Conference, Vol. 4, pp. 2329-2334, 2002.

[133]. Tsuniaki, Y., Nenchev D. N. and Uchiyama M.: Experimental Teleoperation of a Nonredundant Slave Arm at and around Singularities, in Proceedings of the IEEE International conference on Robotics and automation, pp. 385-392, April 1996.

[134]. Turro, N., Khatib, O., and Coste-Maniere, E.: Haptically Augmented Teleoperation, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 386-392, May 2001.

[135]. Uchiyama, M.: A study of Computer control of Motion of a Mechanical Arm (1st Report, Calculation of Coordinative Motion Considering singular Points), in Bulletin of the Japan society of Mechanical Engineers, Vol. 22, No. 173, pp. 1640-1647, November 1979.

[136]. Wahrburg, J., Pieck, S.,Gross, I., Knappe P, Kuenzler S., Kerschbaumer, F.. A Navigated Mechatronic System with Haptic Features to Assist in Surgical Interventions, Journal of Computer Aided Surgery, Vol. 8; pp. 292-299, 2003.

[137]. Wampler, C.W. II: Manipulator Inverse Kinematic solution Based on Vector Formulations and Damped Least-Squares Methods, in IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, January/February 1986.

[138]. Wang, H., Kearney, J., and Atkinson, K.: Arc-Length Parameterized Spline Curves for Real-Time Simulation, in Proceedings of 5th International Conference on Curves and Surfaces, pp. 387-396, June 2002.

[139]. Wang, H., Kearney, J., and Atkinson, K.: Robust and Efficient Computation of the Closest Point on a Spline Curve, Proceedings of 5th International Conference on Curves and Surfaces, pp.397-406, June 2002.

[140]. Whitney, D. E.: Resolved Motion Rate Control of Manipulators and human Prostheses, in IEEE Transactions on Man-Machine Systems, Vol. MMS-10, No. 2, pp. 47-53, June 1969

[141]. Worsnopp, T. *et al.*: Controlling the Apparent Inertia of Passive Human-Interactive Robots. Transactions of the ASME, Vol. 128, pp. 44-52, March 2006.

A Quaternion

The quaternion representation consists of a scalar part $\eta \in \mathbb{R}$ and a vector $\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z \end{bmatrix}^T$.

$$\phi = \begin{bmatrix} \eta \\ \boldsymbol{\varepsilon} \end{bmatrix} = \begin{bmatrix} \eta \\ \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \tag{A.1}$$

If $\|\phi\| = 1$, then ϕ is know as the unit quaternion. The set of unit quaternions constitutes a unit sphere in four-dimensional space. For this special case, there exists a vector $\varepsilon' \in \mathbb{R}^3$ and $\theta \in \left[-\pi, \pi\right]$ such that $\phi = \begin{bmatrix} \cos\theta & \varepsilon' \sin\theta \end{bmatrix}^T$, the unit quaternions play an important part in the relation to general rotation, i.e. it give a physical meaning to a quaternion, where $\boldsymbol{\varepsilon}'$ and θ represent axis of rotation and angle of rotation, respectively [30].

The quaternion ϕ can be interpreted as a complex number with η being the real part and $\boldsymbol{\varepsilon}$ the complex part [40]. Hence, the complex conjugate of ϕ is defined as:

$$\bar{\phi} = \begin{bmatrix} \eta \\ -\boldsymbol{\varepsilon} \end{bmatrix} \tag{A.2}$$

Thus, the inverse rotation matrix can be expressed as:

$$\boldsymbol{R}^{-1}(\phi) = \boldsymbol{R}^T(\phi) = \boldsymbol{R}(\bar{\phi}) \tag{A.3}$$

Since, successive rotations involve multiplication between two rotation matrices, and quaternion multiplication is equivalent to orthogonal matrix multiplication, it can be stated that:

$$\boldsymbol{R}(\phi_1)\boldsymbol{R}(\phi_2) = \boldsymbol{R}(\phi_1\phi_2), \tag{A.4}$$

The quaternion multiplication is defined as:

$$\phi_1\phi_2 = \begin{bmatrix} \eta_1 & -\boldsymbol{\varepsilon}_1^T \\ \boldsymbol{\varepsilon}_1 & \eta_1 \boldsymbol{I}_{3\times3} + \boldsymbol{S}(\boldsymbol{\varepsilon}_1) \end{bmatrix} \begin{bmatrix} \eta_2 \\ \boldsymbol{\varepsilon}_2 \end{bmatrix} \tag{A.5}$$

where ***I*** is the identity matrix and $\boldsymbol{S}(\boldsymbol{\varepsilon}_1)$ is the skew-symmetric matrix of $\boldsymbol{\varepsilon}_1$, such that

$$\boldsymbol{\varepsilon}_1 \times \boldsymbol{\varepsilon}_2 = S(\boldsymbol{\varepsilon}_1)\boldsymbol{\varepsilon}_2 \tag{A.6}$$

A.1 Unit quaternion to rotation matrix conversion

A rotation matrix ***R*** in terms of unit quaternions is written as:

$$\boldsymbol{R}(\phi) = \begin{bmatrix} \eta^2 + \varepsilon_x^2 - \varepsilon_y^2 - \varepsilon_z^2 & 2(\varepsilon_x\varepsilon_y - \eta\varepsilon_z) & 2(\varepsilon_x\varepsilon_z + \eta\varepsilon_y) \\ 2(\varepsilon_x\varepsilon_y + \eta\varepsilon_z) & \eta^2 - \varepsilon_x^2 + \varepsilon_y^2 - \varepsilon_z^2 & 2(\varepsilon_y\varepsilon_z - \eta\varepsilon_x) \\ 2(\varepsilon_x\varepsilon_z - \eta\varepsilon_y) & 2(\varepsilon_y\varepsilon_z + \eta\varepsilon_x) & \eta^2 - \varepsilon_x^2 - \varepsilon_y^2 + \varepsilon_z^2 \end{bmatrix} \tag{A.7}$$

A.2 Rotation matrix to unique quaternion conversion

Conversion of rotation matrix ***R*** to the corresponding unit quaternion ϕ [30]:

$$\begin{aligned} r_{11} + r_{22} + r_{33} + r_{44} &= 4 - 4(\varepsilon_x{}^2 + \varepsilon_y{}^2 + \varepsilon_z{}^2) \\ &= 4 - 4(1 - \eta^2) \qquad , \\ &= 4\eta^2 \end{aligned} \tag{A.8}$$

since $\eta^2 + \varepsilon_x{}^2 + \varepsilon_y{}^2 + \varepsilon_z{}^2 = 1$, i.e. it s norm is equal to 1. This yields into:

$$\begin{aligned} \eta &= \pm\frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + r_{44}} \\ \varepsilon_x &= \frac{r_{32} - r_{23}}{4\eta} \\ \varepsilon_y &= \frac{r_{13} - r_{31}}{4\eta} \\ \varepsilon_z &= \frac{r_{21} - r_{12}}{4\eta} \end{aligned} \tag{A.9}$$

where r_{ij} is the element of ith row and jth column of ***R***. The sign of η cannot be defined. The signs of $\boldsymbol{\varepsilon}$ also depends of its choice. Both choices yield the same rotation, but this may not be trivial when using quaternions for interpolation.

B Spline functions

B.1 Catmull-Rom splines

Spline functions are functions defined piecewise by polynomials. These functions are most frequently used to describe parametric curve:

$$Q(t) = (x(t), y(t), z(t)) \tag{B.1}$$

The Catmull-Rom spline interpolates *n* data points (also called knots) with a piecewise cubic polynomial which produces a C^1-continuous curve that passes through the knots and considers the tangent values at each knot to defines the shape of the curve [7].

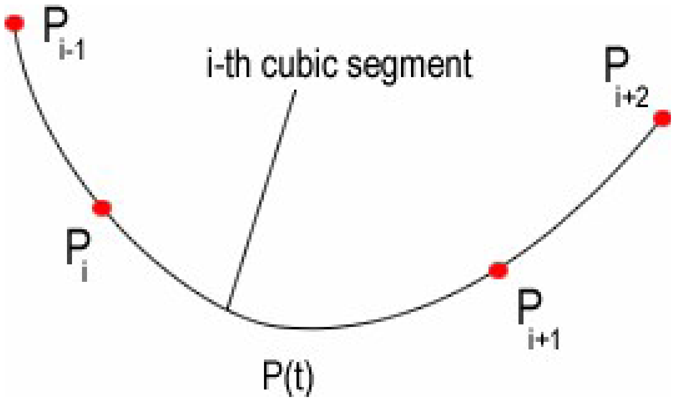


Figure B.1. Segment wise parametric curve

A segment between two points (p_i and p_{i+1}) is considered a single curve $P(t)$ with the condition that:

$$\begin{aligned} P(0) &= p_i \\ P(1) &= p_{i+1} \end{aligned} \tag{B.2}$$

Additionally, two constraints are considered for the definition of the tangents at each knot by using the auxiliary points p_{i-1} and p_{i+2} (see Figure):

$$\begin{aligned} P'(0) &= \alpha(p_{i+1} - p_{i-1}) \\ P'(1) &= \alpha(p_{i+2} - p_i) \end{aligned} \tag{B.3}$$

where α is the tension parameter with values between 0 and 1. As α approaches 1, the bend at each knot reduces. Normally the Catmull-Rom spline uses a tension value of ½. The tangent at each knot is parallel to the chord between the adjacent points. The general expression of a cubic polynomial curve is:

$$Q(t) = at^3 + bt^2 + ct + d \tag{B.4}$$

The derivative of equation (B.4) is:

$$Q' = 3at^2 + 2bt + c \tag{B.5}$$

After applying the conditions (B.2) and (B.3) to equations (B.4) and(B.5), with α=½, and doing some mathematic, it yields

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} \tag{B.6}$$

Equation (B.6) generates a cubic curve of the segment between p_i and p_{i+1}. Notice however that it can be applied to every segment of the curve. The tangent direction at *Q*(*t*) can be straightforward calculated using equation (B.5). A detailed deduction of the Catmull-Rom spline function can be found in [7].

B.2 Arc-length parameterization

The arc-length parameterization of a parametric curve as expressed in equation (B.1) is a two steps process [138], [8]:

- Calculation of arc-length *s* as a function of the parameter *t*: $s = A(t)$. Since *s* is a strictly increasing function of *t*, there is a one to one relationship between *s* and *t*.
- Calculation of t, as the inverse of the arc-length function: $t = A^{-1}(s)$.

The arc-length parameterization of the curve is obtained when substituting $t = A^{-1}(s)$ into *Q*(*t*):

$$P(s) = (x(A^{-1}(s)), y(A^{-1}(s)), z(A^{-1}(s))) \tag{B.7}$$

where $s \in [0, L]$ and *L* is the total length of the curve.

The arc length s of a curve, denoting its total length, is calculated as a function of t by the integration formula

$$s = A(t) = \int_{t_0}^{t} \sqrt{(x'(t))^2 + (y'(t))^2 + (z'(t))^2 dt} \tag{B.8}$$

In general, equation (B.8) cannot be calculated analytically, i.e. the arc-length parameterization must be solved numerically. The method used in this work compute the approximate arc-length parameterized curve in three steps:

- Calculation and summation of arc-length of all the segments of the original spline curve $Q(t)$ to determine the arc length L.
- Find m+1 equally spaced knots located at $0, \tilde{l}, 2\tilde{l}, ..., m\tilde{l}$ along $Q(t)$, where $\tilde{l} = L/m$ equal the length of each segment of the parameterized curve.
- Calculation of the parameter values $\tilde{t}_0, \tilde{t}_1, ..., \tilde{t}_m$ that divide the spline curve into equal arc-length segments.

The final result is an approximately arc-length parameterized piecewise spline curve divided into m cubic segments. The complete algorithm for the arc-length parameterization can be found in [138].

C Quadratic minimization of a cubic spline function

Suppose that a parametric spline function **p**(s) is given and $\mathbf{p}_0 = (x_0, y_0, z_0)$ is a point in the space. The square of the distance from **p**(s) and **p**$_0$ on a spline curve is

$$\xi(s) = (x(s) - x_0)^2 + (y(s) - y_0)^2 + (z(s) - z_0)^2, \tag{C.1}$$

Where $x(s)$, $y(s)$, and $z(s)$ are cubic spline functions of the parameter s. The closest point to **p**$_0$ on the spline curve $\mathbf{p}_1 = (x(\hat{s}), y(\hat{s}), z(\hat{s}))$ is determined by obtaining the $\hat{s}$ value that minimizes $\xi(s)$ [139].

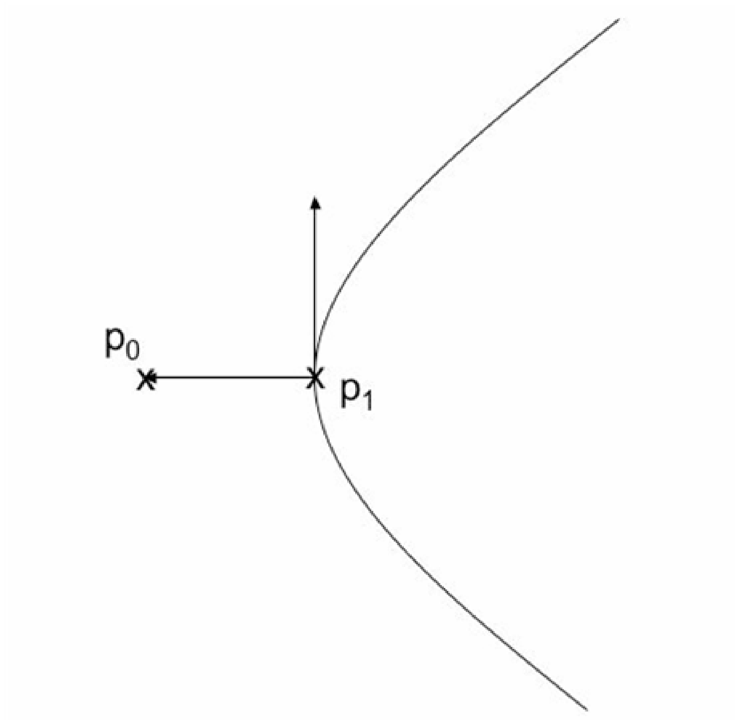


Figure C.1. Closest point of spline curve to p$_0$ and its tangent vector

The quadratic minimization technique is then used to obtain the $\hat{s}$ value that minimizes $\xi(s)$. Let $\hat{s}_1$, $\hat{s}_2$ and $\hat{s}_3$ be three initial estimates of $\hat{s}$. The quadratic polynomial interpolating $\xi(s)$ at $\hat{s}_1$, $\hat{s}_2$ and $\hat{s}_3$ states as follows:

$$P(s) = \frac{(s-\hat{s}_2)(s-\hat{s}_3)}{(\hat{s}_1-\hat{s}_2)(\hat{s}_1-\hat{s}_3)}\xi(\hat{s}_1) + \frac{(s-\hat{s}_1)(s-\hat{s}_3)}{(\hat{s}_2-\hat{s}_2)(\hat{s}_2-\hat{s}_3)}\xi(\hat{s}_1) + \frac{(s-\hat{s}_1)(s-\hat{s}_2)}{(\hat{s}_3-\hat{s}_1)(\hat{s}_3-\hat{s}_2)}\xi(\hat{s}_3). \tag{C.2}$$

The minimum of *D*(*s*) is obtained by calculating the minimum of *P*(*s*):

$$\hat{s}^k = \frac{1}{2}\cdot\frac{(\hat{s}_2^2-\hat{s}_3^2)\xi(\hat{s}_1)+(\hat{s}_3^2-\hat{s}_1^2)\xi(\hat{s}_2)+(\hat{s}_1^2-\hat{s}_2^2)\xi(\hat{s}_3)}{(\hat{s}_2-\hat{s}_3)\xi(\hat{s}_1)+(\hat{s}_3-\hat{s}_1)\xi(\hat{s}_2)+(\hat{s}_1-\hat{s}_2)\xi(\hat{s}_3)} \quad k=1,2,3,..., \tag{C.3}$$

The value giving the largest P(s) among $\hat{s}_1$, $\hat{s}_2$ $\hat{s}_3$ and $\hat{s}^k$ is then eliminated. The expression (C.3) is then evaluated in a like manner with the remaining values until some error tolerance for P(s) is reached. It can be shown that with a good estimate of the initial estimates, the iteration has a superlinear converge rate to $\hat{s}$ [82].

D Singular value decomposition

The singular value decomposition (SVD) is an important factorization of matrices. It provides a powerful method for examining of the characteristics of matrices. Some applications employing the SVD include computing the pseudoinverse, matrix approximation. The SVD is defined as follows [93]:

If $\boldsymbol{A} \in \Re^{m\times n}$ and rank $\boldsymbol{A}$=k, then there exist orthogonal matrices

$$\begin{gathered} U = (u_1 \quad \cdots \quad u_m) \in \Re^{m\times m} \\ V = (v_1 \quad \cdots \quad v_m) \in \Re^{n\times n} \end{gathered} \tag{D.1}$$

Such that $\boldsymbol{A}$ is represented by

$$\begin{gathered} \boldsymbol{A} = \boldsymbol{U\Sigma V}^T \\ \boldsymbol{\Sigma} \triangleq diag(\rho_1, \cdots, \rho_p) \in \Re^{m\times n} \\ p = \min\{m, n\} \\ \rho_1 \geq \rho_2 \geq \cdots \geq \rho_k > 0 \\ \rho_{k+1} = \cdots = \rho_p = 0 \end{gathered} \tag{D.2}$$

where $\rho_i (i = 1, \cdots, p)$ are the singular values of $\boldsymbol{A}$, ρ_1 and ρ_p having the largest and smallest singular values respectively. The singular values are uniquely determined although $\boldsymbol{U}$ and $\boldsymbol{V}$ may not be.

D.1 Relationship between pseudoinverse and SVD

The pseudoinverse of A expressed with the SVD is represented by

$$\begin{gathered} \boldsymbol{A}^\# = \boldsymbol{V\Sigma}^\# \boldsymbol{U}^T \\ \boldsymbol{\Sigma}^\# \triangleq \underbrace{diag\,(\frac{1}{\rho_1}, \frac{1}{\rho_2}, \cdots, \frac{1}{\rho_k}, 0, \cdots, 0)}_{p=\min(m,n)} \in \Re^{n\times m} \\ \rho_1 \geq \rho_2 \geq \cdots \geq \rho_k > 0 \end{gathered} \tag{D.3}$$

Where $1/\rho_i (i = 1, \cdots, k)$ and p-k zeros are the singular values of $\boldsymbol{A}^\#$, among which $1/\rho_k$ is the largest.