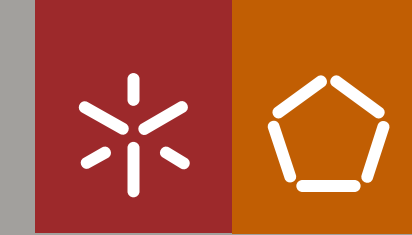




Carlos André de Oliveira Faria

Development of a robotic system to assist
neurosurgeons in minimally invasive
stereotactic procedures



Universidade do Minho
Escola de Engenharia

Cofinanciado por:



Ciência, Tecnologia
e Ensino Superior

FCT

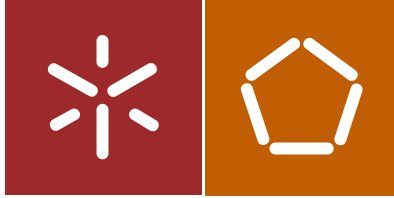
Fundação
para a Ciência
e a Tecnologia



PORTUGAL
2020



UNIÃO EUROPEIA
Fundo Social Europeu



Universidade do Minho
Escola de Engenharia

Carlos André de Oliveira Faria

Development of a robotic system to assist
neurosurgeons in minimally invasive
stereotactic procedures

Tese de Doutoramento
Programa Doutoral em Engenharia Biomédica

Trabalho efetuado sob a orientação de
Prof. Estela Bicho
Prof. Wolfram Erlhagen

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-CompartilhaIgual
CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

[Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que lhe atribuam a si o devido crédito e que licenciem as novas criações ao abrigo de termos idênticos.]

Acknowledgments

Working towards this thesis was certainly an interesting and challenging journey. A journey I would not be able to undertake without the assistance, guidance and friendship of some amazing people.

In first place, express my heartfelt gratitude to my advisors, Prof. Estela Bicho, Prof. Wolfram Erlhagen and Dr. Manuel Rito, for their expertise, guidance and encouragement through rough seas and calm waters.

To my colleagues and friends of the MARLab, Tiago Malheiro, Flora Ferreira, Luís Louro, Toni Machado, Gianpaolo Gulletta, Weronika Wojtak, Carolina Vale, João Sepúlveda and many more to mention, I thank you for your counsel, for your friendship, and for all the fun we had through these years.

To Prof. Giancarlo Ferrigno and Prof. Elena de Momi I thank you for the opportunity to work at the NearLab next to such brilliant and inspiring people and for the chance to participate in the ACTIVE project. It was an experience I will forever cherish.

To Prof. João Vilaça and to Prof. Jaime Fonseca, I am grateful for your knowledge, guidance, and for welcoming me in the AAILab. Without their assistance, this project would not be possible.

To the Centro Hospitalar e Universitário de Coimbra, the Hospital da Luz Guimarães, and Hospital di Garda Milano, I am grateful for allowing me to conduct research at your facilities, and for the time and resources provided.

Finally and at the top of the list, I thank to my mother, to my father, to my brother and to my dearest Rita, they are the closest to my heart and to whom I owe it all.

This work was partially supported by the NETT Project (FP7-PEOPLE-2011-ITN-289146); by the ACTIVE Project (FP7-ICT-2009-6-270460) and by the Foundation for Science and Technology, Portugal (grant number SFRH/BD/86499/2012).

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Resumo

Desenvolvimento de um sistema robótico para auxílio de neurocirurgias em procedimentos estereotáxicos minimamente invasivos

Os robôs revolucionaram a indústria e estão a ser aplicados em aplicações mais complexas e críticas, incluindo cirurgias. O método estereotáxico foi desenvolvido para alcançar estruturas intracranianas sem visualização direta e através de uma abordagem minimamente invasiva, baseada num sistema de coordenadas. Devido ao formato das cirurgias estereotáxicas e funcionais, a adequabilidade de robôs como assistentes aos neurocirurgias é aparente. Os robôs são agentes precisos, repetíveis, e configuráveis, frequentemente nomeados para tarefas de orientação de ferramentas. Apesar destas vantagens, apenas um número limitado de robôs cirúrgicos chegou ao mercado. Sistemas robóticos cirúrgicos atualmente em prática exibem elevados custos com um lento retorno financeiro, uma limitação causada em parte pela dificuldade em transferir tecnologia dos centros de investigação para as salas de operação. Nesta tese, olhámos para os sistemas robóticos neurocirúrgicos-chave para identificar os denominadores comuns e as características mais procuradas, como substituição da frame estereotáxica, e uma solução robótica acessível, flexível, interactiva, e precisa. Propomos duas soluções para abordar este problema. Primeiro, adaptámos um manipulador industrial de 6 graus-de-liberdade com um sistema de controlo monolítico, como prova de conceito e ferramenta de investigação. A segunda solução consiste numa arquitectura de controlo modular, construída sobre um modelo de componente e comunicação bem estabelecido, com um comportamento determinístico e com suporte para restrições de tempo real críticas, testado num manipulador de 7 graus de liberdade. No cerne da arquitectura modular está um método analítico de cinemática inversa para manipuladores redundantes capaz de resolver unicamente o espaço nulo, evitar singularidades, e limites de juntas. Este método oferece ao manipulador a flexibilidade necessária para se adaptar ao espaço de trabalho disponível que é partilhado com a equipa cirúrgica e com outros equipamentos. A arquitectura proposta inclui uma lista de componentes de controlo e supervisão interconectados para executar as diferentes tarefas cirúrgicas. Desenvolvemos e validámos um novo algoritmo para determinar os parâmetros reais do modelo cinemático de manipuladores. Testámos a precisão de aplicação do sistema in vitro e registámos um erro médio de 0.567 mm e um erro máximo de 1.094 mm. A vantagem da solução proposta é a sua performance, flexibilidade em integrar novos módulos, e na capacidade de melhorar componentes sem ter de renovar a solução por completo, ou escalar os custos.

Palavras-chave: Cirurgia Estereotáxica, Controlo de Alto-nível, Robótica.

Abstract

Development of a robotic system to assist neurosurgeons in minimally invasive stereotactic procedures

Technology is often the answer to most of our problems. Robots revolutionized the industry and are finding their way into more complex and critical applications, including surgery. The stereotactic method was developed to reach intracranial brain structures without direct visualization and through a minimally invasive approach, based on a coordinate system. Due to the format of stereotactic and functional surgeries, the suitability of robots as a neurosurgeon's assistants is apparent. Robots are accurate, repeatable, and configurable agents often assigned to tool guidance. Despite these advantages, only a handful of surgical robots reached the market. Current surgical robotic systems in practice exhibit high up-front costs with slow investment return, a limitation in part caused by the difficulty in transferring technologies from the research center into the surgical room. In this thesis, we examine the principal neurosurgical robotic systems to identify the common denominators and the most requested features, replacement of the stereotactic frame, affordable access to flexible, interactive, and accurate robotic solutions. We propose two solutions to address these subjects. First, we adapted an industrial 6 DoF manipulator with a monolithic control system, as a proof-of-concept and research tool. The second solution consists of a modular control architecture, built on top of a well-established component and communication model, with deterministic behavior and supporting hard-real-time constraints, tested in a 7 DoF manipulator. At the core of the modular architecture, is a novel analytic inverse kinematics method for serial redundant manipulators capable of uniquely solving the nullspace, avoiding singularities, and joint limits. This method provides the manipulator with the flexibility to adapt to the available workspace, which is shared by the surgery team and by other equipment. The proposed architecture also includes a list of interconnected controllers and supervisor components to perform the different surgical tasks. We also developed and validated a new algorithm to determine the real kinematic model parameters of manipulators, parameters that may differ from nominal values. We tested the system's application accuracy in vitro and registered a mean error of 0.567 mm, and a maximum error of 1.094 mm. The real advantage of the proposed solution lies in its performance, flexibility to integrate new modules, and in the capacity to upgrade individual components without the need to overhaul the entire solution, or escalate the costs.

Keywords: High-level Control Systems, Robotics, Stereotactic Surgery.

Contents

Acknowledgments	iii
Resumo	v
Abstract	vi
List of Figures	xv
List of Tables	xvi
Nomenclature	xvi
1 Introduction	1
1.1 Contributions	2
1.2 Outline	3
2 Stereotactic and Functional Neurosurgery	5
2.1 Stereotactic Imaging and Technology	5
2.1.1 Stereotactic Imaging	6
2.1.2 Stereotactic Technology	8
2.1.3 Frameless Technique	10
2.2 Stereotactic Procedures	11
2.2.1 Deep Brain Stimulation	11
2.2.2 Stereo-Electroencephalography	14
2.2.3 Other Procedures	17
3 Robotic Neurosurgery	21
3.1 Robotic assistant: neurosurgeons' expectations and demands	21
3.2 State-of-the-art robotic systems	23
3.2.1 Specifically for Stereotactic Neurosurgery	25
3.2.2 General and capable of Stereotactic Neurosurgery	34
3.3 Current perspectives and Future directions	38
4 Industrial Manipulator Adaptation	41
4.1 Robotic System Role	41
4.2 Setup description	42
4.2.1 Motoman MH5	43

4.2.2	Middleware	44
4.3	Control Application for Neurosurgery	45
4.3.1	Communication Module	45
4.3.2	Registration	46
4.3.3	Coordinate Convention	47
4.3.4	Robotic Motions	51
4.3.5	User Interface	51
4.4	Procedure Workflow	53
4.5	Discussion	55
5	Modular Architecture	56
5.1	Monolithic vs Modular Paradigm	56
5.2	Component-based Software Engineering	57
5.2.1	Concepts	57
5.2.2	Alternative Robotic Frameworks	58
5.2.3	OROCOS Project	61
5.3	Modular Surgical Robots	64
5.4	Design of a Surgical Robot Controller	65
5.5	Architecture Layout	67
5.6	Setup description	67
5.6.1	KUKA LBR iiwa 7 R800	68
5.6.2	Other equipment: Micro-drive and Designed tools	71
5.7	Simulation Setup	73
5.7.1	Virtual Equipment Models	74
5.7.2	Developed Plugin	76
5.8	Task Description	78
6	Developed Components	83
6.1	Cartesian Trajectory Controller	83
6.1.1	Geometric Path	84
6.1.2	Velocity Profile	88
6.1.3	Trajectory	89
6.2	Kinematics and Statics Module	90
6.2.1	Kinematics Solver - Problem and State-of-the-art	90
6.2.2	Analytic Kinematics	93
6.2.3	Inverse Kinematics	98
6.2.4	Limits and Singularities	100
6.2.5	Redundancy Resolution	106
6.2.6	Component's Operation and Other Functionalities	111
6.2.7	Statics Solver	117

6.3	Joint Trajectory Controller	118
6.3.1	Point-to-Point Library	118
6.3.2	Motion-through-Points Library	121
6.4	Robot State and Transformations	124
6.5	Task Supervisor	126
6.5.1	Action Interface	126
6.5.2	Robot Management	128
6.5.3	Safe Trajectory Execution	128
6.5.4	Trajectory Approach	130
6.5.5	Nullspace transitions	131
6.5.6	Motion Queuing and Error Handler	131
6.6	Fast Research Interface	133
6.6.1	Robot FRI	133
6.6.2	V-REP FRI	135
6.7	Registration	136
6.8	Surgery Plan	137
6.9	Procedure Coordinator	138
6.10	Graphical User Interface	140
7	Calibration and Application Accuracy	143
7.1	Kinematic Convention and Calibration	143
7.1.1	Features Identification	144
7.1.2	Parameter Extraction	148
7.1.3	Experiment and Results	154
7.1.4	Discussion	156
7.2	Application Accuracy	157
7.2.1	Phantom Design	159
7.2.2	Phantom Calibration	159
7.2.3	Marker Placement	163
7.2.4	Precision Assessment	164
8	System Diversification	167
8.1	Quantification and Tracking of Brain Deformation	167
8.1.1	Stereo vision system	169
8.1.2	Materials	174
8.1.3	Methods	176
8.1.4	Results	180
8.1.5	Brain Data Acquisition	182
8.1.6	Discussion	183
8.2	Haptic Feedback Tele-operation	184

8.2.1	Force Dimension Omega 7 Driver	185
8.2.2	Haptic Control Modes	187
9	Conclusions and Future Work	189
	Bibliography	215

List of Figures

2.1	Talairach proportional grid system based on the AC-PC line.	6
2.2	BRW localizer, frame and phantom-base.	9
2.3	Steps in the procedure for Deep Brain Stimulation surgery.	13
2.4	Steps in the procedure for Stereo-Electroencephalography.	16
3.1	ISIS Robotics SurgiScope.	25
3.2	Renishaw-Mayfield NeuroMate	27
3.3	Prosurgics Pathfinder.	28
3.4	Renaissance MARS.	30
3.5	RONNA G3.	31
3.6	Robocast project robot.	32
3.7	MedTech Rosa.	33
3.8	NeuroBot.	36
3.9	neuroArm	38
4.1	Customized surgical instrumentation.	42
4.2	Working scenario with the Zamorano-Dujovny phantom device to simulate the target, and Yaskawa Motoman MH5 robotic arm.	43
4.3	Motoman MH5 robotic arm, NXC100 controller and programming pendant (different working modes).	44
4.4	YARP message format, containing a key-command field, a text field, floating and integer parameters, and finally an error code (Machado et al., 2012).	44
4.5	Point based co-registration process. A “pivot” end-effector tool is used to touch four selected divot markings on the reference ring, sequentially.	47
4.6	Representation of a surgical trajectory reference frame extracted from the target and entry points.	48
4.7	Example of ZD coordinates with the stereotactic frame mounted in position 0°.	50
4.8	Graphical user interfaces of the control application.	52
4.9	Performing assisted drilling, with orientation and depth-control.	54
4.10	Holding microdrive to position electrode at the surgical target.	54
5.1	Component representation according to UML.	58

5.2	TaskContext state diagram.	62
5.3	Schematic representation of a TaskContext.	63
5.4	Schematic representation of the Modular Architecture developed.	68
5.5	KUKA LBR Med 7 R800 model.	69
5.6	KUKA Sunrise.Connectivity FRI overview.	70
5.7	Connectivity.FRI session state diagram.	71
5.8	Leibinger electrode micro-drive.	72
5.9	Designed end-effector for stereotactic surgery.	72
5.10	4-button debounce PCB.	73
5.11	Operating room virtual scenario created in V-REP.	74
5.12	Virtual representation of the developed end-effector.	75
5.13	KUKA LBR Med model added to V-REP.	76
5.14	Image of the real and the hollow robot with a coincident base frame and with different poses.	77
5.15	OROCOS components embedded in the developed V-REP plugin.	78
5.16	Virtual representation of the surgical trajectories in the V-REP scenario.	79
5.17	Concept of probe guide end-effector with actuated depth control.	79
5.18	Concept of trepan guide end-effector with actuated depth control.	79
5.19	Diagram of Stereotactic Neurosurgery assistance tasks performed by the robotic system.	80
5.20	Relevant procedure points defined along the entry-target trajectory. In the target-entry direction and from the target point one finds the tool point at a distance of d_{tool} . In the target-entry direction and from the entry point one finds the approach point at a distance of d_{app} , the tool-switching point at a distance of d_{tool} and the skull drill point at a distance of d_{drill}	81
6.1	Cartesian Trajectory Controller component and interfaces.	84
6.2	The required start and end frames are represented with larger coordinate frames. The smaller coordinate frames represent the intermediate frames which the end-effector should pass through.	85
6.3	Example of arc generated with the variables specified in equation (6.1).	85
6.4	Example of arc generated with the start, end frame and the reference point.	86
6.5	Arc trajectory examples in orthogonal planes for easier visualization. Both the arc radius, angle, plane and end-effector orientation can be finely tuned.	87
6.6	Example of a composite geometric path with way-points marked as larger frames.	88
6.7	The plots depict segments of the same path length (0.5 m) with different start and end velocity conditions, to show how the different velocity profiles adapt to create an overall trapezoidal shape.	89
6.8	Kinematics and Statics Module component and interfaces.	90

6.9	Manipulator generic structure, joint variables and DH frames assigned. The 7-DoF manipulator model of LBR iiwa 7 R800 from KUKA AG is used to depict the shape of an anthropomorphic arm without offsets.	94
6.10	The “real manipulator” depicts a 7-DoF manipulator at an arbitrary pose. The “virtual manipulator” is the non-redundant (6-DoF) replica of the same manipulator reaching the same pose. The real manipulator is transformed into the virtual manipulator by blocking the 3 th joint at zero ($\theta_3 = 0$).	95
6.11	Representation of the virtual manipulator and variables required for the θ_2^v calculation.	97
6.12	Example of joint limits and singularity intervals inscribed in the elbow redundancy circle for joint i . The $\psi(\theta_i^l)$ and $\psi(\theta_i^u)$ represent the arm angles where the respective joint limits are met, and ψ_{sing} is a singular arm configuration. An avoid interval is set next to the singular point with δ representing the unilateral spacing.	101
6.13	Pivot joint $\theta_i(\psi)$ profile with stationary points.	103
6.14	Pivot joint $\theta_i(\psi)$ profile without stationary points.	103
6.15	Pivot joint $\theta_i(\psi)$ profile when a singular arm angle exists.	104
6.16	Hinge joint $\theta_i(\psi)$ profile when the configuration parameter is positive, function defined in $[0, \pi]$. If only the configuration is changed to a negative value, the function profile is mirrored relative to the x-axis.	105
6.17	Hinge joint $\theta_i(\psi)$ profile when the arm is at a singular configuration.	106
6.18	General control scheme for inverse kinematics resolution with joint limits and singularity avoidance.	107
6.19	Example of intersection of feasible intervals from all the joints. The three resulting intervals are enumerated $m \in \{1, 2, 3\}$	107
6.20	Change of the arm angle as a function of $\psi_{(t-1)}$, defined in the feasible interval $\Psi_{all,m}$	108
6.21	Variation of the arm angle (ψ) and joint angles along the trajectory.	110
6.22	Example of standard trajectory with feasible intervals computed from the method presented in subsection 6.2.4, and after applying the interval adjustment algorithm.	113
6.23	Singularity avoid interval as a function of the “nullspace proximity” parameter.	114
6.24	Manipulability ellipsoid of a KUKA LBR iiwa R800 kinematic structure at $\theta^c = [0.0, 15.0, 0.0, -60.0, 0.0, 30.0, 0.0]$ degrees. The shape of the ellipsoid represents the relation between the joint and end-effector velocity. It is more elongated along the axis where the same joint velocity translates to a higher end-effector velocity.	115
6.25	Joint Trajectory Controller component and interfaces.	118
6.26	Type II Joint Trajectory Generation Algorithm with the input parameters at the left side (green, yellow and red) and the output parameters at the right side (blue).	119
6.27	Effect of the time synchronous property exemplified with the position and velocity progression of the shoulder joints (θ_0 , θ_1 , and θ_2).	120
6.28	Example of the shoulder joints (θ_0 , θ_1 , and θ_2) following a trajectory generated from a sequence of points using the MTP library.	124

6.29	Robot State component.	125
6.30	Transformations component.	125
6.31	Task Supervisor component.	126
6.32	Decision process to guarantee a feasible trajectory.	129
6.33	Set of poses that align the end-effector frame ($\{E\}$) z-axis with the vector created from the two input points ($\mathbf{p}_1$ and $\mathbf{p}_2$). The number of redundant poses n depend on the sampling step (κ). For clarity, only the z- and x-axis of the test poses are represented in the picture.	130
6.34	Task Supervisor motion state machine.	132
6.35	Robot FRI component.	134
6.36	Representation of the last joint position (x_0) and the 4 preceding values.	134
6.37	V-REP FRI component.	135
6.38	Registration component.	136
6.39	Surgery Plan component.	137
6.40	Procedure Coordinator component.	138
6.41	Direct control window.	140
6.42	Registration window.	141
6.43	Surgery plan.	141
6.44	Procedure window.	142
7.1	Example of acquired point ($\mathbf{p}$) contained in the plane defined by the last joint axis ($\mathbf{n}_n$) and the position of the flange. The acquired point must not be contained in the line that passes through the last joint axis, i.e. $\mathbf{p} \neq \mathbf{p}_\emptyset$	145
7.2	The relationship between two lines in space can be parameterized by the shortest distance between lines d (dual part), and the projected angle between the lines γ (real part) (Ketchel and Larochelle, 2006). The common normal is, $\hat{\mathbf{e}} = \ \mathbf{n}_1 \times \mathbf{n}_2\ $. The distance between lines is, $d = (\mathbf{c}_2 - \mathbf{c}_1) \cdot \hat{\mathbf{e}}$. The angle between lines can be determined from, $\gamma = \text{atan2}(\ \mathbf{n}_1 \times \mathbf{n}_2\ , \mathbf{n}_1 \cdot \mathbf{n}_2)$	149
7.3	The line relationship categorization process.	151
7.4	Example of the robot motion that generates the tracked point trajectory for the second joint. The thicker lines represent the position of the tracked point acquired relative to the robot base frame. The thinner lines represent the best fitting circle to the trajectories. . .	155
7.5	Acquired point trajectories. For each revolute joint, the motion axis is represented at the center of rotation.	156
7.6	Phantom box concept.	160
7.7	Assembled Phantom Box.	160
7.8	Pre-imaging procedure of the phantom.	161
7.9	Interlock point.	161

7.10	Phantom meshes after registration with the interlock calibration (red) and the ICP (blue). The CT phantom point cloud is represented by gray-colored dots and the model point cloud is either represented by red-colored dots for the interlock calibration, and blue-colored dots for the ICP calibration.	163
7.11	Model stalks locally registered to the stalks in the CT dataset. Model stalks' outline is represented by the blue dots while the real stalks from the CT image are represented by a gray shape/dots.	164
7.12	Marker placement equipment.	164
7.13	Marker placement setup and example.	165
7.14	Post marker placement CT image scan of the phantom.	165
7.15	System accuracy results. The model stalks (blue outline) are registered to the image stalks (gray dots) and the target, 5 mm above each stalk tip, is represented by the red color while the center position of each marker is identified by a green dot and a black outline.	166
8.1	Schematic representation of an Extended Modular Architecture.	168
8.2	Stereo vision system setup. Stereo camera fixed on a bracket, connected to a computer via USB 2.0.	170
8.8	Static Analysis - Detection error (translation and rotation component) of an immobile object for each test set.	181
8.9	Dynamic Analysis - Tracking error of a moving object across all test sets. Low, medium and high linear and angular speeds depicted in left and right figures. Statistically significant difference of median translation error among all angular speeds. Statistically significant difference of median rotation error among all angular speeds, except between low and medium angular speeds.	181
8.10	Time and Frequency analysis of brain motion in the 3 acquisitions, 25, 33 and 58 seconds. Left plots depict the displacement of the cloud of features' centre of mass throughout the acquisition. Continuous component of the signal was subtracted. Right plots represent the Fast fourier transform (FFT) analysis of the displacement of the cloud of features' center of mass.	182
8.11	Force Dimension Omega Driver component diagram.	185
8.12	Representation of the OmegaOrientation variable.	186
8.13	Haptic Control Modes component diagram.	187

List of Tables

3.1	Most successful robotic systems and projects directed at stereotactic neurosurgery. . . .	24
4.1	Yaskawa Motoman MH5 robotic arm specifications.	43
4.2	Target point ($\mathbf{p}_t$) Cartesian coordinates from ZD coordinates in each of the frame mounting positions.	49
5.1	KUKA LBR iiwa / Med 7 R800 robotic arm specifications	69
6.1	Classic Denavit-Hartenberg parameters of a 7-DoF anthropomorphic manipulator. . . .	93
6.2	Descriptive statistics of computational times recorded of different parts of the algorithm, measured from 10000 samples, in microseconds (μs).	109
7.1	Initial, start and end joint coordinates used during the trajectory execution. For readability, the joint positions are displayed in degrees, except for the Cobra's θ_3 that is displayed in millimeters.	157
7.2	Denavit-Hartenberg parameters for the studied robots. The measured Denavit-Hartenberg parameters ($a_{\text{meas},i}$, $\alpha_{\text{meas},i}$, $d_{\text{meas},i}$ and $\theta_{\text{meas},i}$) and the differences to the nominal values ($a_{\text{error},i}$, $\alpha_{\text{error},i}$, $d_{\text{error},i}$ and $\theta_{\text{error},i}$) for each joint are presented. The a and d parameters are represented in millimeters, the α and θ in degrees. Values truncated at $1e - 9$. . .	158
8.1	Test Performed.	179
8.2	Registration residual errors.	180

Chapter 1

Introduction

The new technology introduced to the robotic surgical field should conform to the three iron pillars of the health-care system: quality, cost-effectiveness, and accessibility. The constant search for new and superior medical technology is primarily driven for the sake of improving health-care quality. The quality principle can be objectively measured by the precision of the tool placement, the incidence of surgical and long-term complications, duration of operative times and so on. Most published reports on research systems and commercially available products tend to present their results in terms of accuracy, effectiveness, and safety against the traditional approach or other homologous systems.

On the other hand, cost-effectiveness and accessibility fall into a secondary objective, which is evident through the lack of studies that assess the immediate and/or long-term cost-effectiveness of the advertised technology. A recent report on a retrospective cohort study involving 416 hospitals and 24000 patients who underwent radical nephrectomy between 2003 and 2015, revealed a trending increase on the use of robotic-assistive technology with higher hospital costs averaging more \$2700 per patient (Jeong et al., 2017).

Fiani et al. evaluated the impact of robot-assisted neurosurgery on the quality of the health-care service and the immediate economic burden using as an example the MAZOR SpineAssist robotic system (Fiani et al., 2018). Considered one of the most affordable spinal surgery systems approved by FDA, it still has setup costs that amount to \$550,000 including an annual maintenance service that costs about 10% of the list price, consumables, and implants that amount to \$1500 per case. One can argue that the faster operating times, reduced number of intra/postoperative complications or the decrease in technical support and maintenance of operative materials may result in long-term savings for the medical center.

The reality is that surgical robotic systems have considerably high up-front costs for the medical institution with no immediate investment return, which makes it only available to an exclusive list of first world health-care providers. Two possible strategies to cope with this problem involve:

1. simplification and acceleration of the surgical robot development process;
2. endowing the surgical robot with a versatile control system.

It naturally follows that new technology draws more diversity to the robotics market, which eventually leads to more accessible robotic solutions. Additionally, systems equipped with the right tools can be reassigned

to multiple procedures, thus expanding its potential and balancing the cost-efficiency scale.

It is also true that transferring surgical devices into the operating room represents a difficult enterprise (section 3.3). Since the first surgical robot, three decades ago, many prototypes have been built, several validated but only a few are currently applied in real surgical scenarios. Sanchez et al. (2014) list 3 factors that contribute to this reality:

- limited access to research and operation on living organisms;
- difficulty to start a business in today's medical industry;
- lack of adequate design methodologies to facilitate the technological transfer, specifically in robots developed in research centers.

Other not so obvious deterrent is the limited access to previous technology as a result of either: commercial products whose information is not disclosed, or projects conducted in private repositories by research centers.

Finally and to preface the rest of the document, it worth noting that the theme and motivation for this project materialized from a direct interaction with the end-users, the Neurosurgery team of the Centro Hospitalar e Universitário de Coimbra.

1.1 Contributions

Since the inception of this project, we soon realized the multi-disciplinarity, complexity, and human/financial resources required for the development of a complete robotic system towards robotic neurosurgery. This thesis will focus on the process and development of an high-level control architecture for robotic systems for neurosurgical applications. Our main goal is precisely the development of a modular, distributable and flexible control system to facilitate the technology transfer of neurosurgical robotic systems.

The thesis main contributions are:

1. Identification and comparison between the main imaging and tool guidance technology applied to stereotactic neurosurgery;
2. Revision and comparison of the key robotic systems for neurosurgical applications, detailing the features common among these systems and putting them in line with user expectations (Faria et al., 2015a);
3. Adaptation of an industrial manipulator and implementation of a control system to operate as a tool guidance device and directly replace the stereotactic frame (Faria et al., 2016);
4. Design and implementation of a modular high-level control architecture for neurosurgical robots with hard-real-time support, deterministic behavior and well-defined component and communication model;
5. Creation of a virtual scenario and virtual robot as a replica of a neurosurgical operating room with a real robotic system that is controlled by the same high-level control architecture;
6. Development of an analytic inverse kinematics method for redundant manipulators capable of avoiding joint limits, singularities and uniquely solving the nullspace (Faria et al., 2018);

7. Development of a kinematic calibration method to determine the real Denavit-Hartenberg parameters of a serial manipulator model with variable kinematic structures and joint types/dispositions (Faria et al., 2019);
8. Development of an in vitro method to assess the system's application accuracy based on a phantom with several distributed targets to mimicking the human brain;
9. Development of two extension modules to the proposed high-level control architecture, to feature teleoperation and to include a stereo-vision system capable of tracking and quantifying the brain surface movement in open-skull neurosurgery (Faria et al., 2014).

The source code repository with the material developed in the context of this thesis can be found at <https://github.com/neuebot/NeueBot>.

1.2 Outline

This thesis is organized in 9 chapters. In this section, we briefly elaborate on the structure of the document and outline the key points of each chapter.

In chapter 2 we introduce the topic of Stereotactic and Functional Neurosurgery. As the surgical application to the robotic system in development, it is fundamental to describe not only the technology involved but also the methodologies, benefits, and limitations. It is listed various imaging modalities applied to stereotactic and functional neurosurgeries, as well as the different stereotactic frames historically and currently used. The chapter finishes with a description of several neurological ailments that are indicated for stereotactic neurosurgery, relating them to procedural variations.

Although most of the technology applied to stereotactic neurosurgery is listed in chapter 2, the category of neurosurgical robots deserved its chapter, chapter 3. This chapter serves to introduce the theme of robotic systems for neurosurgical tasks, the neurosurgeons' expectations and demands, current perspectives and future directions. At the core of the chapter, we enumerate the key players in surgical robots for neurosurgical applications, going into detail about their main features and how they can be perceived as advantages or disadvantages. To limit the scope of the listed systems, we chose to include systems that either obtained clearance for clinical trials or are commercially available.

After the introductory chapters, in chapter 4 we elaborate on a simple robotic system solution to operate as a direct replacement to the stereotactic frame. A 6-DoF industrial robotic manipulator with suitable repeatability, reach and compactness is adapted to perform a surgical procedure. It is described how the different system modules were implemented to directly handle the surgical coordinates and the end-effectors designed and fabricated to interface with the surgical instrumentation.

In chapters 5, 6, 7 and 8 we present and detail on a modular high-level control architecture, an alternative to the monolithic solution from chapter 4. Chapter 5 starts by describing the differences between both approaches and how they affect system development, technology transfer and ultimately the cost and accessibility of surgical robots. The concepts of Component-Based Software Engineering are explored together with a list of frameworks and libraries to implement modular control software for

robotic applications. These ideas crystallize as system design decisions and the architecture layout. This chapter also contains the real and virtual robotic setup descriptions based on a 7-DoF anthropomorphic manipulator. Finally, we converge on the task description, which specifies the robotic system role throughout the procedure.

In chapter 6 we go into detail about the system software components that amount to the proposed high-level control architecture. The control architecture divides into two layers, the Core and the Application Layer. The Core components include controllers for the robot motion in task and joint space, as well as new kinematic solving module specially design for safe and deterministic operation of redundant manipulators. The Application components bridge the user-interface with the core layer to provide the expected functionality for the neurosurgical task.

The setup introduced in chapter 5 and detailed in chapter 6 is calibrated and tested in chapter 7. This chapter is divided into two parts, the first-named 'Kinematic Convention and Calibration' where it is proposed and tested a Denavit-Hartenberg parameter detection method to determine the real robot parameters and thus improve its precision. The method was proposed and tested in 6 serial manipulators with distinct kinematic dispositions. The second part of the chapter describes a method to assess the robotic system's application accuracy. It relies on a cost-effective phantom device with several targets and stereotactic instrumentation to drive small markers to the target location. The precision is measured with a CT imaging scan as it provides better spatial resolution and correctness.

One of the advantages of a modular architecture is the possibility to branch out the system to alternative applications. Two extensions to the high-level controller are proposed in chapter 8, one module to track the brain surface is presented as a system extrapolation for open-skull surgeries; and a teleoperation module was implemented and tested for a potential application in kidney stone removal surgery.

Chapter 2

Stereotactic and Functional Neurosurgery

The surgical act on itself is one of the most complex tasks performed by humans with the objective of intrusively diagnosing, treating a pathological condition, or improving any bodily function. In particular, the field of neurosurgery has been historically considered an observation realm and labeled “*primum non nocere*”, as the surgical outcome often deteriorated the patient condition even further. The current state of neurosurgery is a culmination of centuries of advances in a wide range of science and medicine areas ranging from anesthesia, antisepsis, hemostasis, instrumentation and more recently of imaging technology (Ormond and Hadjipanayis, 2014).

This thesis will focus on a rather contemporary sub-group of general neurosurgery called ‘Stereotactic and Functional Neurosurgery’. This chapter in particular aims to introduce the core concepts involved with this surgical procedure.

2.1 Stereotactic Imaging and Technology

The stereotactic method was first described by Robert Clarke and Victor Horsley in 1906, as a system to define the brain volume in Cartesian coordinates and to access subcortical structures using a 3D positioning device. Lacking the imaging technology to identify intracranial structures, their system did not see much use beyond in vitro animal testing (Horsley and Clarke, 1908). It was not before 1947 that Spiegel and Wycis noted the applicability of stereotaxy in subcortical functional neurosurgery procedures. They pioneered the development of methods to determine stereotactic coordinates based on anatomical landmarks (Spiegel et al., 1947). These findings were mostly based on imaging techniques such as ventriculography (Dandy, 1918) and angiography (Moniz et al., 1931). In 1953, Talairach reported the stereotactic angiography, the first imaging technique to allow direct visualization of intracranial lesions (Talairach et al., 1954). Additionally, and to cope with the limited applicability of contemporary visualization technology Talairach and Tournoux developed a consistent and reliable method to indirectly locate anatomic cerebral structures. Their system relies on easily detectable anatomic references, the intercommisural line that passes through the superior edge of the Anterior Commissure (AC) and the inferior edge of the Posterior Commissure (PC). The AC-PC line establishes the foundation for the proportional grid system, an orthogonal system of lines parallel

and perpendicular to the AC-PC that divides the brain in parallelograms, Figure 2.1. It takes into account the skull size and is based on a landmark with definite relationship to gray central nuclei and sufficiently precise relationship to the telencephalon, the AC-PC line (Talairach and Tournoux, 1988; Benabid et al., 2009). The Talairach coordinate system is one of the first and certainly the first widely accepted Atlas to describe the neural architecture of the human brain, being particularly useful to locate structures not visible by traditional radiologic methods.

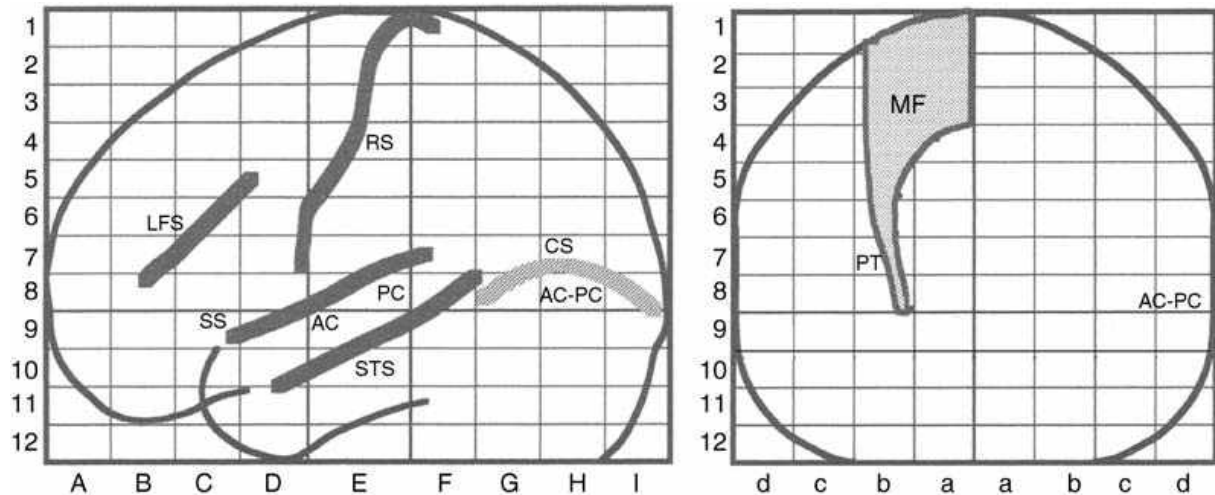


Figure 2.1: Talairach proportional grid system based on the AC-PC line. Other structures located using grid parcellation: SS, sylvian sulcus; STS, superior temporal sulcus; LFS, lower frontal sulcus; RS, rolandic sulcus; CS, calcarine sulcus; MF, motor fibers; PT, pyramidal tract; upper diagram, lateral; lower diagram, anteroposterior (reprint copyright license agreement with Springer Nature).

Thenceforth, the field of Stereotactic Neurosurgery has witnessed continuous expansion, providing remarkable non- or minimally-invasive procedures to accurately localize and treat lesions or abnormal tissue.

2.1.1 Stereotactic Imaging

The developments on stereotaxy have been spearheaded by the advent and development of imaging technology. As of today, the stereotactic and functional surgery planning relies on a wide spectrum of imaging exams. The most common modalities are Computed Tomography (CT) and variations of Magnetic Resonance Imaging (MRI, MRA, MRV, fMRI, etc). Other imaging sources such as Positron Emission Tomography (PET), Single Photon Emission Computed Tomography (SPECT) and ultrasound are also employed (Gorgulho et al., 2009).

Computed Tomography is a radiological imaging exam that uses X-rays to acquire axial slices of a defined volume. CT scans are relative fast, when compared to MRI for example and enjoy high accuracy and low distortion. On the other hand, the exam exposes the patient to a significant dose of radiation, and the brain structures cannot be differentiated due to the low contrast of brain tissue (similar Hounsfield units). Although some lesions are visible in pre-operative CT, for the most part, CT-guided stereotaxy is predicated on a brain atlas and the relation to ventricular landmarks, such as the AC-PC line (Hariz and

Zrinzo, 2009; Schulder and Oubré, 2009).

Anatomical and probabilistic functional atlases have been historically used to support stereotactic planning, whenever direct visualization of target structures is not possible or when the exam introduces significant distortions. Parallel to other advances in stereotactic technology, also the brain atlas has been evolving to better predict anatomical variability, deliver a high-resolution volumetric model that builds on functional population-based atlases (Nowinski, 2009).

Magnetic Resonance Imaging uses strong electromagnetic fields to polarize matter at an atomic level. The scanned material is differentiated by determining the rate at which the atoms return to the equilibrium state. MRI provides high discrimination of brain structures, which allow the images to be used as the patient's personal brain atlas. Moreover, by modifying the properties of the polarizing signal - MRI sequences (T1-, T2-weighted, FLAIR, proton density, etc) - the radiologist may enhance specific tissues or structures in detriment of others. It is recurrent practice to use radiocontrast agents to enhance the contrast between brain tissues both in MRI (gadolinium contrast) as well as in CT (iodine-based contrast) studies. On the negative side, MRI imaging is susceptible to distortions caused by external electromagnetic fields, which provoke geometric inaccuracies (Hariz and Zrinzo, 2009; Schulder and Oubré, 2009).

The surgical planning of today's image-guided neurosurgery often relies on multimodality imaging studies, harnessing the advantages of each technique in order to accurately reach the target structures and avoid surrounding vasculature or eloquent brain regions. Combining different radiological datasets involves a method called "image fusion". Each image set has its own 3D coordinate system that needs to be registered into another to create a coherent output.

In order to match a set of homologous points in two reference frames, one needs common references, such as: anatomical references/surfaces/volumes, skin fiducial markers or bone implanted markers. These markers must be simultaneously visible in MRI/CT imaging and during the actual procedure in the perioperative space by a localizer system (Hartov and Roberts, 2009).

Not delving in exhaustive details, one can split the registration methods into:

- Point Based Co-registration, a set of reference points are located in both spaces, and an algorithm based on singular value decomposition (SVD) or a variant of iterative closes point (ICP) is employed to match point-pairs. This simple and robust strategy, is perhaps the most widely used approach.
- Surface Based Co-registration, per-definition relies on capturing, representing and matching surfaces. Anatomical surfaces are digitized using contact sweeps, laser scanners or stereoscopy, into a set of points that form a mesh. The transformation between the two surfaces is computed using an ICP algorithm.
- Volume Based Co-registration, is a method to calculate the spatial transformation between volumetric datasets based on similarity and mutual information. The transformation results from an optimization of the geometric relationship between neighbor image elements. These relationships are predicated in physiological assumptions.

2.1.2 Stereotactic Technology

Stereotactic neurosurgery is closely related to the stereotactic frame (Gildenberg, 1998). All stereotactic procedures revolve around two core principles: the first being the necessity for an image to physical space coordinate mapping, and the second involves an extracranial device to execute the positions and motions with precision and steadiness. Occasionally, some systems may include a calibrated phantom device to assess the positioner settings mechanically, Figure 2.3b.

Stereotaxy must produce a reliable coordinate system that can be associated with anatomical landmarks, creating invariant references that bypass human variability and device placement asymmetries. Per-standard, it became clear that the system's spatial accuracy should fall within 1 mm (Maciunas et al., 1994). The following paragraphs list some of the most successful and distributed stereotactic apparatus, explaining the fundamentals and main features (Schulder, 2003; Lozano et al., 2009).

Todd-Wells, BRW, and CRW Frames can be considered versions of a stereotactic frame named after the developer physicians in collaboration with the common element, the engineer Trent Wells. These frames are intimately associated with the evolution of stereotactic surgery in the US, and the CRW Precision (Integra LifeScience, Cincinnati, Ohio) is still one of the most widely used stereotactic frame systems.

The Todd-Wells apparatus relied on X-ray technology to achieve a below 1 mm spatial accuracy. The stereotactic system is directly dependent on this imaging technology and as such, the designed system establishes a strategy to fixate the lateral and anterior-posterior X-ray sources within the operative room and fixes the patient's head at a precise focal point. The frame concept derives from a burr hole mounted system with an arc, that was modified to hold a head. This created the very first modular human stereotactic instrument with two arcs to theoretically access a target from any direction. One fundamental downside of the system had to do with the difficulty in reaching low lateral trajectories.

The Brown-Roberts-Wells (BRW) frame was conceived as the imaging paradigm shifted with the introduction of CT-scans. The potential to acquire digital information in all three axes motivated the redesign of the stereotactic devices to cope with this new information. A CT-compatible apparatus was designed together with a CT localizer scheme to relate each axial slice to the frame. A localizer ring is attached to the frame base and supported by "N-pattern" rods, Figure 2.2. The vertical depth "Z" of each slice is then computed from the unique distances between the nearly circular cross-section of each rod. Consequently, the axial plane orientation can be precisely determined against the frame coordinates which creates a uniquely defined map of X, Y and Z coordinates within the volume inscribed by the frame and localizer ring (Mozdy, 2017). This system is more versatile than the Todd-Wells apparatus but relies on a counter-intuitive system of four coordinates to adjust the probe angulation. Adjusting the frame settings to a particular trajectory proved to be a lengthy process for the available computational resources.

The Cosman-Roberts-Wells (CRW) frame, suggested by Eric Cosman in 1980, promised to fix some of the hindrances associated with the BRW design. The frame introduced new features, some of them inspired in other successful frame designs such as, the "center-of-arc" from the Leksell arc-radius system, and the "phantom-base" verification device from the Riechert-Mundinger system. The frame arc was now



Figure 2.2: BRW “N-pattern” localizer and stereotactic frame mounted on a “phantom-base” verification device (Mozdy, 2017).

mounted on a translational system that much abbreviated the calculations required during the surgery planning. The frame base changed to a square-shape with slots to hold the trunion apparatus evenly in the lateral and anterior-posterior direction. Since its debut, the CRW frame has been updated in materials and accessories several times, shipping with its own planning software interface.

Leksell Frame is named after its creator Lars Leksell, one of the greatest contributors to the evolution of stereotactic and functional neurosurgery field. He devoted his career to develop surgical apparatus and less invasive techniques to drastically reduce the mortality and morbidity rates associated with contemporary neurosurgery. The Leksell stereotactic system (Elekta Instrument AB, Stockholm, Sweden) comprises a base ring where a semicircular arc is attached, which in its turn holds the instrument carrier. Perhaps one of the most unique and influential features of the Leksell frame is the fact that the arc center corresponds to the selected cerebral target. It means that the instruments are always directed towards the center, making it possible to adjust the angular configuration of the frame and change the entry point without affecting the target.

The Leksell apparatus contains a fiducial system that attaches to the base ring during the radiological image study. The fiducial system establishes a Cartesian space where any point can be referenced by the X (lateral), Y (anterior-posterior) and Z (vertical) coordinates. The Z coordinate is determined based on the distances of the circular-shaped slices of the “N-pattern” rods that mount on the base ring. The system includes fiducial plates compliant with X-ray, CT and MRI scanners. One important advantage of this system over others has to do with the slightly lower position of the base ring. Although the base ring is fabricated in a compatible material, it still generates image distortions and artifacts, which in this case are further away from the surgical field.

Over the last decade, the Leksell frame system has been more than once updated to accompany the new advances in imaging and instrumentation technology. Besides the listed components, the Leksell apparatus is accompanied by a wide range of accessories for diagnostic, therapeutic and functional brain surgery.

Riechert-Mundinger and Zamorano-Dujovny Frames are two of the most widely accepted and major distributed stereotactic systems, particularly in Europe. Commercialized up until the 1980s by the F. L. Fischer Company, both frames are currently maintained and sold by Inomed (Inomed Medizintechnik GmbH, Emmendingen, Germany).

As the previous listed stereotactic apparatus, these systems combine a base ring that attaches and holds the patient's head, an aiming bow that mounts on the base ring and an instrument holder. The main innovative feature of the RM and subsequently ZD frame was the introduction of the "phantom-base" to allow a stereotactic coordinate check. The target point simulator (phantom) consists of: i) a copy of the RM/ZD circular base ring, ii) a stylet rod with a pointed tip to simulate the trajectory target and iii) a system of 3 screws to control the Cartesian coordinates of the target, Figure 2.3b. Apart from the same base ring and phantom device, the RM and ZD systems are supplied with a wide variety of hardware, targeting instruments and share the same planning software. The RM/ZD systems are also compatible with the most common imaging modalities for stereotactic neurosurgery planning: X-ray, CT, and MRI. Finally, these systems share the same fiducial system based on "Triangular-pattern" rod plates, that similarly to other stereotactic systems, establishes a Cartesian coordinate space, Figure 2.3a. Each plate has a central rod normal to the base ring XY-plane, and two angulated lateral rods. These plates are mounted on the base ring and adapted according to the imaging technology.

Aside from the commonalities, there are two distinct features that set these systems apart:

- The RM translational system vs the ZD center-of-arc system. The RM frame works on polar coordinates that control both the arc position (α and β), the direction of the electrode (ϕ and ψ) and the electrode depth. On the other side, the ZD frame applies the center-of-arc principle meaning that once the target position is defined, one can change the electrode entry angulation without changing the ultimate target. The ZD arc position is adjusted using 3 mechanical sliders that control the X, Y and Z coordinates and define the target, and 2 additional arc sliders that define the entry angle D and E.
- The RM 3-point fixation vs ZD single point fixation. The RM frame is fixed to the base ring in 3 points providing higher stability to the system. The ZD frame attaches to the base ring using a single support point. While the latter may lose on stability and rigidity, it gains in simplicity and compactness. Additionally, the ZD system can be mounted in all 4 quadrants of the base ring, enabling a much more flexible surgical planning.

2.1.3 Frameless Technique

From a conceptual standpoint, the frame is only required due to a lack of a static transformation between the intracranial and extracranial space. It was in the later years of the 1980s that the concept of "frameless stereotaxy" emerged. It rests on the principle of real-time tracking and displaying the exact position of the instrumentation from the physical space into image space, where the surgery is planned¹. Three core concepts are involved: i) localizer system able to acquire the position of a marked instrument in the physical

¹Also known as Neuronavigation.

space; ii) registration technique to map the physical to the image space; and an iii) image display to show the intersection of the instrument and the radiological image in the image space.

The localizer system can either be based on precision articulated arms or emission-receptor technology. Articulated arms with or without coupled sensors are used to pinpoint point or surface landmarks identifiable in the image space, and thus enable point or surface based co-registration. By reading the encoders at each joint of the articulated arm, through Forward Kinematics, one can extrapolate the exact spatial position of the arm. Once registered, the arm is used to position and orient the surgical instrumentation. Alternatively, optical or electromagnetic markers that are both visible in image space and in physical space create the static transformation between intra- and extracranial space. This requires a parity of emission-receptors, being the markers the emission/reflector of optical radiation or electromagnetic energy. The markers' positions are determined in real-time using triangulation. The receptor or tracking system is continuously measuring the position of the markers used for image-registration, as well as any other marker attached to the surgical instruments. Depending on the registration method, it may be required a unique identification of each marker. To achieve the level of accuracy that approaches frame-based methods, bone implanted fiducials are applied.

Although the need for a stereotactic frame is withdrawn, the patient's head still needs to remain immobile throughout the perioperative procedure. If not a base ring, a Mayfield head clamp is commonly applied to firmly secure the head and thus avoid any possible complications that may arise from undesired head movements.

2.2 Stereotactic Procedures

Functional and stereotactic neurosurgery was originally indicated as a method to accurately lesion specific basal ganglia and thalamus structures, in selected patients afflicted by emotional, obsessive-compulsive or movement disorders (Gildenberg, 1998). The field continuous expansion alongside imaging and instrumentation technology culminated in new neuroscience discoveries, which in its turn begot a positive feedback to extend the technique applicability to a broader spectrum of procedures. In this section, we will describe in more detail two of the most iconic stereotactic procedures, Deep Brain Stimulation (DBS) and Stereo-Electroencephalography (SEEG), and summarily explain other operations that fall in the same category.

2.2.1 Deep Brain Stimulation

The electrical excitability of the human motor cortex was first reported in 1874 by Roberts Bartholow in the 'Experimental investigations into the functions of the human brain'. Since then, the positive feedback associated with electrical stimuli was used as an exploratory method to map cortical functions. It was not before the 1970s, that the first large series reports of the therapeutic effects of cerebellar chronic stimulation for treatment of pain, movement disorders, and epilepsy, were published (Perlmutter and Mink, 2006).

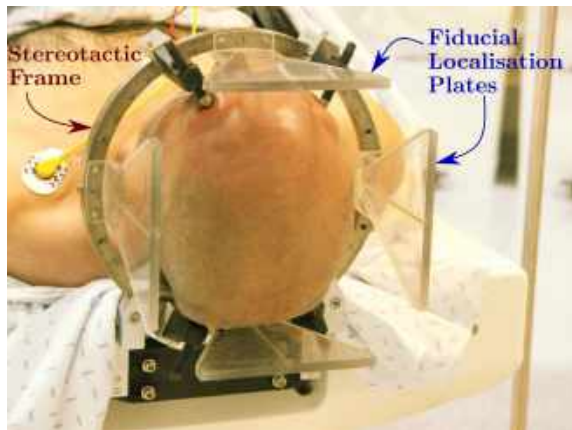
Lippitz and Siegfried (Siegfried and Lippitz, 1994) reported in 1982, a remarkable improvement in tremor symptoms in a Parkinson's disease patient being treated for pain, upon implantation of deep brain stimulating electrodes in the thalamus. This discovery encouraged him, Benabid and colleagues to further investigate into the chronic stimulation of the thalamus to treat parkinsonian tremor (Benabid et al., 1987b, 1991). The deep electrode implantation procedure was initially experimented in patients resistant to L-dopa medication that were selected for bilateral ventral intermediate nucleus (VIM) thalamotomy. The promising results of continuous stimulation associated with its reversibility and adaptability to control the side-effects established chronic stimulation of VIM as the preferable tremor treatment over thalamotomy (Benabid et al., 1996; Montgomery, 2010).

The Deep Brain Stimulation (DBS) procedure consists in applying controlled electrical pulses directly to deep brain structures through implanted electrodes linked to a neurostimulator (Benabid, 2003; Marks, 2010). The auspicious results in parkinsonian limb tremor reduction from VIM stimulation were naturally followed by analogous treatments that target other Parkinson's disease symptoms (Benabid, 2003; Abelson et al., 2005; Bronstein et al., 2011), or other disorders such as: Dystonia (Vidailhet et al., 2005; Kupsch et al., 2006), Tourette's Syndrome (Houeto, 2005; Mink et al., 2006), Depression (Mayberg et al., 2005), Obsessive Compulsive Disorder (OCD) (Abelson et al., 2005), etc. The differences in the DBS treatment for the previously listed disorders are the target structure and the stimuli itself.

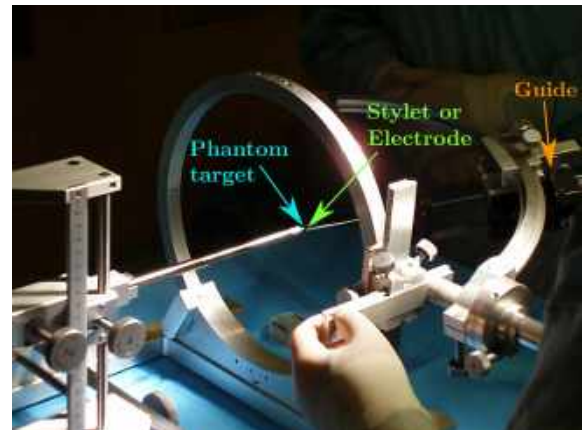
The brain structures to stimulate are usually found in the basal ganglia, being the most common targets: the subthalamic nucleus (STN) and the internal segment of the globus pallidus (GPi). The stimulus itself has several adjustable parameters that can be tweaked for optimal response: signal frequency, amplitude, pulse width and the number of electrodes and contacts (Dostrovsky and Lozano, 2002; Vitek, 2002; Anderson et al., 2003; Kuncel and Grill, 2004). The number of possible combinations easily raises to the thousands, although the majority of the patients in these studies responded well to a known and specific set of configurations. It is important to reinforce, that the best set of stimulation parameters are seldom definitive for any given patient case. On top of the anatomic variability, the imaging exams that precede the electrode implantation, do not provide a conclusive location of the basal ganglia nuclei. The target is usually defined by reference to visual landmarks (AC-PC line, mid-commissural point, etc.) and it is further refined by a combination of microelectrode recordings and intra-operative stimulation. As for the stimulation parameter selection, it is common for the patient sedation to be reduced so neurologists can directly assess the symptomatic alleviation and manifestation of side-effects for different stimulation parameters.

To better illustrate a typical DBS workflow, it is described as performed in the neurosurgery unit of the Coimbra Hospital and University Centre. The bilateral DBS surgery with microelectrode recording was conducted on a patient with Parkinson's disease. More information regarding DBS surgical technique can be found in Seijo et al. (2009); Starr et al. (2010); Perlmutter and Mink (2006); Machado et al. (2006).

Following the paradigm of Image-Guided Surgery (IGS), the patient initially undergoes a Magnetic Resonance Imaging (MRI) and/or any anatomic-functional imaging scan. On the day of surgery, after sterilizing the patient's head and attaching the stereotactic ring (Figure 2.3a), a Computed Tomography (CT) scan is taken. The MRI and CT scans are registered to the stereotactic space, i.e. – the transformational



(a) Pre-operative imaging with the stereotactic ring attached and four fiducial localisation plates.



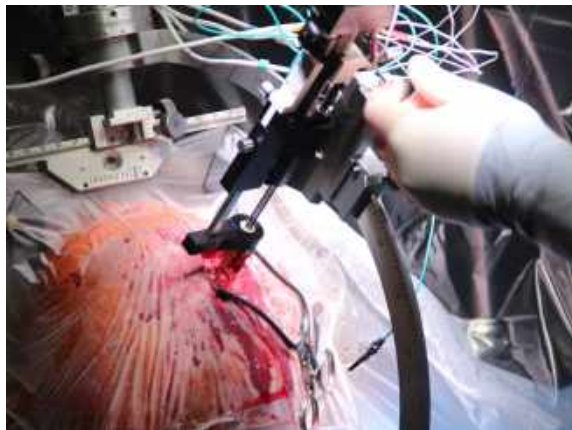
(b) The confirmation of the pre-operative coordinates using a phantom to simulate the target point.



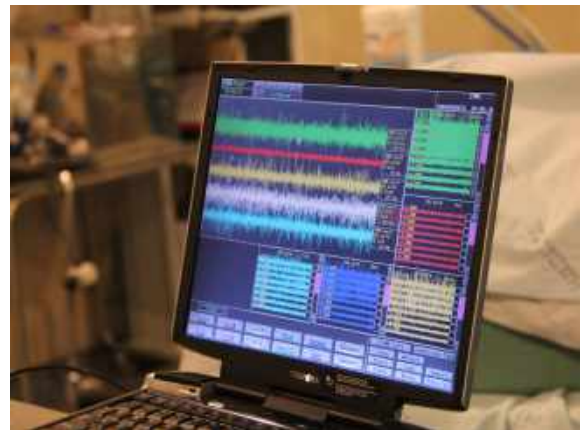
(c) Scalp incision.



(d) Skull drilling.



(e) Placement of multi micro-electrodes to register neurological signals and stimulate target structures.



(f) Micro-electrode recording and calibration of macro-stimulation parameters.

Figure 2.3: Steps in the procedure for Deep Brain Stimulation surgery.

relationship between the two three-dimensional spaces is determined (Galloway, 1998). Four fiducial localization plates, attached to the stereotactic ring during the CT scan (Figure 2.3a), allow for the calculation of the transformation between the image space, with reference to the anterior-posterior commissure line, and the stereotactic reference frame.

In the planning software, the medical team selects the target and entry points of the electrode insertion trajectory, avoiding vessels and ventricles. Based on the selected trajectories, the planning software computes the stereotactic coordinates for each electrode.

A phantom device is used in the operating room to visually confirm the stereotactic coordinates (Figure 2.3b). The phantom is attached to a stereotactic ring (similar to the one fixated on the patient's skull) and simulates the target point to be reached by the electrode. The stereotactic frame is mounted onto the phantom's stereotactic ring and is adjusted to the desired coordinates. A stylet is then placed in the stereotactic frame guide and if the stylet tip and the phantom tip are coincident the computed coordinates are confirmed.

The frame is subsequently removed from the phantom, placed in the patient's stereotactic ring and the stylet is used to mark the scalp entry point. Then the frame is moved aside to make the scalp incision and to drill the burr hole in the skull to access the brain (Figure 2.3c-2.3d). Afterward, the frame is adjusted once again in order to advance the electrodes/cannulas to the defined depth (Figure 2.3e).

Multi micro-electrodes are used to map the sensorimotor region by recording the neuroelectrical activity near the planned target. Initially, these electrodes are positioned along the planned trajectory with the aid of guiding cannulas, 10 mm to 15 mm before the target. After this, they are iteratively lowered – millimeter by millimeter – until they are positioned 5 mm from the target and then half a millimeter between iterations, recording the neuroelectrical signals during each step. In the end, the data recorded are analyzed to select the closest location to the sensorimotor region within the nucleus (Figure 2.3e).

The same recording micro-electrodes have a macro-stimulation lead, which is used to stimulate the previously located sensorimotor region. Following an iterative approach once more, the current and the depth of leads is increased (Figure 2.3e-2.3f). During each step, the team of neurologists qualitatively evaluates the patient's symptoms, seeking the best response and verifying side effects.

After finding the ideal electrode placement and stimulation signal properties, the micro-electrodes are replaced with a definitive quadripolar macroelectrode. Intra-operative imaging is used to check if the macro-electrode placement coincides with the micro-electrode position. The macro-electrode is later connected to an implanted pulse generator or neuropacemaker. If bilateral brain stimulation is required, all the intra-operative processes must be repeated for the other side. Due to the long duration of the procedure, the neurosurgeon may choose to implant the implanted pulse generator afterward or in the following day.

2.2.2 Stereo-Electroencephalography

Back in 1953, Talairach and Bancaud began a series of studies based on implanted depth electrodes in human patients. Advances in the stereotactic radiologic visualization of a number of brain structures permitted the neurosurgeon to establish a spatial coordinate system to map the brain, the stereotactic atlases. These atlases would gradually cover more telencephalic structures and concurrently new methods to reach these areas would be developed. Two decades of uninterrupted expansion culminated in a new surgical exploration methodology to investigate epilepsy, reported in 1973 by Talairach and Bancaud, the Stereo-electroencephalography (SEEG) (Talairach et al., 1962; Talairach and Bancaud, 1973).

Surgical treatment of refractory epilepsy presented indisputable evidence of success. The surgical procedure is to resect or to control-damage the epileptogenic zone (EZ). The success of surgical treatment is thus directly dependent on a clear segmentation of the EZ as well as the delimitation of the nearby eloquent brain regions, e.g. primary motor, sensory, speech, visual cortex, etc. The study of ictal symptomatology, the radiologic image data, and non-invasive electroclinical studies may present non-conclusive evidence of the EZ nor discard the proximity to functional areas.

The SEEG method relies on the introduction of several multi-contact “depth” electrodes to record the disturbed electrical activity of the brain in the vicinity of the supposed EZ. These electrodes provide a time accurate high-resolution information regarding the focus and propagation of the seizure onset. The methodology proposed by Talairach and Bancaud relied on angiography and X-ray images. To guarantee spatial coherency and correctness, images had to be calibrated to eliminate enlargement, displacing or distorting artifacts. This proved to be a complex and time-consuming process. Besides, the reported Tailarach stereotactic frame used to position the electrodes was limited to lateral orthogonal trajectories, which limits the implantation site workspace.

The emergence and advancement of new imaging techniques such as CT and MRI proved to be a major improvement in the surgery planning phase, by simplifying and shortening the process with inherent greater spatial resolution and accuracy. In parallel, new stereotactic frames designs, frameless methods, and even robotic actuated implantation have pushed the procedure’s precision and repeatability to its excellence standard, reduced the setup time and the discomfort to the patient, among other advantages (Kratimenos et al., 1993; Guenot et al., 2001; Ortler et al., 2011; Cardinale et al., 2013; Dorfer et al., 2014; Munyon et al., 2015).

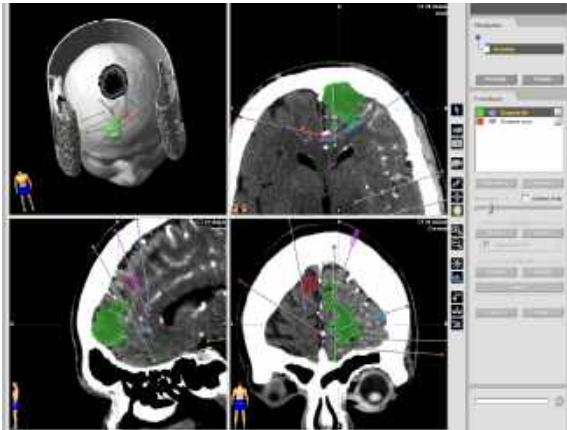
Even though the core principles behind the SEEG procedure are standard, different medical centers slightly differ in their approach in terms of imaging techniques, implantation setup or execution. To keep the document consistent with the previous Deep Brain Stimulation subsection 2.2.1, we will describe the SEEG surgical workflow as performed in Coimbra Hospital and University Center.

Once indicated for surgery, the patient first undergoes a gadolinium-enhanced MRI examination, including volumetric T1-weighted and volumetric T2-weighted fluid-attenuated inversion recovery (FLAIR) sequences for targeting. The collected image data sets are used in the surgery planning step to establish the electrode trajectories, which should reach around and beyond the putative epileptogenic zone while avoiding any considerable vases or important brain structures.

In the surgery day, a volumetric and contrast-enhanced CT scan is acquired of the patient’s head, with the stereotactic reference ring and fiducial localization plates attached. Both MR And CT image data sets are registered and the coordinates of the planned electrode trajectories are confirmed or adjusted, Figure 2.4a. Aside from the typical entry and target point, other metrics are recorded for each electrode path such as: bone thickness, distance between the inner skull to the cortical surface and distance from the cortical surface to the target.

Given the usually large number of electrodes implanted, the surgeons usually opt for checking the coordinates of the first trajectory with a phantom device (similar to the DBS procedure Figure 2.3b).

The stereotactic frame with its guide is now attached to the reference ring fixed to the patient’s head.



(a) Pre-operative trajectory planning with stereotactic CT registered with pre-operative MRI.



(b) Skin and skull piercing step. Drill depth controlled by a stopper mechanism.



(c) Dura mater and capillary vases coagulation.



(d) Attachment of the screw port.



(e) Stylet introduced to open the electrode path.



(f) Electrode depth controlled implantation and fixation.

Figure 2.4: Steps in the procedure for Stereo-Electroencephalography.

The stopper at the guide is adjusted according to the bone thickness in the path of the selected electrode trajectory, to prevent the drill piece from damaging the cortical surface. A sleeve that fits the thin drill is introduced in the stereotactic guide, and both the skin and skull are pierced, Figure 2.4b.

The drill is withdrawn and a monopolar coagulator is passed through the guide to perforate the dura mater and coagulate surrounding capillary vases, Figure 2.4c. Then, a 15 to 35 mm screw is fixed in the

drilled skull hole. It will work as an access port to the electrode, Figure 2.4d.

Then, the stopper is adjusted to match the distance between the entry point and the “deep” target point, calculated as such: to the screw length, 5 mm are subtracted, corresponding to the screw thread that enters the skull, added the skull thickness, plus the distance between inner skull to the cortical surface, and from that to the deep-seated target. A stylet is inserted through the screw port to clear a path within the brain tissue for the electrode, Figure 2.4e. The stylet is then replaced by a semi-flexible electrode, which itself has a washer acting as a stopper mechanism to prevent going beyond the target point. The washer in each electrode is length adjusted previously to implantation. The electrode has a screwed cap that fits the screw port, sealing and locking the electrode in place, Figure 2.4f.

The process is thoroughly repeated for each indexed electrode. The choice of completing the electrode insertion process for each iteration, instead of drilling the holes sequentially, is part of a thoughtful strategy to limit the leakage of cerebrospinal fluid as well as the entrance of air in the skull cavity. By doing so, it minimizes the brain shift phenom, which is further constrained by the “anchor effect” of the introduced electrodes.

After implantation, the electrodes are connected to an electroencephalogram which records the brain signals for a period lasting between a couple of days to several weeks. The signals from electrode contacts are monitored to establish the location of the epileptogenic focus, based on the irritative zone (interictal spikes and high-frequency oscillations) and the seizure onset zone (electric activity at ictal onset and ictal baseline shift). Before the electrodes removal, the neurosurgeon applies a stimulating signal, to the vicinity of the delimited region, to map cortical and subcortical eloquent brain areas. Bipolar coagulation may be applied if the electrode contacts register marked epileptic activity, and prior stimulation supports the remoteness of eloquent brain areas.

2.2.3 Other Procedures

The applicability of the stereotactic technique extends well beyond the domain of functional electrode implantation. In order to keep this introductory chapter concise, instead of mentioning each other surgical procedure, we chose to partition them into categories: biopsy, destructive and resection surgery. The goal of this subsection is to generally describe the concepts and goals of these operations, with an emphasis on the surgical process. To keep a coherent line of thought, we narrowed the scope to minimally invasive brain procedures.

Stereotactic Biopsy is the least invasive surgical method for tissue sampling known. It can be performed under local or shallow anesthesia, suited for patients with poor medical status and not amenable for general anesthesia such as infants or people of advanced age. While neuroimaging studies with CT or MRI indicate the presence or not of lesions, precise histological diagnosis is crucial for selecting the best treatment of intracranial masses and infiltrating lesions. The minimally invasive nature of the procedure yields its low morbidity and mortality rates. Stereotactic brain biopsy is considered a relatively safe surgery prescribed to patients with: multiple, poorly defined, hard to access or radiosensitive lesions (Elder et al.,

2009).

The patient initially undergoes an imaging examination preferably a combination of CT and MRI for high tissue contrast with low spatial distortion. The lesions are located in relation to the established stereotactic frame of reference, and the instrumentation guidance trajectory selected to avoid, whenever possible, sulcus, cortical arterial or venous structures. Likewise to other stereotactic procedures, the patient's head is shaved and sterilized, the stereotactic frame attached and sterile drapes applied. Usually, local anesthesia is administered, followed by the scalp incision and skull drilling. The dura is punctured with a blunt tip probe, and then, the biopsy needle advances to the stereotactic target. Intra-operative imaging can be performed to confirm target location (Field et al., 2001; McGirt et al., 2005).

Stereotactic Resection and Evacuation Surgery encompasses procedures that involve access to a volume defined region of the brain for lesion removal or drainage, i.e. resections (Harris et al., 2005; Kelly, 1992) and abscess/hematoma evacuation (Brouwer et al., 2014). These procedures are prescribed for an ample variety of neurological pathologies: tumors, cysts, hematomas, stem lesions, etc. Volumetric stereotaxy differs from the point-in-space surgeries mostly during the planning phase. The surgery plan of the later can be summed by entry, target and a few more point coordinates, volumetric stereotaxy depends heavily on information such as the volume and shape of the lesion (Kelly, 1998).

Previous to the surgery a volumetric imageology study is conducted and information such as evacuation volume or lesion boundaries are extrapolated. The surgical approach is decided upon the histology and anatomical location of the lesion. As for any other stereotactic procedure, a head fixating device is attached to the patient's head that remains immobile or calibrated to the instrumentation guiding apparatus. The scalp is prepared, incised and the craniotomy performed. The size of the brain access hole depends on the procedure and on the instrumentation used.

- In resection or excision surgeries, a wider access is required to remove the sectioned tissue or to pass instrumentation. Naturally, in these surgeries, the size of the burr hole is proportional to the dimensions of the tissue to remove. In case of, stereotactic endoscopy removal (e.g. intra-ventricular colloidal cyst), the burr hole is dimensioned according to the conduit introduced, which creates the channel to pass the endoscope and micro-instrumentation required for section and cauterization.
- In abscess/hematoma evacuation, a small incision suffices to pass the fine tube that perforates the lesion walls and aspirates its contents. In some cases, an ultrasonic disintegration device is employed to facilitate the liquidification and suction of the contents.

Depending on the procedure and resources of the surgical center, a tracking device is used to verify instrumentation placement in respect to the surgical field, and/or an imaging device is used to confirm the effectiveness of the treatment (volume resected/evacuated).

Volumetric stereotaxis has several advantages when compared to the open brain approach: i) facilitates lesion location by the surgeon, ii) imparts a three-dimensional volume and shape of the lesion to address, iii) it enables a minimally invasive procedure with respect to the configuration of the lesion, normal brain,

and vascular anatomy, and finally iv) it indicates where the lesion ends and normal tissue begins by means of a real-time display, interactive software, and stereotactic instrument tracking.

Stereotactic Destructive Surgery aims to achieve selective lesioning, inactivation or modulation of abnormal brain tissue. we will specifically target radiofrequency ablation, laser-induced thermal therapy (LITT) and brachytherapy, some of the most common and successful types of minimally invasive stereotactic ablative surgeries.

Ablative or lesioning procedures were historically one of the first functional neurosurgeries to treat drug recalcitrant movement and non-psychotic psychiatric disorders (Greenberg et al., 2010; Curry et al., 2012). Many physical principles have been employed to disrupt pathological neuronal activity such as radio-frequency (RF) heating, chemical destruction, ionizing radiation, mechanical methods (leucotomy), focused electromagnetic waves and lasers. RF ablation is perhaps the most popular method with several advantages ranging from well-circumscription of lesions to seamless temperature, target control, and treatment versatility. The minimally invasive procedure rests on the generation of a high-frequency oscillating electromagnetic field, created by the implanted lesion electrode at the intracranial treatment site paired with a large area dispersive electrode placed at the skin surface. The high-frequency oscillation of ionized particles provokes a temperature increase radial to the lesion electrode, which permanently damages the surrounding tissue when it heats beyond 45° (Cosman et al., 1984). Functional ablative procedures are arguably being replaced by brain stimulation procedures (refer to subsection 2.2.1), which advertise the same therapeutic output with additional advantages: i) reversibility, ii) versatility, iii) safety and iv) novelty (Hariz and Hariz, 2013).

Laser Induced Thermal Therapy or LITT, relies on the application of a laser fiber directly to the deep-seated target, and by photo-coagulation the tissue surrounding the fiber tip is heated to temperatures between 55 and 95°C. The surgical placement of the optic fiber follows a similar procedure to the described SEEG electrode placement (refer to subsection 2.2.2), the greatest difference being the use of MRI compatible instrumentation. The patient is placed in the MRI and the fiber placement verified both visually and with a low-power test pulse. An accurate and real-time map of tissue temperature is acquired by magnetic resonance thermal imaging. More than one laser position and activation may be used to cover the full target volume. LITT is of particular use in neuro-oncology to treat systemic cancer (Carpentier et al., 2012; Buttrick and Komotar, 2016).

Radiotherapy is predicated on a differential radiosensitivity between neoplastic and normal brain tissue. Two different approaches exist: teletherapy (Andrews et al., 2004) and brachytherapy (Nachbichler and Kreth, 2018; Magill et al., 2017). The first is a closed-skull, non-invasive procedure based on multiple collimated beams of radiation, stereotactically directed towards the treatment site. It is performed with high-energy radiation technology such as linear accelerators (LINAC), the gamma knife, or other charged particles produced by cyclotrons/synchrocyclotrons. Alternatively, interstitial brachytherapy is an invasive procedure that involves the direct permanent or temporary placement of single to multiple radioactive sources within the prescribed tumor volume. Compared to teletherapy, the invasive brachytherapy achieves higher therapeutic ratios through selective administration of radiation with a steep falloff near normal brain

tissue. It causes a well-defined small region of neoplastic tissue necrosis that develops radially from the radioactive seed, which is subsequently removed by macrophage activity. Surgery planning involves the determination of the volume and shape of the target, the choice of isotope, number of seeds, activity, and location within the tumor.

The stereotactic surgical workflow for both ablative or brachytherapy treatments involve the accurate placement of electrodes/catheters, which is a procedure similar to those described in subsections 2.2.1 and 2.2.2. Aside from the therapeutic principle underlying each treatment, the most noticeable difference to the described procedures lies on the pre-operative planning step. A careful study of the brain target volume and shape is required for a precise configuration of the equipment used, and subsequent control over the radiation isodoses or the isothermal ablative surfaces.

Chapter 3

Robotic Neurosurgery

For clearness and topic containment, we avoided referring to robotics in the medical introductory chapter 2, even though the neurosurgery field and the robotics realm extensively overlap. It was nothing more than a matter of time until someone realized the applicability of robotics within an operating room.

Robots are by definition mechanically automated devices, programmable to perform ordinary and repetitive tasks with utter precision and consistency. In stereotactic surgery, more so than in open surgery, the whole procedure can be broken down to a sequence of well-defined steps conducted in reference to absolute coordinates, in order to accurately guide instrumentation to the surgical field. Such a formulaic approach closely resembles the typical industrial robot task. In fact, over the past years, it has become increasingly evident that robotic systems significantly enhance the surgeon's surgical capabilities (Nathoo et al., 2003; Eljamel, 2007; Nathoo et al., 2005), as evidenced by fewer intraoperative complications and an improved patient outcome.

The application of robotic technology in the surgical field is relatively recent (late 90s), but it has already been adopted in a vast number of medical domains: neurosurgical, orthopedic, laparoscopic, thoracic etc. In this chapter we will be focusing on the robotic technology that targets stereotactic neurosurgery, listing the most successful and distributed systems, and discussing their features and advantages (Faria et al., 2015a).

3.1 Robotic assistant: neurosurgeons' expectations and demands

How do robotic systems improve the work conditions for neurosurgeons, neurologists and other staff? What tasks can be delegated to the robot? What are the benefits for the patient? What can be expected of a neurosurgical robot? These are some of the most common and fundamental questions often posed to and by developers regarding robotic neurosurgery, which will be addressed below.

As stated previously, typical stereotactic surgery lasts several hours, throughout which the surgical team must remain completely focused. After attending stereotactic surgeries and brainstorming with neurosurgeons and robotic engineers, we were able to answer the first two questions (which are somewhat

related) and concluded that a simple and intuitive robotic system may improve the standard procedure in various aspects and thus:

- Enable the coordinates and electrode's path information to be managed between the planning software and the robotic controller software, instead of manually handling this information.
- Avoid stereotactic frame and mechanical driver slacks or loose parts.
- Avoid the slow process of repeatedly mounting and setting the frame and driver coordinates for both the phantom and patient.
- Allow neurosurgeons to select and insert electrodes in eccentric trajectories, overcoming the constraints imposed by the stereotactic frame apparatus. This is extremely helpful when more than a single trajectory is needed, such as during SEEG, where up to 20 electrodes need to be inserted in a single procedure.
- Enable the robotic manipulator to handle multiple end-effectors and surgical instrumentation to execute restrained skull drilling and the swift positioning of electrodes with improved precision. The manipulator can limit these tasks so that they are executed specifically along the predefined path, instead of executing them on the basis of a marked entry point.
- Enable medical teams to easily take control over the task of increasing the depth of electrodes - while evaluating the patient's symptoms - by simply interacting with a robot graphic interface, which aids neurosurgeons with that task.
- Ensure flexibility and ease in changing the entry point once the burr hole is performed and in the event of encountering an unexpected vascular structure after opening the dura matter.
- Reduce the risk of data loss or human errors.
- Enable online monitoring of the absolute coordinates of the instrumentation tips, based on their physical dimensions and on the manipulator position in relation to the base referential.
- Finally and most importantly, making frameless surgery under robotic guidance possible.

It is important to note that, even though frameless surgery implies no frame, the transformation between the instrument guiding device and the patient must be constant. The most common approach to this problem relies on the use of a Mayfield 3-point pin fixation device (Integra LS, Plainsboro, New Jersey), in order to immobilize the patient's skull. A rigid link connecting the Mayfield and the instrument guiding device then ensures the constant transformation. There are computational solutions in IGS for the active robot compensation of patient motion, which presents acceptable accuracy but this is still rather limited in the compensation time delay (Haidegger et al., 2009).

Robotic systems enhance accuracy, precision and steadiness (Cardinale and Mai, 2011), which are directly reflected in fewer intraoperative complications and produce a positive impact on the patient's outcome (Nathoo et al., 2003; Camarillo et al., 2004). It is not only the patient but also the healthcare institution that benefits from shorter patient recovery times and lower occupancy rates.

When consulted about the expectations related to the robotic system for stereotactic neurosurgery, neurosurgeons look forward to: i) a simple system of intuitive usage, ii) a cost-effective solution, iii) a small, easily mountable and movable device. Thus, apart from the main goal of positioning and manipulating

surgical equipment, the most sought-after assets reside in the human factors and in the integrability of the robotic system. These aspects should thus be targeted by engineers when devising a robotic platform for stereotaxy.

3.2 State-of-the-art robotic systems

Since the first report of a robotic neurosurgical system in 1985, a wide range of neurosurgical solutions has been developed (Kwoh et al., 1988). In order to keep this section concise, we chose to include the most successful robotic systems or projects directed at stereotactic neurosurgery that either reached the market or were clinically tested, with reported in vivo results¹ (see Table 3.1). Robotic platforms for endovascular or radiosurgery were not included in this review. The listed robotic systems were divided into three categories according to user-interaction (see Nathoo et al. (Nathoo et al., 2005)):

- Supervisory Controlled, the robot motion performed during the operation is explicitly or implicitly specified by the surgeon offline. During the procedure, the robot moves autonomously under the surgeon supervision.
- Telesurgical, the robotic manipulator (slave) is directly controlled by the surgeon through an input device like a joystick (master), which is usually endowed with force feedback.
- Shared Control, surgeon and robot share the control over the surgical instrumentation. The surgeon still controls the procedure while the robot provides steady-hand manipulation or active-restrain over surgical safety areas.

The list is organized in two parts to differentiate robotic systems: those developed specifically for stereotactic neurosurgery and general robotic systems, which are capable of performing/assisting stereotactic neurosurgery although they were not solely designed for it.

¹The Robocast and NeuRobot projects did not report clinical trials, but were involved in major European-funded programs, and were included for their contribution.

Table 3.1: Most successful robotic systems and projects directed at stereotactic neurosurgery.

	Project	Phase	Category	Institution	Main features
SPECIFIC	SurgiScope	Commercial use	Supervisory Controlled	ISIS Robotics, Saint Martin d'Hères, France	Delta parallel ceiling-mounted robotic manipulator with 7 DoF, modular architecture (user chooses the modules to work with)
	NeuroMate	Commercial use	Supervisory Controlled	Renishaw-Mayfield, Nyon, Switzerland	Serial robotic manipulator with 5 DoF, low-speed profile with sensor redundancy, mobile base, integrated planning system, frame/frameless ultrasound and CT-based registration
	Pathfinder	Discontinued	Supervisory Controlled	Prosurgics Ltd., High Wycombe, United Kingdom	Serial robotic manipulator with a 6 DoF robot, mobile base, integrated planning system, frameless registration using fiducial markers
	Renaissance	Commercial use	Supervisory Controlled	Mazor Robotics Ltd., Caesarea, Israel	Hexapod parallel robotic manipulator with 6 DoF, small and portable, directly mounted on the skull, integrated planning system, frameless and markerless, low-cost
	Robocast	Project terminated	Supervisory Controlled, Telesurgical	NearLab, Politecnico di Milano, Milan, Italy	Serial, parallel and linear multi-robotic tele-operated system with 6 + 6 + 1 DoF, mobile base and integrated planning system
	RONNA G3	Experimental setup	Supervisory Controlled	University of Zagreb and University Hospital Dubrava, Croatia	Single or dual serial robotic manipulator 6 DoF, mobile base, integrated planning and tracking system, frameless
	Rosa	Commercial use	Supervisory Controlled, Shared Control	MedTech SAS, Montpellier, France	Serial robotic manipulator with 6 DoF, low-speed profile, mobile base, integrated planning system, frameless registration, shared control manoeuvrability
GENERAL	MKM	Discontinued	Supervisory Controlled, Telesurgical	Carl Zeiss, Oberkochen, Germany	Operating microscope mounted on a 6 DoF serial robotic manipulator for microscope navigation and tool guidance in biopsy applications
	NeuroRobot	Project ended	Supervisory Controlled	Imperial College of Science, Technology and Medicine, London, United Kingdom	4 DoF rigid platform to hold and manipulate an endoscope around a pivot point, dynamical workspace constraint, frame reliant
	Evolution 1	Discontinued	Telesurgical	Universal Robot Systems, Schwerin, Germany	4 DoF hexapod robot with tele-operated parallel actuator, mobile base, integrated planning system, for brain and spine applications
	neuroArm / SYMBIS	Experimental setup	Telesurgical	IMRIS, Winnipeg, Canada	Two 7 DoF tele-operated manipulators with an extra DoF due to the tool actuation mechanism, integrated with MRI technology for intraoperative instrumentation tracking

3.2.1 Specifically for Stereotactic Neurosurgery

3.2.1.1 SurgiScope

SurgiScope (ISIS Robotics, Saint Martin d'Hères, France) development started in 1989 and ensued from a cooperation between the University of Grenoble and the industrial company AID. It is currently available at an operating level (Benabid et al., 2006), and has been produced and installed worldwide with 40 fully operating units, implemented by more than 10 surgical teams (mainly in France).

The ceiling-mounted 7 DoF robotized manipulator is based on a parallel delta mechanism (Figure 3.1) and is mainly dedicated to endoscopy and biopsy procedures or neuronavigation applications (Briot et al., 2007). SurgiScope is particularly useful in intracranial operations when the procedure requires navigation between sensitive neural elements, visible through a restricted access (Benabid et al., 2006). Additionally, its neuronavigation function facilitates resections or targeting procedures, when the boundaries of the surgical target volume are not visually distinct (Amin and Lunsford, 2004).



Figure 3.1: ISIS Robotics SurgiScope (Courtesy of ISIS Robotics).

SurgiScope is the basis for multiple integrable upgrade modules including: 1) image import/conversion and treatment/planning software, 2) the microscope kit, 3) a handle set to single-handedly control system motion, 4) a tool holder kit to position and hold surgical instrumentation, and 5) a head-up display to exhibit

customized surgical plan data in the microscope oculars.

Preoperative targeting and trajectory planning are performed in the SurgiScope workstation (Lollis and Roberts, 2009). The patient's head is fixated to the operating bed through a Mayfield, and the registration between preoperative planning and intraoperative space is achieved with scalp fiducial markers using a handheld probe (Bekelis et al., 2012). After the craniotomy, the SurgiScope robot can operate in two modes. In microscope mode, the robot acts as a platform to operate a microscope. It aligns its optical axis with the predefined trajectory and adjusts the microscope focal point to the surgical target. In biopsy mode, an arm attachment with a probe carrier is attached to the robot. The SurgiScope robot then aligns its arm to the prescribed trajectory (Lollis and Roberts, 2009). Through the bushings of the robotically positioned stereotactic guide, the insertion needle is advanced to the planned target (Spire et al., 2008). Lollis and Roberts (Lollis and Roberts, 2009) have reported the application accuracy of SurgiScope: the mean displacement from the catheter tip to the target is 1.6 ± 3.0 mm, in the robotic placement of a central nervous system ventricular reservoir.

One of the greatest advantages of SurgiScope is the possibility to acquire and work with individual system modules, which allows surgical teams to avoid superfluous features and thus reduce system cost. SurgiMedia, a modular platform to cope with the SurgiScope multimedia part, guarantees system compatibility with any type of surgical material available, which further enhances the system's flexibility. Extended operative time, acquisition costs and lack of mobility are considered to be the main drawbacks (Lollis and Roberts, 2009).

3.2.1.2 NeuroMate

NeuroMate (Renishaw-mayfield; Nyon, Switzerland) was the first neurosurgical robotic device to obtain the EC brand in Europe and FDA approval in 1997 for stereotactic neurosurgical procedures (and in 1999 for frameless), thus constituting a major milestone and setting the standard (Haidegger et al., 2008) (Figure 3.2). The NeuroMate works as an image-guided, passive assistant for holding, supporting and stabilizing instrumentation controlled by the surgeon, thereby increasing surgical safety and improving surgery efficiency (Li et al., 2002; Varma and Eldridge, 2006). This robotic system shows appropriate mechanical stiffness, good accuracy and convenient workspace for stereotactic keyhole neurosurgery applications. Its advantages are even more evident in surgeries or biopsies that target multiple structures (Li et al., 2002; Xia et al., 2008). For a thorough explanation of a surgical workflow involving NeuroMate, refer to (Cardinale et al., 2012).

It includes kinematic positioning software, as well as a 5 DoF arm that achieves a technical accuracy of 0.7 mm and a precision of 0.15 mm, guaranteeing payload stability up to 5 kg (Benabid et al., 1987a; Varma et al., 2003). The neurosurgeon may choose to purchase the basic NeuroMate platform and acquire additional modules for frame-based, frameless and other functionalities on demand. Alternatively, the NeuroMate system may be integrated into a custom workflow, coping with existing solutions (Xia et al., 2008; Cardinale et al., 2013; De Momi et al., 2013). Its design enables the use of conventional stereotactic localizer frames or an exclusive frameless method that resorts to an ultrasound system to register the



Figure 3.2: Renishaw-Mayfield NeuroMate.

robot's position in relation to the patient's skull (Varma et al., 2003). Developed for neurosurgery, the NeuroMate presents unique features that distinguish it from industrial robots, such as low speed, sensor redundancy, and safety devices (Li et al., 2002; Varma and Eldridge, 2006).

Li et al. (Li et al., 2002) have reported the NeuroMate's in vitro application accuracy using the frame-based (0.86 ± 0.32 mm) and frameless (1.95 ± 0.44 mm) approach. It was concluded that there is no statistically significant difference in accuracy between the frame-based traditional approach and NeuroMate's frame-based application. Other studies (Golash et al., 2000; Varma et al., 2003) have validated and demonstrated the reliability of the frameless method against frame-based surgery. Cardinale et al. (Cardinale et al., 2013) have reported the in vivo localization error of the NeuroMate frame-based approach in 91 SEEG procedures as being 0.86 ± 0.54 mm at the entry point and 2.04 ± 1.31 mm at the target point. Recently, von Langsdorff et al. (von Langsdorff et al., 2014) studied the application accuracy (better than 1 mm) of the NeuroMate frame-based approach in vivo for DBS electrode implantations.

The negative aspects pointed out reside in the bulk robot structure and the cost of the system's acquisition. According to neurosurgeons, one desired upgrade would be to endow NeuroMate with drilling capabilities (Cardinale et al., 2013).

3.2.1.3 Pathfinder

The Pathfinder system (Prosurgics Ltd., High Wycombe, United Kingdom) (Figure 3.3) is a robot built for neurosurgical procedures. It constitutes a response to instrumentation miniaturization and to the demand for greater accuracy which, as stated by Eljamel (Eljamel, 2007), will soon transcend even the most skilled surgeon capabilities. A 6 DoF robotic arm is installed on a mobile and stable platform, which

can be easily moved around the operating room and firmly fixed to the Mayfield during surgery. One of Pathfinder trademarks lies in the fiducial markers (reflectors) attached to the patient's scalp or skull, and their continuous tracking using an embedded vision system to register the robot to the intraoperative space (Deacon et al., 2010). These markers consist of a black titanium sphere, coated in a reflective material, which is easily visible in CT scans and by the camera system (Eljamel, 2007; Morgan et al., 2003).



Figure 3.3: Prosurgics Pathfinder.

An initial CT exam is used to pinpoint the marker positions in relation to the surgical volume, while the MRI dataset is required to segment the target brain structures. The CT and MRI datasets are then matched to overlay the targets and fiducial markers' locations. The Pathfinder planning software allows the neurosurgeon to view, edit and mark medical images of the patient, and to plan the probe's trajectory (Finlay and Morgan, 2003). The Pathfinder can fixate itself to the Mayfield, opposite the surgical side or at an acute angle parallel to the patient. By doing so, the robot can operate with some flexibility without interfering with or obstructing the neurosurgeon's workspace (Eljamel, 2007).

Pathfinder frameless registration allows for target acquisition with millimetric accuracy (Finlay and Morgan, 2003). Furthermore, the robot can be repositioned in the operating room without the need to rescan or replan (Sivakumar et al., 2003). External fiducial markers allow the robotic system to constantly track its position in relation to the patient, thus solving one of the greatest issues regarding preoperative image-guided robots, and relieving the need for intraoperative online image scans (Eljamel, 2007; Deacon et al., 2010). The most commonly reported problems with the Pathfinder system are: possible skin movements between

preoperative and intraoperative scans, and registration failures caused by the misidentification of markers due to abnormal lighting conditions (Eljamel, 2007).

Upon contacting the Pathfinder manufacturers, we were informed that this project terminated at the beginning of 2009 due to lack of substantial funding and certification issues, and that ProSurgics was later acquired by FreeHand 2010 Ltd.

3.2.1.4 Renaissance

The Renaissance robotic system (Mazor Robotics Ltd., Caesarea, Israel), originally developed for spine pedicle screw insertion by Prof. Shoham, was adapted for minimally invasive neurosurgeries (Devito et al., 2010; Ringel et al., 2012; Hu et al., 2013). The system is composed of the MARS robot and controller, a custom robot base, a targeting guide, and a registration jig. It is also accompanied by an "off-the-shelf" 3D laser scanner and a standard PC (Shoham et al., 2007). The system comprises 4 software modules: i) preoperative planning; ii) surface scan processing; iii) 3-way registration and iv) intraoperative execution. The system fits into the category of Supervisory Controlled, and mainly serves the purpose of tool guiding and drill assistance.

MARS is a small portable 6 DoF parallel robot ($5 \times 8 \times 8$ cm and a weight of 250 g) with a motion accuracy of 0.1 mm and a resolution of 0.02 mm (Figure 3.4). The robot can be directly mounted on the patient's skull through the custom robot base or mounted on a Mayfield. It is endowed with a lock mechanism, which is activated upon aligning the guide with the predefined entry point/target axis. The robot remains locked and rigid throughout the guiding and drilling phase and is able to withstand lateral forces of up to 10 kg and actuation forces up to 1 kg.

The surgical procedure using the Renaissance system follows the premises of IGS. Initially, a markerless and frameless CT/MRI scan of the patient is undertaken, where the surgeon defines entry and target points, and the type of robot mounting selected (custom base or Mayfield) (Joskowicz et al., 2006). The registration between preoperative planning and the intraoperative space is achieved through the surface matching of the CT/MRI and laser scan cloud of points (Shamir et al., 2005; Joskowicz et al., 2005). The transformation between the MARS robot base and the intraoperative space is computed based on a surface cloud of points containing both the registration jig (high relief wide-angle tetrahedron shape) and the patient's forehead or ear. The MARS robot now replaces the registration jig, and automatically positions its guide along the planned insertion trajectory. On surgeon demand, it automatically changes its guide position to a new trajectory (Joskowicz et al., 2006).

The Renaissance system surface registration error was reported to be close to 1 mm, while the target registration error was approximately that of 1.7 mm (Shamir et al., 2005; Joskowicz et al., 2005, 2006). Recently, a target registration error of 0.65 mm was reported by Joskowicz et al. (Joskowicz et al., 2011) in a phantom study.

As a frameless and markerless system, Renaissance overcomes the morbidity and head immobilization requirements associated with stereotactic frames; it further eliminates the line-of-sight and tracking requirements of navigation systems and still provides a rigid platform for mechanical guidance without the



Figure 3.4: Renaissance MARS robot (Courtesy of Mazor Robotics, Inc.).

bulk and costs associated to large robots. The system's cost was initially aimed to be under 100 000 USD, unlike other robotic solutions which range from 300 000 to 500 000 USD (Joskowicz et al., 2005, 2006). A recent article in an investment research platform revealed the listed system price to be 849 000 USD and the disposables 1 500 USD (Capital, 2014).

3.2.1.5 RONNA G3

The RONNA G3 system is the third generation of a robotic neuronavigation system that is the product of a cooperation between the Faculty of Mechanical Engineering and Naval Architecture from the University of Zagreb and the University Hospital Dubrava. Tailored for stereotactic frameless minimally invasive surgery, more specifically brain biopsy, the system prototype has been upgraded and was recently tested in clinical trials. The system features three main components: i) a robotic arm mounted on a universal mobile platform, ii) a surgery planning system and iii) a global tracking system (Dlaka et al., 2018).

The RONNA G3 system is designed to work with a single robotic arm - master, which acts as a passive navigation guide leaving the neurosurgeon in charge of drilling and handling the surgical instrumentation. An extended version of the system includes a dual-arm system, being the second arm - assistant - responsible for inserting the operating instruments in the tool guide. The patient registration is based on retroreflective fiducial markers, either freely distributed on the head and secured by bone screws or assembled to a single cross-pattern support. The fiducials are located in the stereotactic image scans using the system's

planning software (RONNAplan), through a manual or automated process. Perioperative and after fixating the patient's head with a Mayfield clamp, the markers are located and tracked by a wide range infrared stereo camera (Polaris Spectra, Northern Digital Inc., Canada). It tracks the coarse position of the markers in the patient and of the robotic arm(s). While the global tracker information is used to assure safe and collision-free motion planning, a high-precision localization device (RONNAstereo) identifies the exact position of the markers which is used for the image-patient registration (Švaco et al., 2017; Dlaka et al., 2018).

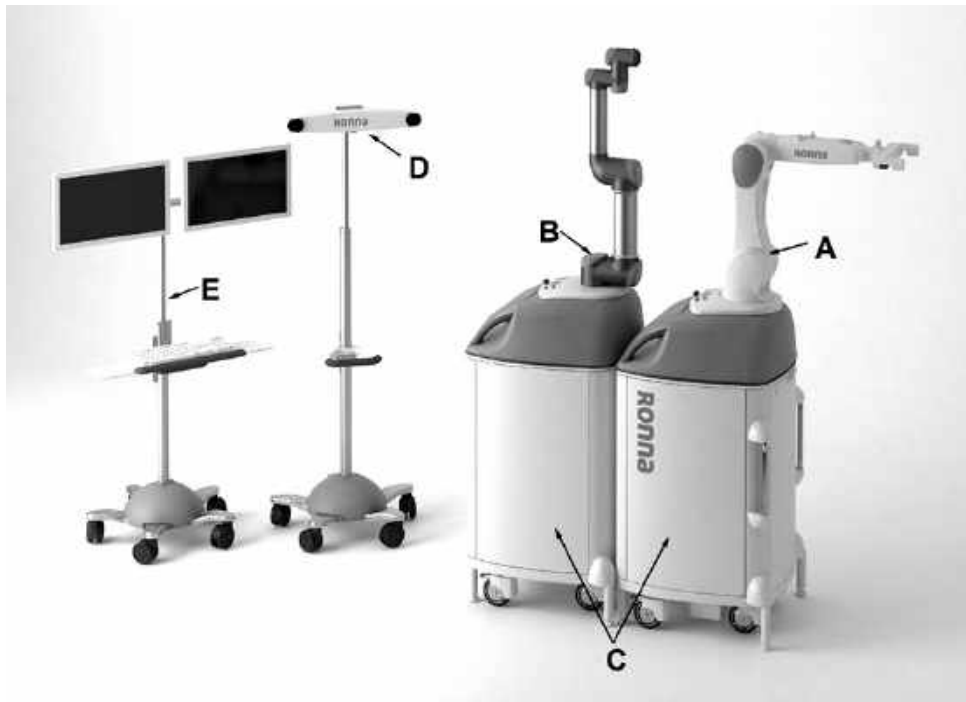


Figure 3.5: RONNA G3 neuronavigation robotic system: A) master robot; B) assistant robot; C) universal mobile platforms; D) optical tracking system; E) control and planning software interface (reprint copyright license agreement with John Wiley and Sons).

The robotic arm mobile platform is fixed to the operating table to ensure a stable link throughout the procedure. Once the registration is concluded, the RONNAstereo end-effector is replaced by a tool guide and the localization procedure is verified by pointing at anatomical landmarks. Thence, the robot moves to a pre-planned trajectory for the skull drilling and needle insertion steps. An in vitro study reported the system's mean positioning accuracy to be 0.65 mm with a maximum error of 1.89 mm (standard deviation of 0.39 mm) (Dlaka et al., 2018). The error was strongly affected by the proximity between the markers and the target points, with lower errors in shallow targets. In the reported clinical trial, the authors measured the entry and target point errors of 2.24 mm and 2.33 mm respectively, as the Euclidean distance between the planned and effective points (Švaco et al., 2017). The RONNA G3 system poses itself as a precise frameless neuronavigation robotic system, however, more clinical trials are required to assess the system's performance. The project is funded by the Croatian Scientific Foundation (Research project ACRON, grant no. 4192).

3.2.1.6 Robocast

The Robocast – an acronym of Robot and Sensor integration for Computer Assisted Surgery and Therapy project (FP7 ICT-2007-215190) – aimed to create a modular system to integrate image-guided navigation and robotic devices for keyhole surgery (Figure 3.6). The project developers envisaged a human-robot interface with context-intuitive communication, embedded haptic feedback, a multiple robot chain with kinematic redundancy, an autonomous trajectory planner and a high-level controller (Comparetti et al., 2011a; De Momi et al., 2009).



Figure 3.6: Robocast robot (Courtesy of De Momi, E. and Ferrigno, G. - Robocast Project).

The Robocast system consists of an optical and electromagnetic tracking system, ultrasound and three robotic actuators with haptic devices. The first robot, or gross positioner, is the Pathfinder robot with 6 DoF. There is another called fine positioner, which is the MARS (Renaissance) parallel robot with 6 DoF to further improve accuracy. The third is a linear piezo actuator to ensure the linear insertion of electrodes or biopsy probes. The optical tracking system is used to register the intraoperative environment according to the preoperative plan. A single DoF haptic feedback actuator is used to control probe depth (De Lorenzo et al., 2011).

The software platform can be divided into six subsystems: preoperative planning, human-computer interface, sensor manager, high-level controller, haptic controller and safety check (De Momi and Ferrigno, 2010). After the neurosurgeon has selected the target and entry area, the preoperative planning software autonomously calculates the lower risk optimal entry point and trajectory (De Momi et al., 2009, 2013). Human-Computer Interface allows the surgeon to interact with the navigation system, while the sensor manager assembles data from the ultrasound and tracking system and inputs this to the system control

center. The high level controller manages information from the preoperative planning and sensor manager subsystems, and iteratively calculates the gross positioner and fine positioner kinematics (Comparetti et al., 2012b). The haptic controller interfaces the linear actuator robot with the haptic device, transmitting a force-feedback reaction to the surgeon so that the probe will be moved. Finally, the safety check module runs regular state verifications for each subsystem; in the case of failure, it stops the probe movement (Comparetti et al., 2011b).

The technical accuracy of the iterative targeting approach based on continuous optical feedback was evaluated in vitro, in optimal and noise-induced conditions. The largest reported translation median error was 0.6 mm and 0.4 mm for the entry and target points, respectively. While the largest rotation median error was 6.5×10^{-3} rad (Comparetti et al., 2012b). The accuracy reported fits the requirements for clinical applications.

The Robocast project ended in 2011 and has been continued by the Active project - an acronym for Active Constraints Technologies for Ill-defined or Volatile Environments (FP7-ICT-2009-6-270460) (Active Project, 2012; De Momi et al., 2014), which proposes an integrated redundant robotic platform. This relies on two autonomous cooperating robotic manipulators for neurosurgery, which form a light and agile system with 20 DoF.

3.2.1.7 Rosa

The Rosa robotic system (MedTech SAS, Montpellier, France) is the latest generation of neurosurgical computer-controlled robots for stereotactic surgery (Figure 3.7). The Rosa system comprises a mechatronic part consisting of a 6 DoF serial robotic manipulator and a control software part for neurosurgery planning, registration, and guidance (Medtech S.A, 2012).



Figure 3.7: MedTech Rosa (Courtesy of MedTech Surgical).

The planning software (Rosana, MedTech) allows for the merging of different and complementary imaging techniques when studying the best surgical approach. The patient initially undergoes an MRI exam (with or without contrast, various supported sequences) to visualize the target anatomical structures, and to plan the optimal guiding trajectory (Gonzalez-Martinez et al., 2014; Serletis et al., 2014). This plan is then registered to a CT scan, performed near the time of surgery, and which serves as the reference due to its homogeneous geometric accuracy. An intraoperative Flat-Panel CT can be integrated into the surgery workflow to compensate for brain shift or robot registration errors (Lefranc and Le Gars, 2012; Lefranc et al., 2014b).

After uploading the plan to the Rosa system, the robot is firmly fixed to the skull clamp. The surgery team may choose to register the robot to the intraoperative scene in a frame-based (Leksell frame) or frameless approach. The frameless method is carried out using fiducial markers attached to the scalp/skull, or via the Rosa patented automatic surface scan. The latter combines robot motion and laser telemetry to provide a non-invasive registration (Lefranc et al., 2014a; Medtech S.A, 2010).

The robot is draped after a satisfactory registration and, upon the surgeon's command, automatically moves to the planned guiding trajectory. It remains in a locked state while the entry point is marked and prepared. Scalp incision and skull drilling are performed with a cordless power drill (Gonzalez-Martinez et al., 2014). The neurosurgeon may choose to insert the probes or electrodes manually through the adapted reducers held by the arm, or may use the haptic robot interface to lower the instruments (Lefranc et al., 2014b). This shared-control feature allows for intuitive interaction and control by the neurosurgeon with tremorless and motion restriction advantages.

Lefranc et al. (Lefranc et al., 2014a) present a study comparing different modalities of image and robot registration with a phantom and in actual procedures. The Rosa system achieves an accuracy below 1 mm for frame-based and fiducial registration, and a 1.22 mm accuracy for frameless surface registration, both with CT as well as reference imaging².

The greatest asset of the Rosa system, when compared to the other solutions, is its flexibility. It is easily integrated into the institution workflow and is reported to be well-accepted (Lefranc et al., 2014b). No other robotic system offers so many options regarding robot registration. The Rosa system provides consistent and accurate instrument guidance while keeping surgery times comparable to conventional methodologies (Lefranc and Le Gars, 2012; Lefranc et al., 2014b; Gonzalez-Martinez et al., 2014). With regard to its negative aspects, users point to the robot's learning curve and bulk dimensions, which limit the neurosurgeon's workspace.

3.2.2 General and capable of Stereotactic Neurosurgery

3.2.2.1 MKM

The MKM system (Carl Zeiss, Oberkochen, Germany) stands for "Multicoordinate Manipulator", and consists of three components: 1) an operating microscope mounted on 2) a 6 DoF motor-driven robotic arm, and 3)

²Surface registration with MRI scans are error-prone due to image-related distortions, leading to significantly lower overall accuracy.

a computer workstation (Pillay, 1997). Its initial goal was to serve as a frameless stereotactic navigation system, by joining the concepts of intraoperative microscopy and neuronavigation in minimally invasive IGS (Roessler et al., 1997).

The surgical procedure is planned and based on preoperative image scans, which are then registered to the intraoperative scene using scalp or bone fiducials. Inside the operating room, the neurosurgeon visualizes the neuroimaging plan, superimposed onto the microscope optical field, and showing the entry point, target point, lesion contours and other structure markings (Pillay, 1997; Roessler et al., 1998). Several advantages arise from this fusion: the potential to outline and minimize the size and shape of skin incision, craniotomy, and corticotomy; the capacity to decide between different surgical approaches and the possibility of performing more aggressive resections with a lower risk of damaging surrounding structures (Roessler et al., 1997).

Willems et al. (Willems et al., 2001) extended the applicability of the MKM system by introducing an instrument holder for frameless stereotactic procedures to be mounted on the microscope. This instrument holder, also developed by Carl Zeiss, consisted of an extension arm which rigidly fixed to the microscope with a large channel for tool guidance. Plastic reducers are fitted to the channel to constrain different instrumentation, for probe guidance or bone drilling (Willems et al., 2001). The MKM software was equipped with a “tool mode” module, which sets the instrument holder so as to align it with the surgery planned trajectories, rather than the optical axis (Willems et al., 2003). Additionally, instead of telemanipulating the microscope with a spherical sensor joystick, the microscope holder automatically moves to the predefined position (manual repositioning possible). During instrument insertion, however, the system movements are disabled for safety reasons (Willems et al., 2001).

In vitro and in vivo studies were performed with the mounted instrument holder to assess the MKM system's accuracy. Willems et al. (Willems et al., 2001) reported a slightly lower application accuracy with the robot when compared to the BRW frame; yet, there was a comparable target localization error. Willems et al. (Willems et al., 2003) reported an average biopsy localization error of 3.3 mm and 4.5 mm, depending on the registration method used (bone screws or scalp adhesive fiducials). While this is acceptable for brain biopsy procedures, further accuracy is required for functional neurosurgery.

The MKM system presents a rapid, flexible and reliable alternative to stereotactic frames in biopsy brain surgeries and stereotactic neurosurgery guidance (Willems et al., 2001). On the other hand, its high costs of acquisition, bulky structure and lack of mobility, constitute some of its negative features (Willems et al., 2003; Lefranc et al., 2014b).

3.2.2.2 NeuRobot

NeuRobot³ stemmed from the European Community funded project ROBOSCOPE to provide a joint solution for common problems in Neurosurgery. The project involved a robotic arm (NeuRobot) and a simulator

³Do not confuse this with another system called the NeuRobot (Hongo et al., 2006, 2003), which is a telecontrolled micromanipulator system with a master-slave control hierarchy to perform minimally invasive procedures using an endoscope and three robotic arms. There is also another system, also called NeuroBot, which is used in skull-based surgeries (Handini, 2004).

image-guided system, ROBO-SIM. Focusing on the robot platform, the NeuRobot is described by Auer et al. (Auer et al., 2002) as "an active manipulator with inbuilt robotic capabilities" that includes: active constraint mechanisms of the manipulator motions based on mapped permitted regions, a precise pattern control and the capacity to automatically track moving features (Figure 3.8).

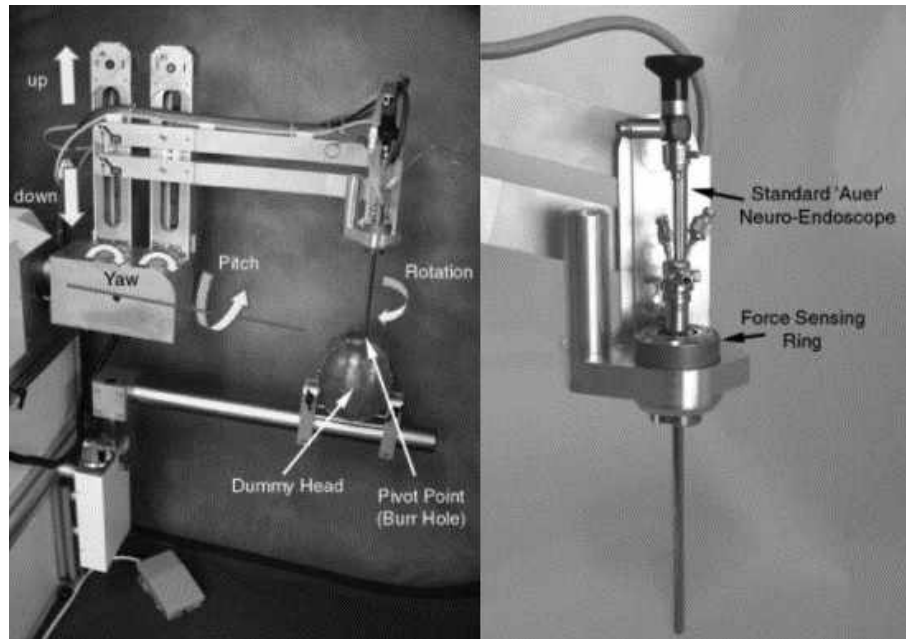


Figure 3.8: NeuRobot (Courtesy of Prof. Brian Davies, at Imperial College of London).

The robotic manipulator has no more than 4 DoF to manipulate instrumentation around a pivot point – the burr hole entry point in stereotaxy. These 4 DoF control the probe orientation around the Yaw, Pitch, Endoscope rotation and the position along an Endoscope depth DoF, which implies that the NeuRobot cannot reach the pivot point on its own and must, therefore, be previously positioned. This is one of the system's disadvantages since, if more than one trajectory is required, the robot needs to be repositioned and readjusted to the surgery table (Davies et al., 2000).

The manipulator includes a control mechanism developed from a flight-simulator experience by Fokker control systems b. v., and enhances precise motion and force-control using low force inputs (Auer et al., 2002). Special attention was paid to safety issues. The system thus includes: a dead man's switch and a workspace which is physically constrained in a safe operating volume based on MRI segmented data. An ultrasound imaging system is used to track tissue deformation during the procedure, and the probe position is dynamically compensated in real-time. The NeuRobot was able to operate autonomously, yet it raised concerns about "who is in-charge" of the surgery (Davies et al., 2000).

Despite its advantages, the system is still dependent on a stereotactic frame to register the robot with the surgery reference (Davies et al., 2000). The robot was initially projected to hold and manipulate a neuroendoscope but, as stated by the authors, it could in principle be used to handle other stereotactic instrumentation. One remarkable advantage of the NeuRobot system is the integrated ROBO-SIM software, which enables the same manipulator to be used in real or simulated interventions to train and help neurosurgeons to become acquainted with the system (Auer et al., 2002).

3.2.2.3 Evolution 1

The Evolution 1 robotic system (Universal Robot Systems, Schwerin, Germany) was specially designed for neurosurgical and endoscopic applications for micro scale brain and spine procedures. Unlike the previous examples, Evolution 1 is a 4 DoF hexapod with a parallel actuator, which combines high accuracy with great payload capacity. Its 6 mechanical parallel axes work as a spherical joint to move a platform with a slider mechanism that holds the endoscope. The parallel actuator approach enhances motion precision achieving an absolute positioning accuracy of 20 μm and a motion resolution of 10 μm , even under loads of up to 500 N (Nimsky et al., 2004; Zimmermann et al., 2004).

Evolution 1 is able to compute the movement of all axes in less than 120 μs . It comprises a universal adapter enabling it to incorporate different types of surgical instrumentation like endoscopes and high-speed drills. Due to the rather reduced working range, however, it must be pre-positioned in the desired orientation, approximately 5 cm above the entry point. Its user-interface is a touch screen and a master joystick device to control end-effector motion and speed (Nimsky et al., 2004; Zimmermann et al., 2004).

Following IGS methodology, end-effector instrumentation follows a trajectory set preoperatively and based on MRI scans and planning software (VectorVision, BrainLab). Intraoperatively, the patient's face is scanned for surface recognition by using infrared technology or laser surface scanning. Later, this information is matched with the preoperative MRI to ensure that the robot knows its position in relation to the surgery reference frame (Zimmermann et al., 2004).

The main advantages of Evolution 1 are high precision and steady positioning/manipulation of endoscope, smooth and slow execution of movement within critical anatomical areas while handling surgical equipment. This system can be potentially adapted to assist stereotactic surgeries. However, a high payload capacity is superfluous since the weight factor is not an important aspect for the instrumentation and tasks at hand. Consequently, a parallel actuator is not always the best choice as it is typically large, thus restraining the neurosurgeon's workspace, and possesses a relatively limited reach/flexibility.

3.2.2.4 neuroArm / SYMBIS

The award-winning system, neuroArm, was developed by Dr. Garnette Sutherland from the University of Calgary in association with engineers from Macdonald Dettwiler and Associates (MDA). It was introduced in 2002, and was recently acquired and renamed SYMBIS (IMRIS, Winnipeg, Canada). The project's main goal is to take advantage of the MR-environment and haptic feedback technology, adding 3D image reconstruction and high-end hand-controller design. It claims the title of being the first image-guided, MR-compatible surgical robot, capable of microsurgery and stereotaxy. It consists of two 7+1 DoF manipulators, which are semi-actively actuated in a master-slave control type and moved by hand control at a remote workstation. The human-robot interface filters undesired hand tremors and can scale the movement of the controls in relation to end-effectors (Pandya et al., 2009; Sutherland et al., 2008).

The neuroArm is built for neurosurgery precision tasks so that each arm has a limited payload of 0.5 kg, a force output of 10 N, a tip speed that ranges from 0.5 to 5 mm/s and sub-millimetric accuracy. Patient safety was a paramount concern throughout the development of the robotic system, and features



Figure 3.9: University of Calgary neuroArm (Courtesy of neuroArm Project, at University of Calgary).

such as active workspace constraints were added in case the robot leaves the safe operating zone. These policies granted neuroArm the approval of the Canadian Standards Association in 2007, as well as that of Institutional Ethics and Investigational Testing by the University of Calgary and Health Canada in 2008 (Figure 3.9).

This robotic system is capable of microsurgery and stereotaxy, which has granted it a place among the general robotic platforms (Sutherland et al., 2003). Despite increasing surgery time, its precision, steadiness, and compatibility with planning software have resulted in reduced trauma and blood loss (Pandya et al., 2009). The end-effector positioning can be verified by overlaying 2D and 3D MRI preoperative and intraoperative information, respectively. After positioning, a Z-Lock feature is used to restrict the tool motion along the defined longitudinal trajectory.

The main advantage of the neuroArm system also constitutes a drawback in some types of stereotactic neurosurgery, due to the need for an MRI scanning machine during the entire surgery with the associated maintenance and acquisition costs. Furthermore, the robotic system costs are also considerably higher since the robotic manipulator is manufactured exclusively with non-ferromagnetic materials (primarily titanium and polyetheretherketone) (Sutherland et al., 2008).

3.3 Current perspectives and Future directions

If one is to compare the most successful robotic systems/projects for stereotactic procedures, one will find several similarities. All the systems follow a very standard and similar surgical protocol related to the IGS paradigm. The main differences are mostly related to technical aspects.

Starting with the robot structure itself, most systems rely on serial instead of parallel actuators. Parallel robots excel at precision associated with larger payload requirements; even so, a larger payload capacity

will seldom be a requirement in stereotactic procedures. Additionally, parallel actuators have very limited access to the surgical target and typically occupy more space next to the patient. Serial manipulators, on the other hand, present greater flexibility and compactness, while still providing a larger workspace, i. e., easier access to surgical targets and trajectories. It is important to note the Renaissance system's unorthodox solution, which takes advantage of the sturdiness of parallel actuators to miniaturize and create a portable robot. Although its narrow workspace prevents its use in SEEG applications, it is used in DBS and biopsy surgeries.

The number of the manipulators' DoF is application-dependent, but this tends to vary between 4 and 7, except for the Robocast project's robot, which follows a multi-robotic 13 DoF approach (for enhanced precision). The number of manipulation DoF affect not only the workspace but the robot's dexterity and flexibility, thus conditioning surgical planning. Fewer DoF and a more reduced workspace mean less flexibility, which directly influences how the robot should be placed in order to reach the planned trajectories, often implying obstructions to the medical team's workspace and vision of the surgical field. Although more DoF and high dexterity is generally an advantage regarding collision avoidance problems, a larger number of joints – particularly in serial manipulators – means more sources of errors that accumulate along the robotic chain.

Most of the robotic systems and projects for stereotactic neurosurgery enable a frameless approach and are gradually becoming detached from the dependency on stereotactic frames. While frameless is one of the flags of robotic systems, the accuracy and repeatability of frameless systems are still surpassed by frame-based systems (Lefranc et al., 2014a; Cardinale et al., 2013). Especially in the case of functional neurosurgery in deep-seated targets, frame-based is still the preferred solution. This is because the frameless approach maximizes accuracy and precision at the entry point rather than at the target point, as in the arc-centered approach (Bjartmarz and Rehncrona, 2007; Zrinzo, 2012). Improving efficiency and developing new frameless registration/fixation methods constitutes a timely endeavor and a research opportunity.

The robotic systems listed converge in other aspects, such as their portability and embedded imaging and planning technology. The lack of mobility in systems like Surgiscope and MKM is seen as a disadvantage. The fact that they are easy to transport and quick/easy to set up is certainly a premise for future robotic system developers. Additionally, the system's modularity and the possibility of choosing from different surgical approaches depending on the clinical case greatly improves the system's acceptance.

Safety is of paramount concern and should be addressed from the early stages of the system's development (Taylor and Stoianovici, 2003). It is the most cited reason underlying a medical team's apprehension in the face of robotic technology (Lavallee et al., 1992). To achieve clinical clearance, a robotic system must at no single point of failure lead to a loss of control and injury to the patient. Critical safety systems like these are typically endowed with redundant position encoders and mechanical limits for speed and exerted forces. Any sensory mismatch or consistency failure should cause the robot to freeze or go limp, while assuring a safe retract mechanism to resume the surgery in a traditional fashion (Talamini et al., 2003; Kortenkamp et al., 2008). Regarding sterilization, the system parts which are in direct contact with the patient must be either disposable or robust enough to withstand autoclaving or other sterilization

methods. Non-sterilized components need to be covered in sterile drapes or pre-sterilised bags (Taylor and Stoianovici, 2003). Lastly, the neurosurgeon can also constitute a source of errors, and must thus be carefully trained with the robotic system, and with the new procedure workflow involving the robot. Surgeons need to be instructed as to the capabilities and limitations of the system, and become acquainted with the execution of the new surgical plan to check for any potential changes/problems (Kortenkamp et al., 2008).

The most referred drawbacks of surgical robots are the high acquisition costs for hospitals and academic institutions (Mattei et al., 2014). One can argue that the passive behavior expected of a robot assisting stereotactic surgery in manipulation and placement tasks is somewhat similar to industrial tasks. An obvious choice would be to import industrial technology to the operating room. However, according to Davies (Davies, 2000), for an industrial manipulator to comply with healthcare safety regulations, it should undergo several modifications which will further increase the robot's costs. In any case, the major obstacles for the development of new surgical robotic systems can be attributed to: long and costly developments with little return; insurmountable barriers of regulatory approvals or legal battles for intellectual property (Gomes, 2011).

For new robotic platforms to achieve significant clinical acceptance, they should present unambiguous advantages over conventional approaches (Kortenkamp et al., 2008; Marcus et al., 2014). The trend of technological progress is currently oriented towards the miniaturization and development of cost-effective robotic systems without jeopardizing performance; and the upgrading of human-machine interfaces possessing enhanced haptic feedback and a seamless integration with several imaging modalities in a surgically relevant and yet intuitive manner (Motkoski and Sutherland, 2014; Mattei et al., 2014).

Chapter 4

Industrial Manipulator Adaptation

Robotic neurosurgery is a vibrant and active research field, with only a handful of robotic systems validated and market-ready. However, all of these systems share high acquisition and maintenance costs and have their control scheme closed (Nathoo et al., 2005; Faria et al., 2015a; Doulgeris et al., 2015). The development of a robotic system for neurosurgery follows one of two approaches, either the robot is originally designed for the task, or it is adapted from an industrial robot. Nonetheless, this development process seldom follows a straight path from concept to development. It is precisely here that the hence presented system fits its role, not only as a proof and testing tool but also as a solution for research problems like (electrode implantation in mice).

In this chapter, we describe a simple robotic system for precise instrumentation guidance that includes the features of the listed commercially available products such as, dexterity, precision, steadiness as well as several safety mechanisms. The robotic system is conceived to be a direct replacement to the stereotactic frame, hence the surgical workflow is kept unchanged for the most part.

A robotic manipulator was selected, based on repeatability, compactness and controller accessibility. The control application developed includes a registration module to calibrate the robot to the surgical reference frame and several algorithms that allow a precise and consistent, drilling and electrode guidance. Safety was the most important factor when designing the control application (Faria et al., 2016).

4.1 Robotic System Role

The first question to contemplate is exactly what are the robot tasks as a surgical assistant. This problem was partially addressed in chapter 3.

To keep the robotic system simple and yet effective, we decided on a supervisory controlled, neuronavigation robotic assistant to hold instrumentation along pre-defined surgical trajectories. This implies that the robot motion performed during the operation is explicitly or implicitly specified beforehand, by the surgeon. During the procedure, the robot moves autonomously, but under close supervision.

To act as a direct replacement for the stereotactic frame, the system should be able to interpret the stereotactic coordinates from the planning software and translate them to the robot's own reference frame.

Thus, the first step consists in mapping the surgical entry and target coordinates or frame coordinates, from the surgery reference frame to the robot's reference frame, using a registration method. To avoid adding extra vision systems, commonly used for registration purposes, we resort to a touch-based point-registration method. A custom designed “pivot” end-effector is attached to the robot's flange and used to touch divots in the stereotactic ring that holds the patient's head and establishes the surgery reference frame, Figure 4.1a.

Once the surgery coordinates are defined in the robot's task space, the user can command the robotic manipulator to position itself along the pre-planned trajectories and hold the instrumentation for the trepanning or electrode insertion step, Figure 4.1b.

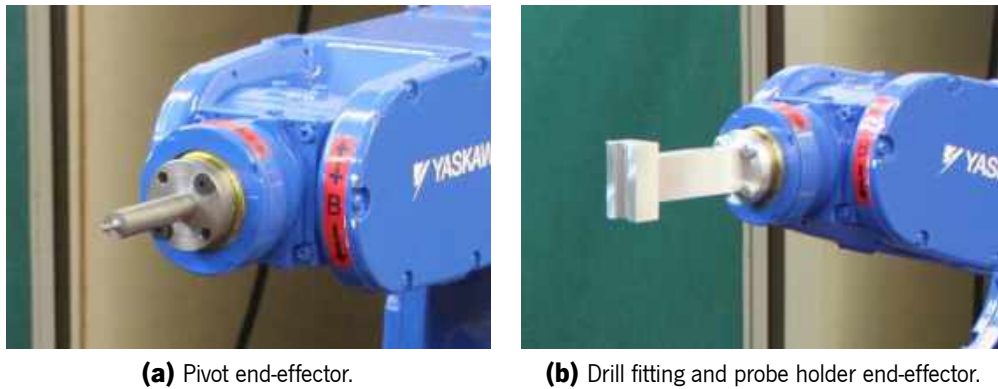


Figure 4.1: Customized surgical instrumentation.

The motion of the robot from its home position to the target pose is divided into two steps:

1. **Positioning:** the robot moves to align its tool with the electrode insertion trajectory defined by the entry and target points, from its home position or way-pose¹. Namely, when the robot moves its tool away from the electrode's trajectory, back to its home position or way-pose.
2. **Manipulation:** the robot tool is aligned with the electrode insertion trajectory and the robot is only allowed to move its tool along the electrode's trajectory. A scalar value quantifies the distance from a surgical reference point (entry or target) to the instrumentation tip.

Being a supervisory controlled robot, when it reaches the target pose (e.g. to hold the trepan) it locks its joints, for safety reasons. With the robot locked, the neurosurgeon manually maneuvers the drill - which is depth- and orientation-restricted by the robot tool - to perform the trepanation. Afterward, the robot repositions its Tool Centre Point (TCP) to reach the target pose for the electrode's drive.

4.2 Setup description

The proposed robotic system goal is to provide a direct and seamless replacement to the stereotactic frame. After testing different robotic systems in a simulation environment (Faria et al., 2013, 2015b), we implemented the control code in the most fitting robot platform. The system described in this chapter

¹Intermediate pose, where the robot passes from the initial to the target pose.

includes a 6 DoF robotic arm and respective controller, a stereotactic phantom to test the target position and a remote computer host, Figure 4.2.

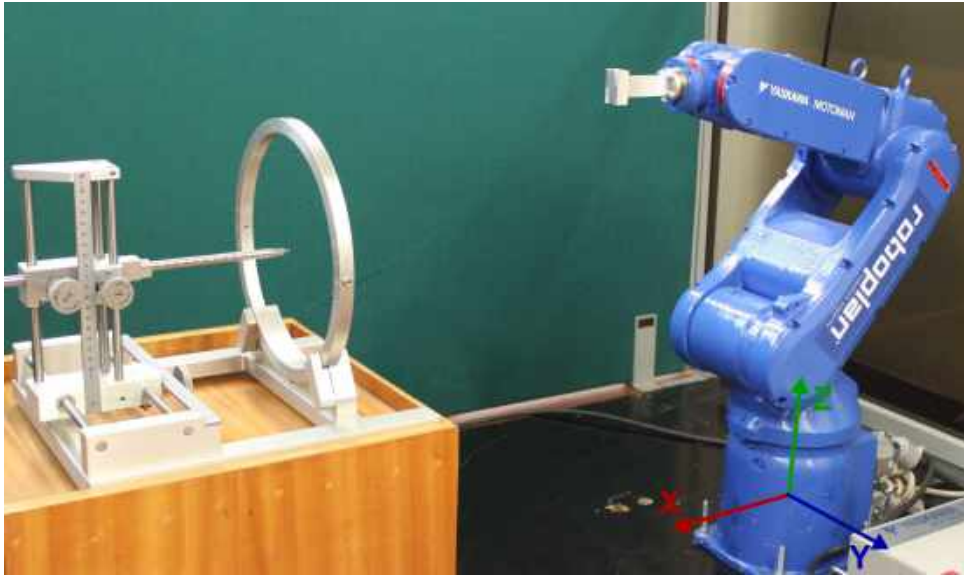


Figure 4.2: Working scenario with the Zamorano-Dujovny phantom device to simulate the target, and Yaskawa Motoman MH5 robotic arm.

The remote computer interacts with the robot's controller to send the commands that move the robot at each surgery step. Additionally, some custom end-effectors were fabricated to handle the electrode driver and drilling fitting, as well as to perform the point-registration.

4.2.1 Motoman MH5

The lightweight Yaskawa Motoman MH5 robotic arm and the NXC100 controller were selected for their repeatability, compactness and remote control feature, Table 4.1. The internally routed cables and hoses

Table 4.1: Yaskawa Motoman MH5 robotic arm specifications.

Axes	6
Payload	5 kg
H. Reach	706 mm
Weight	27 kg
Repeatability	0.02 mm
Installation Position	Any
Protection Class	IP 54

maximize the system portability and facilitate the system cover-up with sterile drapes. The robotic arm is accompanied by the NXC100 small-size certified controller, that incorporates all the basic motion and teaching commands. Most importantly, the controller features a remote connection protocol through standard TCP/IP to access the robot status and send commands in run-time, the "Ethernet Server Function".

A local network was created and the connection established the remote host and NXC100 controller. Static IPv4 addresses are defined in both the host machine and in the controller within the network, and the controller is shift to “Remote” control mode, Figure 4.3.

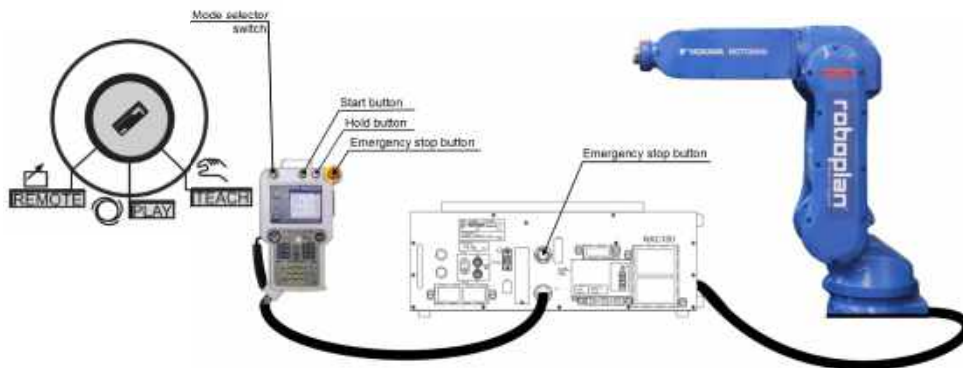


Figure 4.3: Motoman MH5 robotic arm, NXC100 controller and programming pendant (different working modes).

By default, the controller connects a socket to the TCP port 80, which is listening to command requests. The remote control API includes an extensive list of functions to: i) read and monitor the robotic system status; ii) control system operation mode, servo and brakes actuation; iii) command joint, task space or incremental movements according to specified reference frames; iv) access to I/O signals, etc. Any command request from the host to the controller follows a series of protocol message exchanges, that start by connecting a socket and requesting robot access, then to specifying the command(s) and parameter(s), to the receiving the error code (and requested status), and then closing the socket².

4.2.2 Middleware

The proposed system expandability, integrability, and modularity perks derive from the middleware communication layer, YARP (Yet Another Robot Platform) (Metta et al., 2006). This open-source software library offers tools to implement a language and operating system independent communication protocol between distributed components in a network. A minimalistic message standard (Figure 4.4) was used to convey information between the two components in our system, the host computer that connects with the robotic controller and the developed control application for the neurosurgery scenario.

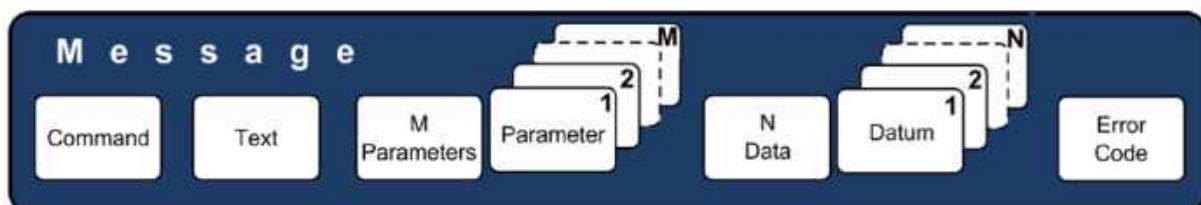


Figure 4.4: YARP message format, containing a key-command field, a text field, floating and integer parameters, and finally an error code (Machado et al., 2012).

²For more information about the “Ethernet Server Function” consult the Yaskawa manuals.

To comply with the middleware standard, a wrapper component was developed in C# to smooth the communication between the network and the robotic controller. The wrapper, named “translator”, implements a cleaner and YARP compatible interface of the “Ethernet Server Function” commands. The translator software permits single-command requests or setting up a continuous flux of messages, e.g. reading the system status.

4.3 Control Application for Neurosurgery

On the other side, we find the control application developed for the proposed neuronavigation robot assistant. The application is written in MATLAB for a multitude of reasons including rapid prototyping, GUI development interface and familiarity with end-users.

The control application includes: i) the registration module to calculate the transformation between the robot and the stereotactic ring, ii) the coordinate conversion module to translate the ZD-frame coordinates to Cartesian space, iii) the motion planning module that defines the robot movements during each step, according to the NXC100 controller functions, iv) the communication module and v) a user interface module. The graphical user interface (GUI) includes simple surgery related commands and different windows to guide the neurosurgeon through each procedure step.

4.3.1 Communication Module

The communication module is the control application component responsible for the interaction with the YARP middleware layer (subsection 4.2.2). The communication is established so that the robotic controller acts as a server that feeds information and answers to the MATLAB application requests. The client-server communication uses TCP to guarantee a reliable, ordered and error-checked transmission of messages (Figure 4.4). The abstraction of message standard requires the nodes that communicate over the network to have an additional marshaling/unmarshaling step.

The NXC100 commands available to the middleware need to be mirrored in the control application communication module. This module supports reply-required and non-reply message exchange. Structure of a reply awaiting request,

```
[rCommand, rText, ruparams, rfdata, rErrorCode] = SendWithReply(command,
description, uparams, fdata, errorcode)
```

The `SendWithReply` method is a synchronous blocking operation that awaits a reply from the network. For completeness, the `SendWithoutReply` allows the program to forward a command without waiting for any reply. All messages exchanged with the robotic controller fall within the `SendWithReply` category since the reply was always required to confirm the correct message exchange and operation execution.

To send a command, the user needs to supply the message with a key integer `command` to uniquely identify the requested operation with a facultative `description` field containing a string variable. The

uparams and fdata contain vectors of integer and floating point numbers respectively. The errorcode contains an integer value.

The message received has 5 analogous fields. The rCommand is expected to contain an integer incremented by one from the command sent. The rText command generally contains a string with a small description of the errorcode received. The ruparams and rfddata contain vector of integer and float values. The errorcode returns: **0**, successful command; **-1**, command not accepted; **-2**, normal connectivity with the robot controller not allowed (e.g. alarm); **-3**, YARP not running; **-4**, implemented exception handling; **-5**, other errors.

The implemented command functions check on the system's state and availability to perform the required action, before transmitting motion commands such as Cartesian or joint motion.

4.3.2 Registration

In an effort to keep the procedure as close as possible to the current surgery procedure, we decided to use the ZD frame as the head fixating device and surgery frame reference. The preoperative planning is conducted according to the established standard and the instrumentation trajectories identified in the surgery reference frame. Then, a point based co-registration approach is employed to find the transformation matrix that maps the robot to the surgery reference frame, refer to subsection 2.1.1.

The general idea consists of selecting a set of points with known coordinates in the frame of reference (surgery frame), and use a measuring device to detect these points in their own reference system. The closest path closes the loop and achieve this registration is to operate the robotic manipulator as the detection device.

The ZD base ring contains several divots engraved in its structure at well-defined positions (${}^S\mathbf{p}_{h1,2,3,4}$) in the surgery reference frame,

$$\begin{aligned} {}^S\mathbf{p}_{h1} &= [-126.1, \quad +18.8, \quad 0.0], \\ {}^S\mathbf{p}_{h2} &= [+126.1, \quad +18.8, \quad 0.0], \\ {}^S\mathbf{p}_{h3} &= [-126.1, \quad -18.8, \quad 0.0], \\ {}^S\mathbf{p}_{h4} &= [+126.1, \quad -18.8, \quad 0.0]. \end{aligned} \tag{4.1}$$

The robotic arm is maneuvered to sequentially touch each of these points, and their position (${}^R\mathbf{p}_{h1,2,3,4}$) is recorded relative to the robot's base frame using forward kinematics, Figure 4.5.

Once the two sets of identified in both surgery and robot reference frame, the rigid transformation that maps one set of points into the other (${}^R\mathbf{T}_S$) is computed based on a least squares fitting algorithm (Horn, 1987).

Horn proposed an optimization algorithm to find the affine relationship between pairs of measurements in different reference frames. The algorithm takes 3 or more pairs of points and starts by computing the translating component of the transformation that is based on the distance between centroids. The scaling between the two sets of points is calculated using the root mean square deviations of each set of points from their respective centroid. The quaternion representing the best fitting rotation is computed

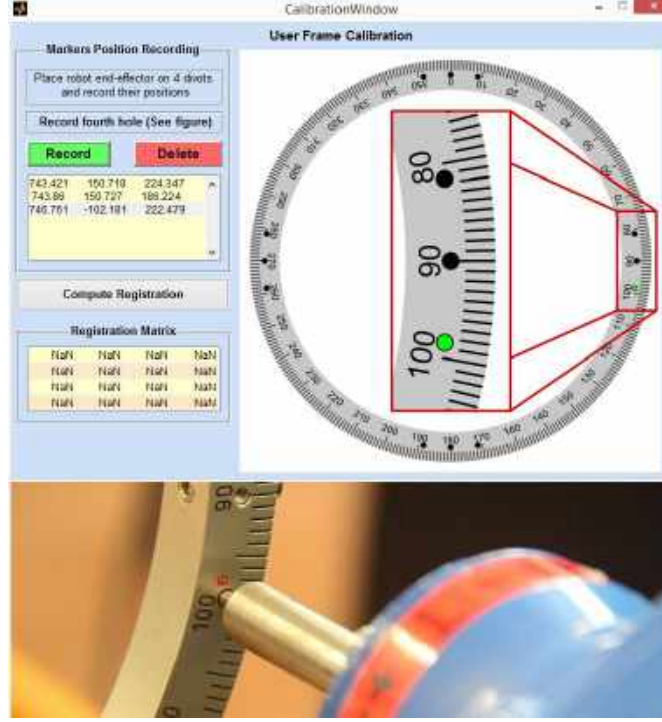


Figure 4.5: Point based co-registration process. A “pivot” end-effector tool is used to touch four selected divot markings on the reference ring, sequentially.

from the eigenvector with the most positive eigenvalue of a symmetric 4×4 matrix. Matrix that results from arithmetic operations between sums of products of coordinates from both sets of points.

4.3.3 Coordinate Convention

The Coordinate Convention module is no more than a package that handles the surgical trajectory coordinates from the preoperative plan. The developed library accepts two types of inputs: i) the Cartesian coordinates of the target and entry point as defined in the surgery reference frame; or the ii) ZD frame coordinates (we worked with the available ZD frame, although the process can be replicated to other commercial frames). To act as a direct replacement for the frame, the control software should handle the frame coordinates as output from the image planning software and translate them into coordinates known in the robot space.

4.3.3.1 Cartesian coordinates Target and Entry Points

In case the target ($\mathbf{p}_t$) and entry points ($\mathbf{p}_e$) are specified in Cartesian coordinates, the surgery trajectory representation as a vector (${}^e\mathbf{v}_t$) in space is straightforward,

$${}^e\mathbf{v}_t = \mathbf{p}_t - \mathbf{p}_e, \quad (4.2)$$

however, the robotic system usually requires a completely defined transformation matrix as a motion reference. It is not possible to uniquely construe a reference frame, from a directional vector in space, because one orientation variable is unspecified. This is a typical case where the robot kinematic chain is

considered redundant because the number of DoF required for the robot to conform to the trajectory is less (5) than the number of structural DoF (6).

The homogeneous transformation matrix that represents a surgical trajectory ($\mathbf{T}_1$) is constructed from the target and entry points as depicted, Figure 4.6a.

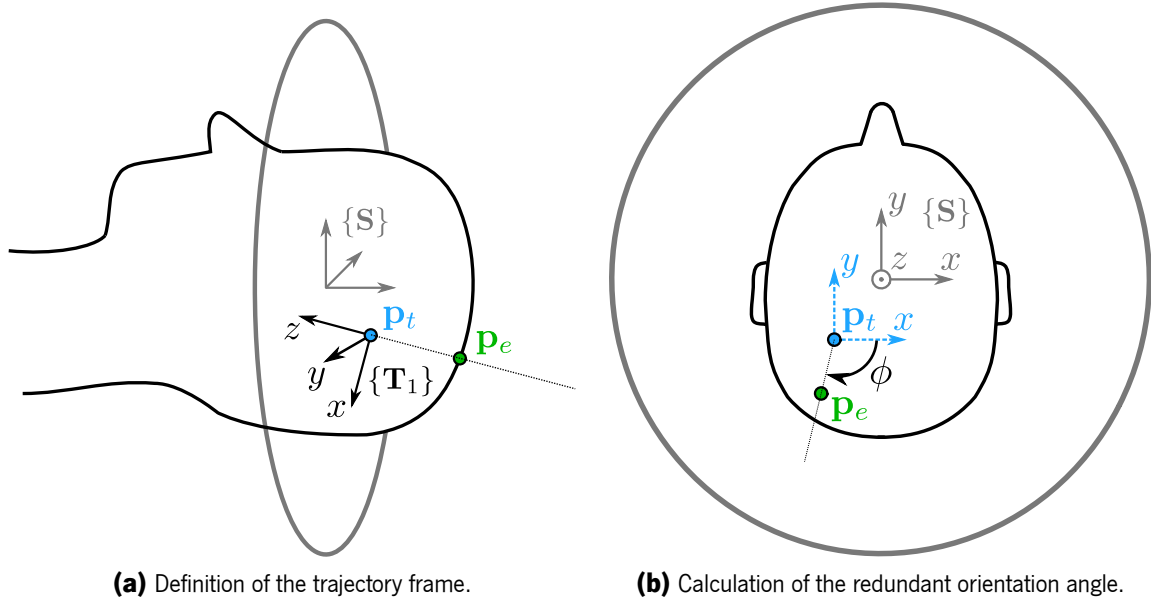


Figure 4.6: Representation of a surgical trajectory reference frame extracted from the target and entry points.

The position sub-matrix ($\mathbf{p}_t$) is directly defined by the target Cartesian coordinates. By definition, we chose to assign the surgery trajectory unit vector (${}^e\hat{\mathbf{v}}_t$) to the rotation sub-matrix z-axis vector,

$$\mathbf{T}_1 = \left[\begin{array}{ccc|c} \mathbf{R}_{x,x} & \mathbf{R}_{y,x} & {}^e\hat{\mathbf{v}}_{t,x} & \mathbf{p}_{t,x} \\ \mathbf{R}_{x,y} & \mathbf{R}_{y,y} & {}^e\hat{\mathbf{v}}_{t,y} & \mathbf{p}_{t,y} \\ \mathbf{R}_{x,z} & \mathbf{R}_{y,z} & {}^e\hat{\mathbf{v}}_{t,z} & \mathbf{p}_{t,z} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4.3)$$

The end-effector angle around z-axis is undefined, and as a result the rotation sub-matrix vectors $\mathbf{R}_x$ and $\mathbf{R}_y$ are unspecified. We start by calculating temporary axes ($\mathbf{R}'_x$ and $\mathbf{R}'_y$), based on an arbitrary vector³,

$$\mathbf{R}'_y = \begin{cases} right, & \text{if } (front \cdot {}^e\hat{\mathbf{v}}_t) = 1, \\ \frac{{}^e\hat{\mathbf{v}}_t \times front}{\|{}^e\hat{\mathbf{v}}_t \times front\|}, & \text{if } (front \cdot {}^e\hat{\mathbf{v}}_t) < 1, \end{cases} \quad (4.4)$$

which follows,

$$\mathbf{R}'_x = \frac{\mathbf{R}_y \times {}^e\hat{\mathbf{v}}_t}{\|\mathbf{R}_y \times {}^e\hat{\mathbf{v}}_t\|}. \quad (4.5)$$

The angle ϕ is calculated from,

$$\phi = \text{atan2}(-{}^e\hat{\mathbf{v}}_{t,y}, {}^e\hat{\mathbf{v}}_{t,x}). \quad (4.6)$$

³To facilitate the vector representation, let $front = [1 \ 0 \ 0]$, $right = [0 \ 1 \ 0]$ and $up = [0 \ 0 \ 1]$

Let,

$$\text{rot}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \text{rot}_x(\psi) = \begin{bmatrix} \cos \psi & 0 & \sin \psi \\ 0 & 1 & 0 \\ -\sin \psi & 0 & \cos \psi \end{bmatrix}, \text{rot}_z(\phi) = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.7)$$

The final rotation matrix of $\mathbf{T}_1$ is calculated as a product of the temporary matrix and the rotation matrix around the z-axis ($\text{rot}_z(\phi)$),

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}'_x & \mathbf{R}'_y & {}^e\hat{\mathbf{v}}_t \end{bmatrix} \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.8)$$

The transformation matrix here proposed, defines the target position and an initial orientation. Although the reference frame does not contain information about the entry point, it can be directly calculated by translating along the z-axis of a distance of $\|{}^e\hat{\mathbf{v}}_t\|$. During the procedure, the user has the opportunity to adjust the end-effector orientation around the z-axis, while keeping it aligned with the surgical trajectory. In this case, the new motion reference is obtained by changing ϕ (4.8).

4.3.3.2 Zamorano-Dujovny coordinates

The system is also endowed with an algorithm to convert ZD coordinates to a reference frame representing a trajectory. It should be noted that the ZD frame is designed to have a single point fixation (subsection 2.1.2) and can be mounted in the 4 quadrants of the base ring (positions 0° , 90° , 180° , 270°). Consequently, the ZD coordinates depend on the stereotactic frame mounting position, Figure 4.7.

The A, B and C coordinates specify the target position (Table 4.2), whilst the D and E coordinates

Table 4.2: Target point ($\mathbf{p}_t$) Cartesian coordinates from ZD coordinates in each of the frame mounting positions.

	0°	90°	180°	270°
$\mathbf{p}_{t,x}$	B	A	B	A
$\mathbf{p}_{t,y}$	A	B	A	B
$\mathbf{p}_{t,z}$	C	C	C	C

specify the surgical trajectory orientation. The D coordinate represents a rotation around A-axis, and E represents a consecutive rotation around B-axis. Then, the rotation matrix can be extrapolated from D and E as,

$$\mathbf{R}' = \begin{cases} \text{rot}_y(-(90^\circ - D)) \text{rot}_x(-(90^\circ - E)), & \text{if frame mounted at } 0^\circ, \\ \text{rot}_x(-(90^\circ - D)) \text{rot}_y(90^\circ - E), & \text{if frame mounted at } 90^\circ, \\ \text{rot}_y(90^\circ - D) \text{rot}_x(90^\circ - E), & \text{if frame mounted at } 180^\circ, \\ \text{rot}_x(90^\circ - D) \text{rot}_y(-(90^\circ - E)), & \text{if frame mounted at } 270^\circ, \end{cases} \quad (4.9)$$

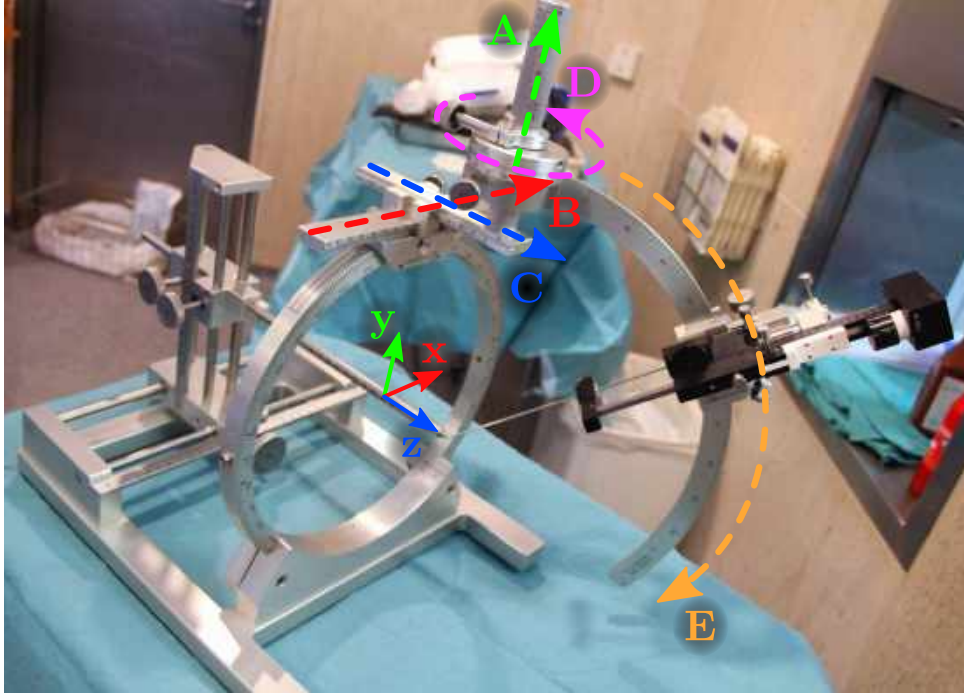


Figure 4.7: Example of ZD coordinates with the stereotactic frame mounted in position 0° .

A final rotation is applied to point the z-axis trajectory vector towards the target point from the entry point,

$$\mathbf{R} = \mathbf{R}' \text{rot}_x(180). \quad (4.10)$$

4.3.3.3 Rotation Matrix and Euler Angles

The NXC100 controller works exclusively with Euler angles to represent rotations while the controller application with rotations in the form of matrices and quaternions. Matrix to/from quaternion conversions are unambiguous, but the same cannot be said for Euler to/from matrix form conversions, as there are up to 12 sequences divided between the proper Euler angles and Tait-Bryan angles. The convention used by the NXC100 controller is the extrinsic Tait-Bryan x –, y – and z –, that relates to the angle coordinates (θ, ψ, ϕ) . Translating from angle coordinates to a matrix form,

$$\text{rot}_{x,y,z}(\theta, \psi, \phi) = \text{rot}_z(\phi) \text{rot}_y(\psi) \text{rot}_x(\theta) \quad (4.11)$$

and from the matrix form to the angle coordinates,

$$\begin{aligned} \theta &= \arctan_2(r_{32}, r_{33}) \\ \psi &= \arctan_2(-r_{31}, r_{32} \sin \theta + r_{33} \cos \theta) \\ \phi &= \arctan_2(-r_{21}, -r_{11}) \end{aligned} \quad (4.12)$$

The r_{ij} variables correspond to the matrix coefficient at i^{th} line and j^{th} column.

4.3.4 Robotic Motions

To keep the system simple yet functional, it was used the motion commands of the NXC100 controller. The commands listed below are common among industrial controllers. For this particular application, three types of motions were used.

- Joint motion, typical point-to-point motion where the position of each joint is interpolated to move from the initial to the target position individually, but synchronously; the end-effector trajectory in Cartesian space is not relevant.
- Linear motion, each joint executes a coordinated motion, such that the end-effector describes a line in Cartesian space to reach the specified Joint or Cartesian position;
- Incremental motion, similar to Linear motion, however, it is specified the increment of the end-effector position in Cartesian coordinates (only), instead of the absolute target;

Additional parameters for the motion commands include joint and Cartesian velocity limits, the tool dimensions, and the reference frame whenever Cartesian coordinates are involved. The motion planning was divided into the approach and the tool handling phase. The approach phase involves the movements between the robot home or safe position and the stereotactic trajectory – manipulator is at least 250 mm away from the patient's head. The tool handling movement phase starts when the manipulator end-effector is already aligned with the stereotactic trajectory and it just translates along this trajectory. To enforce safety, the manipulator joint speed is maxed at 15% during its approach phase, and 3 to 5% during the tool handling phase. As neither high-forces nor velocities are required in stereotactic neurosurgeries, functioning at a lower speeds causes less stress to the joints, makes the robot motions more predictable and provides the operator more control over the robot actions (Fei et al., 2001).

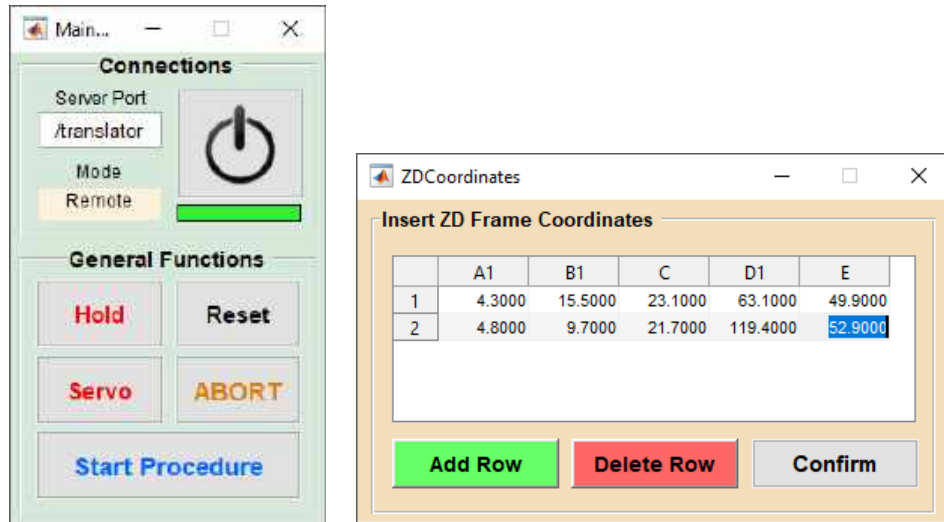
4.3.5 User Interface

The control application launches with the MainWindow, where the user can input the Server Port field with the name of the peer YARP component to connect to (in this case, the “/translator”). It expects a running YARP server, as well as a running module that interfaces with the robotic application. If the communication is established a green led is light up below the connection button, and the “Mode” label will indicate the NXC100 controller functioning mode. If the mode is set to Remote the user can now remotely control the manipulator; also an additional group of buttons appears under the “General Functions” panel to manipulate basic robot functions such as, activate the servos or trigger the joint brakes, Figure 4.8a.

When a new procedure is started, the registration window shows up and asks the user to move the robot (with the attached pivot) to reach the reference ring divots. The divots should be touched in a specific order, recording the robot's position after reaching each one. When the final position is recorded, the transformation between the robot and the reference ring is calculated. For more information about this step please refer to subsection 4.3.2.

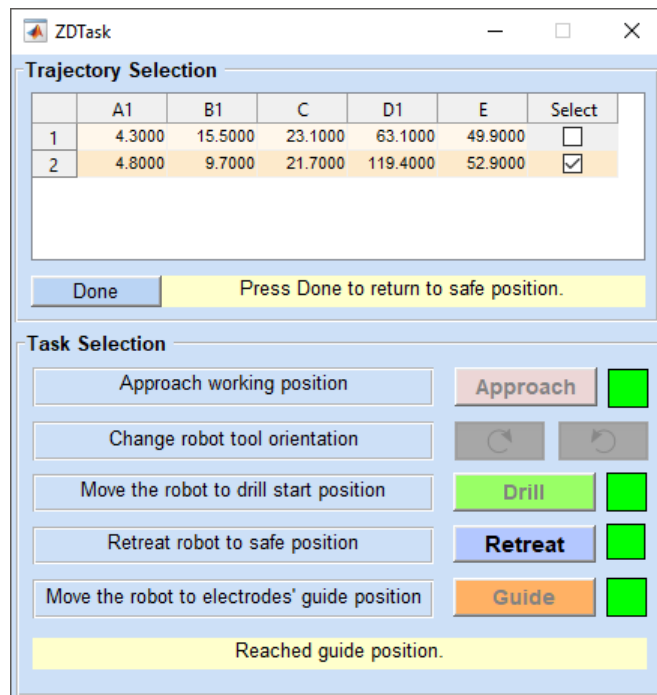
After the registration step, a question dialog appears to select the coordinate convention to use (check subsection 4.3.3). If the user is working with ZD coordinates a window such as Figure 4.8b appears,

otherwise, a similar window shows up to introduce target and entry point coordinates. These coordinates should be directly available from the planning software. Once the trajectory coordinates are checked and confirmed, a new window will pop-up Figure 4.8c, again, with different coordinates depending on the chosen convention.



(a) MainWindow, connection interface, basic robot controls and button to launch procedure.

(b) ZDCoordinates, ZD frame coordinate insertion interface; an alternative interface for target and entry point input is available by user choice.



(c) ZDTask, window that shows the motion options available for each selected trajectory.

Figure 4.8: Graphical user interfaces of the control application.

The ZDTask window Figure 4.8, appears with the inserted trajectories. The GUI contains several tips to guide the neurosurgeon throughout the process, prompting messages between each step and to verify any robot motion. The “Trajectory Selection” panel contains a list of the mutually exclusive trajectories

previously inserted as well as an Instructions label to guide the neurosurgeon through each step. Below, the “Task Selection” panel contains several buttons that permit the user to maneuver the robot during the sequence of motions to surgically introduce the electrode. The robot actions can be summarily described by a finite-state-machine, where each button represents an input and each step (Selection, Approach, Adjust, Drill, Retreat, Guide) a state. A led next to the button will indicate a successful transition to the new state, which depends on the user allowance, reaching the target frame within the precision interval and the absence of any hardware, software interruptions.

4.4 Procedure Workflow

The developed control application provides a platform for the user to insert and execute the preoperative plan when connecting the control application to the robot’s workstation. The system’s user interface guides the surgeon through the execution of this plan - that defines actual trajectories of electrodes toward the target – by following a sequential task-based controlled process for positioning and orienting the robotic arm according to the selected trajectories. This process is divided into six main steps, presented below.

To assess the system’s performance, it is conducted a complete demonstration of the robotic role in an assisted stereotactic neurosurgery using the developed control application⁴. It is portrayed the different postures assumed by the supervisory-controlled robotic manipulator, when performing the different tasks, using snapshots of the actual system. At the same time, it is shown how the system’s user interface dynamically changes as the procedure advances.

Approaching the selected trajectory After selecting the trajectory, the Approach task is ready to be executed. It is calculated the end-effector coordinates to position the robot’s tool holder z-axis collinear to the stereotactic trajectory, distanced 350 mm from the target. Then, the robot executes a sequential Joint motion from its home/safe position to this new calculated reference. Immediately after, another command is sent to move the robot in a Linear Motion along its tool reference z-axis until reaching a distance of 250 mm from the target. The neurosurgeon can now attach the surgical tool to the tool holder, and adjust the robot’s orientation to a more favorable posture. This change in the robot’s orientation involves rotating the robot’s end-effector around its tool’s z-axis in steps of 10° , using an Incremental motion and without changing any other position or rotation variables.

Assisted-drilling When executing the Drill task, the manipulator advances its end-effector in an Incremental motion collinear to the electrode’s trajectory towards the target. The distance between the tool and the target was set in accordance with the drill length. After reaching the target position, the joint brakes are engaged and the neurosurgeon may perform the drilling step. The tool holder controls the drill orientation as well as its depth, to avoid damaging structures beyond the skull’s inner wall, Figure 4.9.

⁴The procedure demonstration can be visualized at <http://marl.dei.uminho.pt/public/videos/RoboticNeurosurgeryDemonstration.html>

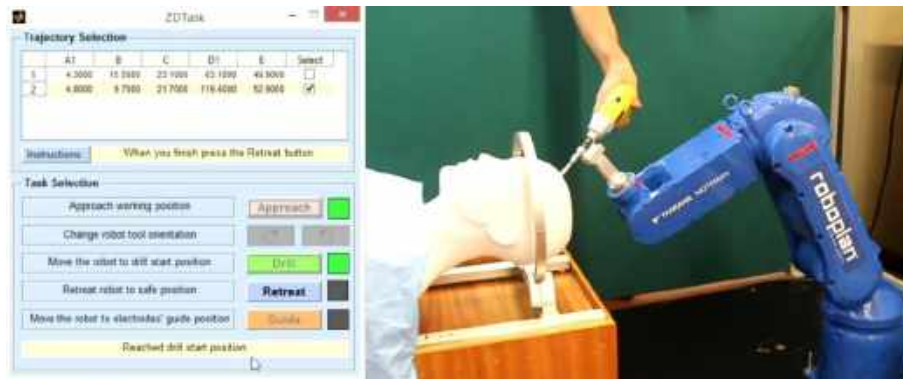


Figure 4.9: Performing assisted drilling, with orientation and depth-control.

Returning to a safe position At this point, the Retreat task is the only step available. By selecting it, the robot moves its end-effector in a Linear motion along the electrode's trajectory out-wards/away from the patient's head, until a distance of 250 mm from the target. The current tool is substituted by the electrode's microdrive.

Assisted electrode's guidance Once more, the robot advances its end-effector collinear to the electrode's trajectory towards the target in an Incremental motion (again, for demonstration, the distance was set in accordance to the microdrive with the defined electrode depth), and assisted electrode's guidance is executed. The surgery phantom device is used to simulate the target coordinates instead of the dummy model in order to demonstrate the electrode, positioned along the selected trajectory, reaching the selected target, Figure 4.10.



Figure 4.10: Holding microdrive to position electrode at the surgical target.

Finishing the procedure - returning to home position In the end, the robotic manipulator retracts in a composed motion mirroring the initial approach movement. Initially, the robot moves its end-effector outwards and along the trajectory in a Linear motion until reaching 350 mm from the patient's head. Then, and after removing the electrode's microdrive, the robot moves to the home position in a Joint motion.

By following a sequence of centripetal motions when approaching or moving away from the patient, it makes the robot action more predictable and avoids movements between trajectories that could potentially

collide with the patient. For safety reasons and when initiating a new procedure – before approaching a selected trajectory – the robotic controller checks the robot and connection state, and if the arm is at its home position. If the robotic system gathers all the ideal conditions to operate, the user is informed and asked to give consent to move the robotic arm. Whenever the robot moves, the servo power is activated, and the user is informed that it is moving. When the robot is not moving, its joint brakes are activated. At any moment the user can stop/prevent the robot movement, either using a hardware or software trigger - Emergency and Hold buttons - found at the programming pendant, robot controller or in the graphical user interface. If the desired position/orientation is not reached, in case robot movements are stopped during the process or, for example, if the robot reaches a joint limit, an alarm is triggered reporting the position and orientation errors. When executing a stereotactic procedure using the developed control application, task buttons are enabled/disabled according to the next step of the procedure, guiding the user through the execution of the procedure. During the robot motion, all task buttons are disabled. A green/red color is used to notify the user for the success of performing the required task.

4.5 Discussion

Robotic intervention in neurosurgery is a growing trend with new market available systems and continuous technological breakthroughs. Robotic agents offer multiple advantages when compared to more traditional hardware like stereotactic frames, namely, in terms of improved precision, consistency and flexibility. However, market available robotic systems are pricey and closed. Here it was presented a simple solution that has the potential to work as a direct replacement to the stereotactic frame, combining the advantages of a robotic system while scaling down on system complexity. It provides an easily integrable, expandable and low-cost system for stereotactic neurosurgery. Even though the system is not permitted to perform neurosurgery as-is, it can be used as a proof-of-concept instrument, or in a research environment to catalyze the development of new systems for stereotactic neurosurgery.

Chapter 5

Modular Architecture

Throughout this chapter, we describe how the developed framework for surgical robotics addresses the need for more accessible and cost-effective systems, how it shortens the development cycles of future surgical robots and facilitates the technological transfer.

5.1 Monolithic vs Modular Paradigm

Addressing the above-listed topics means to shift from the paradigm of the standard robotics controller as introduced in chapter 4, to an open, modular and distributed architecture. To further elaborate on the subject, we categorize robotic controllers as: i) monolithic systems and ii) modular architectures.

The monolithic system consists in a software controller block of one or more closely related components, designed to operate on top of a specific set of hardware. It can manage all tasks and keep performance because the system integration is confined to a small set of components. However, replacing a single device or driver implies a time-consuming system revamp. Additionally, it is unfeasible for peer institutions to conduct research or develop new control strategies without an exact replica of all devices that constitute the system.

The alternative - modular architectures - are controller schemes structured into a network of local or distributed independent components that are allocated to various functions such as, data processing, decision making, supervising operations, etc. These components are generally peered software blocks with a generic structure that can be pieced together to form arbitrarily complex robotic systems. The architecture uses a middleware environment to establish the communication between components.

Modular architectures advantages range from the reuse of code and components, speed-up of system development and encapsulation of proprietary technologies. Also, the component distribution through the middleware improves the computational capabilities and robustness as well as facilitates support for specialized devices (Bzostek et al., 2000).

5.2 Component-based Software Engineering

Component-Based Software Engineering (CBSE) can be described as an effective programming model specially tailored to handle large and complex systems. It pursues the separation of concerns principle that sections the system into loosely coupled working components. The CBSE promotes the idea of software reuse in two different forms: 1) opportunistic, the components are reused in different problems; 2) systematic, the components are built to fit similar applications, being reusable in a larger sample of problems. Thus, the paradigm of system development changes from, system analysis, design, and implementation, to composing a mixture of off-the-shelf custom-built components, which naturally reduces the development time and effort.

Similarly to other areas, the component-based models were adopted by the robotics community in response to the growing complexity and multifunctionality of robotic systems. With several sensors and actuators, robotic systems' software needs to handle concurrency, real-time requirements, distributed devices while guaranteeing properties such as safety, reliability and fault tolerance.

5.2.1 Concepts

According to the Object Management Group (OMG) Unified Modeling Language (UML), a component is formally defined as,

A class representing a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. One component may, therefore, be substituted by another only if the two are type conformant. Larger pieces of a system's functionality may be assembled by reusing components as parts in an encompassing component or assembly of components, and wiring together their required and provided interfaces.

From the definition, we take that a component is an independent agent capable of functioning outside a particular system while maintaining self-contained internal states and properties. It is a replaceable unit, potentially reusable in alternative applications by connecting to one or more components through their interfaces.

The component's provided interface encompasses methods realized directly by the component or provided public ports. The required interface, on the other hand, designates methods that the component depends on or required public ports. The component's interface is usually divided into:

- **ports** - are input/output typed connection terminals that allow a continuous data-flow between components;
- **methods** - are analogous to functions that take arguments and return a value. These are callable from other local components or distributed from another process or across the network.
- **properties** - are the component's variables that can be configured to tune its state or operation.

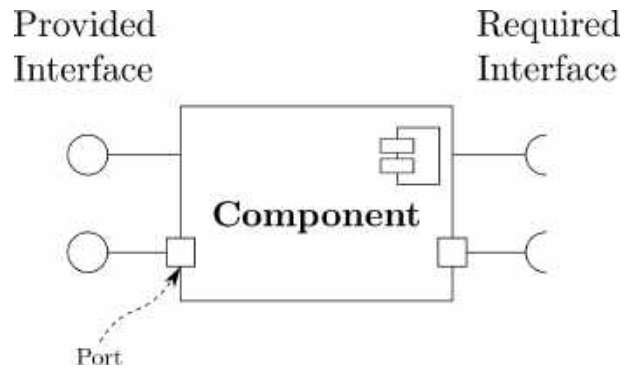


Figure 5.1: Component representation according to UML.

Once all components are coded and compiled, they need to be organized into an architecture that performs as expected from the system. That is the responsibility of the **deployer**. This tool handles different tasks ranging from importing and loading the different components to connecting and configuring their interfaces and finally to control their life-cycle (i.e, initialization, start, run, stop and cleanup). By tweaking the deployment setup, one can reassign components into a new architecture suitable to alternative applications, or reconfigure existing architectures to integrate new components.

5.2.2 Alternative Robotic Frameworks

The combination of CBSE and Robotics is not a new reality. There are, in fact, several component-based cores that were specifically designed to build robotic applications. In this subsection, we list the most popular of these robotic middlewares/frameworks. We point out the main features and drawbacks that eventually lead to our choice, the OROCOS project.

Player Project is an open source software project¹ composed by a collection of libraries, devices and simulation tools for rapid development of robot and sensor systems (Vaughan et al., 2004). The project itself is divided into three parts:

1. Player offers a network interface (Player Abstract Device Interface - PADI) to a variety of hardware device drivers. It provides a language and system independent interface to connect devices based on a client/server model. Supports multiple concurrent client connections enabling distributed and collaborative sensing and control.
2. Stage is a multiple robot 2D simulator for mobile robots with several abstract sensor models. It has a standard Player interface to facilitate the transition between simulation and the real hardware.
3. Gazebo is a multiple robot 3D simulator for outdoor environments analogous to the Stage simulator. Its own native interface is extended with a Player standard interface.

With regard to the disadvantages, this framework does not fully support Windows OS nor does it comply with hard-real-time requirements.

¹<http://playerstage.sourceforge.net/>

ROS abbreviation for Robot Operating System², is the fastest growing and most widely used open source platform for developing robotic applications (Quigley et al., 2009). It provides a set of tools and services to catalyze the process of building complex and robust robot systems. The success of the platform is, in large part, due to the seamless installation and intuitive usage, which is reflected by the size of the user-base. It also results in the development and sharing of more and more diverse control software solutions, as well as introspection tools. ROS exceeds any other robotics framework in terms of the completeness of provided libraries, for subjects as comprehensive as navigation, localization, planning, and manipulation. At its core is the middleware, a communications infrastructure that provides message publish/subscription, recording/playback, a distributed parameter system and a request/response mechanism to execute remote calls. With all that being said, ROS is still fundamentally supported in Linux distributions and as of now does not provide full hard real-time support, which is one of the main endeavors of ROS2 (in development).

Orca Orca is yet another open-source software project oriented to robotics applications based on CBSE principles (Makarenko et al., 2006). This project is in many ways close to other similar efforts in the sense that it seeks to accelerate the development process of robotics related software by promoting modular systems and software reuse. Orca's innovation aspects are tied to the use of a commercial open-source library for communication and component interface definition - ICE middleware, to the cross-platform development tools. The ICE middleware is a language and platform agnostic communication mechanism, an alternative to CORBA and XML-based technologies, which were considered exceedingly complex and inefficient, respectively. Orca is a close relative to the OROCOS project (we elaborate in more detail below), with the major differences being that OROCOS uses CORBA instead of ICE and that OROCOS supports hard real-time.

cisst When choosing between the different 'frameworks', the closest contenders were the cisst package and the OROCOS project. Other projects listed above lack either real-time support or do not run on conventional operating systems such as Windows. The cisst package besides complying with these requirements is also a middleware oriented for medical robots, thus deserving a look at.

The Computer-Integrated Surgical Systems and Technology, in short cisst, libraries are part of an open source, open architecture software package originally developed in the John Hopkins University, for medical robotic applications. Built on knowledge acquired from previous experience with surgical robots, ROBODOC (Taylor et al., 1994), the cisst project was originally developed to create well-tested, clinically certifiable medical robot controller frameworks. The framework design obeys four premises: 1) support commercial off-the-shelf and custom hardware/software solutions; 2) provide a documented and validated core software separated from the application layer; 3) the robot controller is a piece of a larger system and should provide a consistent and compatible interface; 4) the design of the framework should not condition the architecture hierarchical relationships (Kapoor et al., 2006).

The cisst package is a collection of software libraries grouped into categories:

²<http://www.ros.org/>

- Foundation Libraries, containing common utility tools, algebraic calculation, numerical methods, interactive environment;
- Real Time Support, operating system abstraction, device interfaces, tasks;
- Interventional Devices, trackers and robots;

Similarly to other frameworks, the cisst package builds on CBSE concepts, where the component is re-branded as a task. Each task encapsulates its own thread, a state table (time-indexed circular buffer) and one or multiple interfaces. The task inherent polymorphism conferred by its abstraction enables a task deployment manager to handle the loading, interaction, and execution of each module. To the cisst framework, this deployment tool uses a Python shell - Interactive Research Environment (IRE) to configure and interact with the local and global component managers (Kazanvides et al., 2011). The local component manager (LCM) permits a dynamic reconfiguration of the components within a process, i.e. creation, removal, connection, and disconnection. The global component manager (GCM), on the other hand, spans system-wide, handling LCM services as well. The IRE together with the Component Viewer tool provides a series of methods for data collection, interactive control, and introspection.

The multi-threaded environment is proposed as a complement to the multi-process and distributed alternative. The framework aims to centralize the processing (within a single process) and deal with the distributed I/O. To support this multi-threaded environment the cisst package includes efficient, lock-free and thread-safe mechanisms for data exchange; as native OS mechanisms (semaphores and mutexes) are deemed unfitting for real-time requirements due to the introduced over-head and the possibility of dead-locks (Kazanvides et al., 2008).

The inter-process and distributed system communication part of the cisst framework was only introduced afterward. The distributed communication model was designed to be consistent with the multi-thread implementation, meaning that the component's interface is dissociated from the connection - same process, between processes or across the network. The cisst currently uses ICE as the network layer by default, although an abstraction layer permits the use of alternative middlewares. The local connection is extended to distributed components by applying the "Proxy Pattern" design, similarly to other implementations (e.g. CORBA). Simply put, this technique uses surrogate objects that locally represent the real objects residing in different address spaces. The communication between real and proxy components is handled locally, and then the middleware is responsible for forwarding the requests between proxy components across the network to the encapsulated real object.

More recently, the Surgical Assistant Workstation (SAW) was introduced by the same developers of the cisst package as a modular, open-source framework for fast development of computer-assisted surgery systems. It includes a library of cisst components oriented for medical applications that encapsulate hardware devices, or software components that deal with robot motion, collaborative control, speech recognition, 3D user-interface, and video processing. The SAW components are designed to guarantee a certain level of functional abstraction, by establishing a compromise between the specificity of the task with the diversity of alternative hardware/software solutions. Without breaking the first premise of the cisst package, there was an effort to group components with a similar functionality under a standard interface and with this facilitate the substitution of analogous modules (Kazanvides et al., 2010). The developers go

one step further to adapt the components that deal with imaging data to have an interface compliant with established protocols such as, OpenIGTLink.

As described in these last paragraphs, the *cisst* package gathers a set of very suiting features to potentiate the development of medical equipment. This robotics middleware received several upgrades through the years, which is evidenced by the chronologically indexed literature. Looking from another perspective however, it is remarkably easy to establish a parallel between these new features and the OROCOS project. Moreover, it is explicit in the cited papers how the *cisst* developers try to demarcate from the later.

Facing the possibility to work with either framework, we chose to develop our system with the OROCOS project based on their larger user-base and on the quantity and quality of available documentation. Being a better known and more stable platform makes it easier to develop, troubleshoot and disseminate our solution. Its usage in surgical robotics frameworks is also a reality (Muradore et al., 2015; Daniele Comparetti et al., 2014)

5.2.3 OROCOS Project

The OROCOS project is an open source software framework for machine and robot control. Started in September 2001 as one of the first robotic middlewares, the project was designed to become the “one-stop” solution for all robotic software, from real-time motion control, through simulation and analysis of complex kinematic and dynamic systems, to intelligent sensor processing. Special attention was paid to maximize the system’s flexibility by: i) enabling a local or distributed control over a network, ii) extending its portability to several RTOS (Xenomai, RTAI, RTLinux, Ecos...) and non-RTOS (Linux distributions, Windows), iii) enabling the configuration and scalability of limited or fully functional high performance solutions (Bruyninckx et al., 2003).

5.2.3.1 The Component

The project’s selling point is its software component’s model. It focuses on thread-safe time determinism, which makes it a fitting candidate for critical control systems such as, surgical robotic controllers. The component, as the basic unit of functionality, is capable of executing one or more (real-time) programs in a single thread. A program can range from a mere C/C++ function over a script to a real-time hierarchical state-machine. The OROCOS component model enables:

- single process, lock-free, thread-safe inter-component communication;
- multi-process, thread-safe communication between distributed components;
- communication between hard and non-real-time components;
- deterministic execution time during communication for the higher priority threads;
- synchronous and asynchronous communication;
- run-time component introspection;
- C++ class implementations and scripting interface for all the above;

Components are implemented by subclassing the **TaskContext** object, a base class that defines the context/environment where each task is executed. At the core of the component is the ExecutionEngine, which is responsible for processing asynchronous operations, calling plugin functions and executing its programs. The ExecutionEngine is launched as soon as the **TaskContext** is created. It calls on the “*Hook()” implemented functions according to the task state or state transition, Figure 5.2.

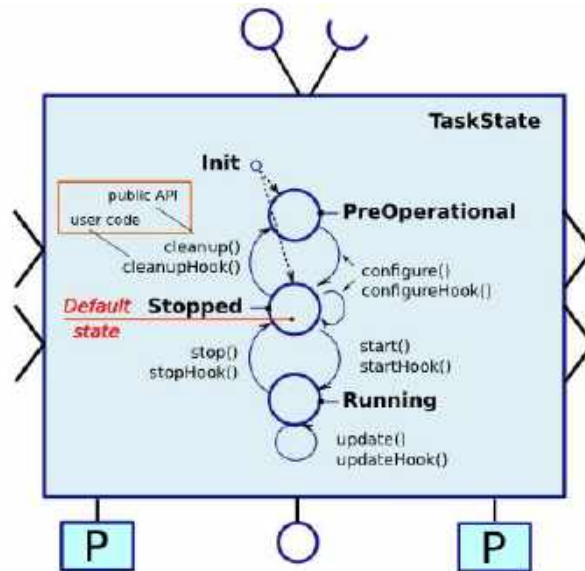


Figure 5.2: TaskContext state diagram. The component has 4 regular states: Init, PreOperational, Stopped and Running. After creation, the component is at the Init state, at construction it transitions to PreOperational or Stopped (default). If the component is at the PreOperational state, it needs to “configure()” before calling the “start()” method.

The code in the “configurationHook()” and “cleanupHook()” is used to read/write XML files, to allocate/free resources, or to perform any other operation that may compromise the real-time requirement. The real-time code belongs to the “startHook()”, “updateHook()” and “stopHook()”. The ExecutionEngine runs in a thread allocated by an Activity object. At construction, a period (0 if aperiodic), a priority and a scheduler are passed to build the Activity, which in its turn sets the pace for the ExecutionEngine.

As depicted in Figure 5.3, the component’s interface comprises **Operations** and **OperationCallers**, **Input** and **Output Ports**, and **Properties**. Additionally, the component’s functionality can be augmented by adding the provided (scripting, marshalling, etc) or custom plugins.

5.2.3.2 Data Ports

Data flow ports are the communication mechanism to send or receive a stream of data. An output port publishes data that can be received by a connected input port - both supporting multiple connections. The output port sends data whenever prompted, while the input port splits into: i) standard port, polls new data, ii) event port, that triggers a new running step once data arrives. Reading and writing data ports are always real-time and thread-safe, as long as the transferred data assignment operator is as well.

A port is defined by a unique name within the component, and a data-type. Any data-type can be exchanged by a port, as long as its type and transport are known. By default, standard C++ and

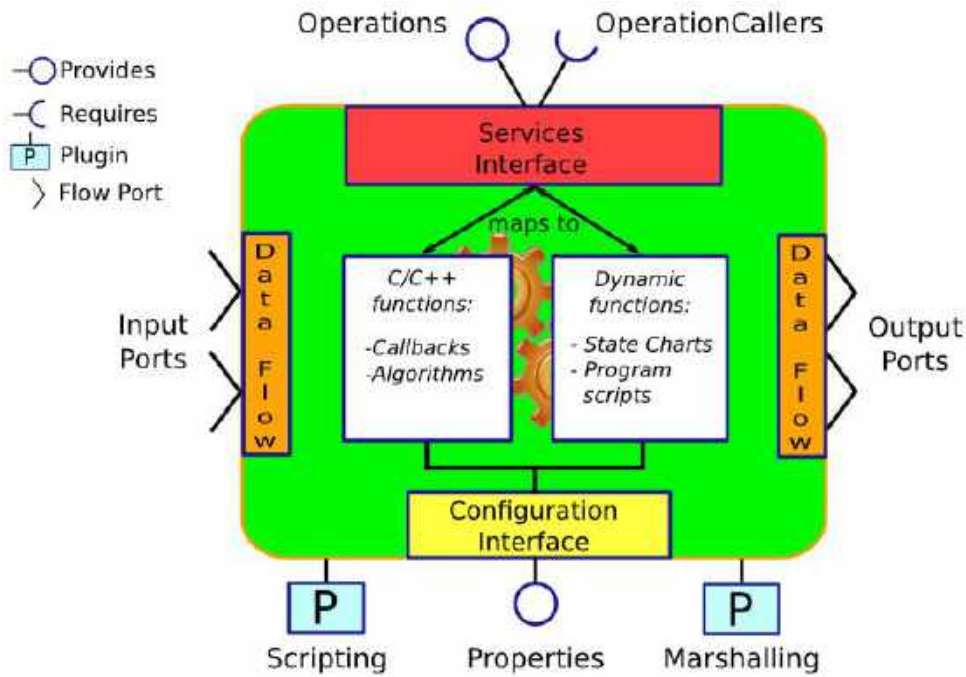


Figure 5.3: Schematic representation of a TaskContext.

“std::vector<double>” data types are supported. To include custom data-types, one must register the type and transport with the type system ³. Finally, the communication middleware allows transparent data port connection regardless of the involved component’s proximity (local, inter-process or distributed). The data port connection, much like any other interface type, can be effectuated in the coded functions or at run-time using the deployer tool (more information below).

5.2.3.3 Operations

The **Operations** and **OperationCallers** define what functions the component provides and requires, respectively. Implemented as C++ functions, operations can take multiple arguments and return a single value. They can be called from different processes or across the network as long as the types and transports of its arguments and returns are known.

When setting up a component’s operation, one must specify a name, a function pointer, as well as an execution type flag. The later indicates whether the operation is executed in the client’s or the owner’s thread. Client or caller thread execution does require data locks to avoid concurrent access to shared resources and is often used in non-real-time operations or “getter” functions. Owner thread execution does not require data locks as the operation is uniquely executed by the component’s ExecutionEngine. This approach is typically used for process-intensive operations or operations that can be concurrently called from several places.

The **OperationCallers** are the operation proxies implemented in the caller’s component. They are tasked with calling or sending an operation from a remote component:

³There are tools integrated into the OROCOS toolchain to expedite the typekit creation and registration.

- calling an operation is a synchronous method, i.e. blocks the current thread execution until the operation returns;
- sending an operation is an asynchronous non-blocking method, i.e. it immediately returns a `SendHandle`, which can be polled to check the current status or to collect the result once the operation concludes.

Calling or sending operations implicates memory allocation to collect the return values, a non-schedule operation that may compromise the real-time requirement. To deal with this dilemma, the deployment tool uses a real-time memory allocator that turns to a fixed amount available.

5.2.3.4 Properties

Properties and **Attributes** are component “member variables” that store information and share the perk of being accessible and writable in run-time. They are commonly used to configure the component’s parameters. Properties can be read or stored from XML files, thus befitting variables of persistent nature (e.g. robot link lengths), while the lifetime of the attributes overlaps with that of the application. Accessing and writing these variables is a real-time procedure but not thread-safe.

5.2.3.5 Deployment

The deployer provided by the OROCOS framework is itself a component with permission to load and configure other components using an orocos script file (“`.ops`” or “`.osd`”) an XML file. However, the deployer can only load components within the same process.

The deployment process starts by importing components, auxiliary libraries, and typekits. The components are then loaded (created) and configured - setting the Activity(ies) parameters, loading, and connecting services and connecting ports. Depending on each component initial state (PreOperational or Stopped), they can be either configured or started.

5.3 Modular Surgical Robots

Taylor et al. (1991) report a robot for orthopedic surgical applications. Being one of the first surgical robots, it was soon realized the scaling complexity of these systems and the need for a flexible, distributed solution that could incrementally integrate new sensors, actuators, boards, etc. A more detailed look at the controller software of the system called “ROBODOC” (Kazanides et al., 1992) introduces the first task partitioned architecture with 5 modules to address servo actuation, motion control, surgery sub-tasks, safety and to monitor real-time. The architecture design complies with the surgical application requirements and enables an incremental and flexible development. There are however, several complex computational requirements involved to guarantee the synchronicity and real-time requirements of these components that run across different processes/computers. This design would eventually evolve to the cisst-SAW Framework (Kazanides et al., 2010) whose features were already discussed in subsection 5.2.2.

Daniele Comparetti et al. (2014); Comparetti et al. (2012a) report a high-level surgical robot architecture for multiple neurosurgery scenarios, a joined effort from the European funded project ACTIVE (FP7-ICT-2009-6-270460). The control scheme involves a complex setup with several frameworks and middlewares including ROS, OROCOS and CORBA to deliver a versatile system capable of neuronavigation, cooperative control, and teleoperation. The system comprises several components from different middlewares that work in a distributed environment to manage the robot action from trajectory planning to servo interpolation passing through force control and telemanipulation. However, only limited information is disclosed about the system's modules and the source code is not available, which makes it difficult to reuse or apply this research.

Muradore et al. (2015); Bonfè et al. (2014) describes the development of a comprehensive cognitive system towards surgical tasks, I-SUR European project (FP7-ICT-2009-6-270396). The technologies developed during this project extend to soft organ model fabrication, surgical planning and robot operation in deformable bodies, intuitive user-interface design and intra-operative event handling. One product of this project is a component-based software architecture motivated by the need to merge sensory data and to maneuver several robotic apparatus including the ISUR robot with a macro unit (3 DoF delta configuration) coupled to a micro unit (4 DoF serial arm) to move and orient the needle; and a 6 DoF serial manipulator that handles the ultrasound probe. The control architecture was implemented in five OROCOS components to take advantage of its real-time properties: 1) motion planner, 2) trajectory generator, 3) variable admittance control, 4) robot driver and a 5) supervisor. Similarly to the previous listed system, there is no in-depth information about the actual modules besides concepts, so any subsequent research that aims to add or improve to the system's components needs to re-implement the whole architecture from scratch.

Sandoval et al. (2018) introduce a framework for robot-assisted minimally invasive surgery based on torque control of a redundant 7-DoF serial robot. The system categorizes as a shared-control platform that requires the user to initially manually move the robot to the incision cut where the trocar is placed. Thenceforth, a Cartesian compliance control moves the robot in accordance with a Remote Center of Motion (RCM) constraint. A 3D vision system synchronized with the surgical workspace feeds information to the Cartesian controller that generates the trajectories performed by the robot's end-effector. A secondary task is embedded in the controller to permit the robot's nullspace motion triggered by a user input force, without affecting the higher priority surgery task. The system's force control approach makes it less viable for autonomous neuronavigation tasks that are fundamentally based on positions in the work frame.

5.4 Design of a Surgical Robot Controller

In the current chapter, we introduce our approach of a modular architecture towards surgical robotic applications. After elaborating on the pros and cons of the modular methodology, and selecting the most appropriate framework to implement our solution, it is necessary to clarify some of the choices made in the design of our control architecture.

The first real design choice relates to the manipulator kinematic structure. From the analysis of the surgical procedure and determination of the robotic gesture, it was concluded that a serial robotic

manipulator would be more flexible and compact, providing a broader workspace for an easier access to the surface or deep surgical targets from normal or eccentric trajectories. The ideal number of DoF was also discussed in section 3.3, and would ideally be comprehended between 4 and 7. Having presented a solution for 6 DoF surgical robot prototype (refer to chapter 4) (Faria et al., 2013, 2016), we decided to develop a 7 DoF prototype for completeness. Other reasons weighted on the choice of a 7 DoF manipulator: i) the existence of a certified robot model for human collaboration in medical applications, the KUKA LBR Med ⁴, and ii) the inherent manipulator redundancy that permits the robot to perform secondary tasks such as, to conform to the operating room space, avoid joint limits and singularities. The capacity to use the extra-DoF to perform these secondary motion objectives is an innovation when compared to the commercial 5 and 6-DoF systems (Table 3.1).

In terms of user-interaction, we define our approach as a Supervisory Controlled system that acts as a passive agent to precisely position surgical instrumentation according to pre-defined trajectories. To demonstrate the developed system's potential we also show how it can be easily adapted to a Telesurgical application. As for feasibility, a neuronavigation system does not have extraordinary requirements. The separation of the preoperative imaging stage from the intraoperative procedure and the limited intersection between the tool positioner and intraoperative imaging systems mean that the robot materials do not need to comply with the CT or MRI scanners. The robotic system is designed so that only the end-effector interacts with the surgical tools. Thus, the end-effector is the only part of the system that needs to be sterilized/able and covering the robot's body with sterile drapes often suffices to avoid biological contamination.

A lot of emphases was put on having a deterministic system with predictable movements. While this proposition has several ramifications, in terms of robot control, special care was taken to avoid reaching singularities or joint limits (semi-singularities) using a novel Inverse Kinematics method and to generate consistent and safe movements. The physical "dead man" switches and emergency stop buttons have been translated and implemented in software. The system receives external signals that, at the moment, are software generated but can easily be adapted to handle hardware triggers. Additionally, the joint brakes are activated whenever the robot is not moving and on top of that, the end-effector is designed to be: sterilizable, permit quick tool exchange, and in case of failure, allow the robotic system retraction to proceed with the standard tools. We opted to use a touch-based registration method similar to the one presented in subsection 4.3.2 as it does not require additional hardware. Other popular registration techniques using fiducial markers or the facial surface, can be later incorporated by taking advantage of the incremental system development model.

The software implementation is actually where the previous arguments on component-based technology and its perks meet some of the requirements for surgical robots. Firstly, by partitioning the control code into functional blocks we create a structure that is easier to configure and validate. Each part of the controller is more clearly contained and has a detailed interface, so it can be individually and thoroughly tested. Special attention was paid to ensure that the framework where we implemented our system was real-time compliant. Any safety-critical robotic system should impose real-time constraints. It also adds to the system

⁴<https://www.kuka.com/en-de/industries/healthcare/kuka-medical-robotics>

predictability and consistency as it guarantees that the system meets absolute time deadlines. The critical components that require real-time run in a single host with an RTOS and communicate intra or interprocess. The HMI and the application layer components can run in the same host using a non-real-time scheduler or can run in another host within the same network.

The developed software components are documented using Doxygen⁵ and the Qt Creator plugin. The code is versioned under SVN⁶ in an online repository. The component's design and structure are well detailed according to the OROCOS framework standard. The system is set up in an Ubuntu 14.04 (kernel version 3.18.20) host patched with a Xenomai 2.6.5 RT-kernel. For a detailed description of the system and included components refer to chapter 6

5.5 Architecture Layout

Before describing in depth each of the developed components, we introduce a representative diagram of the robotic manipulator controller's architecture Figure 5.4. The structure is divided into two main groups: the Core and the Application. The Core group stands for a set of components responsible for executing coordinated task- and joint-space robot movements in a deterministic and reliable way, i.e. each action is uniquely and unequivocally defined and executed only when the entirety of the motion is possible. Unlike the Core group, the Application components are designed with the surgical task in mind. Its components form the bound between the user and the generic core functionality, handling coordinates and motion sequences adjusted to the task.

This high-level controller was designed to keep a minimalist interface with the hardware setup and thus reduce the dependency on the device's available API. In regard to the robot actuation, the interface between the Core components and the physical robot control unit is limited to the communication of joint variables (positions and external forces) at a defined time step.

The components colored in blue (Procedure Coordinator, Task Supervisor and Robot State) represent 'server units', i.e. capable of communicating with components residing in other processes or across the network. The components colored in green are embedded in other applications. The Human Machine Interface component is integrated with the graphical user interface created using Qt, while the V-REP Plugin component is integrated in the plugin created for the V-REP simulator. Both components function as clients that interact with the blue components.

5.6 Setup description

To evaluate the performance of the developed controller architecture, the solution was tested in a robotic manipulator, model KUKA LBR iiwa 7 R800 (c.f. Figure 5.5 and Table 5.1) with different end-effectors that interface the robot flange with the surgical tools.

⁵www.doxygen.org/

⁶<https://subversion.apache.org/>



Figure 5.5: KUKA LBR Med 7 R800 model.

Table 5.1: KUKA LBR iiwa / Med 7 R800 robotic arm specifications

Axes	7
Payload	7 kg
H. Reach	800 mm
Weight	23.9 / 25.5 kg
Repeatability	0.1 mm
Installation Position	Any
Protection Class	IP 54

Another disruptive and inviting feature of the KUKA LBR iiwa robots has to do with their Sunrise.OS programming interface that operates fully on JAVA - arguably the most popular programming language as of date. The advantages of this design choice are twofold: allow the user to make use of many of the language perks, and also the familiarity to work with JAVA instead of a proprietary language, as is common among other robotic manufacturers. The Sunrise.OS ships with the Sunrise.Workbench, the IDE used to install and configure the robot station, safety specifications, IOs; and where the robot applications are programmed, managed and synchronized with the controller. The software bundle provided by KUKA includes a couple of useful packages for fast application development, among them two are particularly relevant for this project: the Human Robot Collaboration and the Connectivity FRI.

Certified by the norm EN ISO 13849-1:2008, the Human Robot Collaboration module enables the user interact with the robot in Performance level d with an architecture category 3, where single faults should be detected and do not cause a loss of the safety function. In practice the system enforces a safe interaction with torque/force, velocity monitoring and collision detection. From this module, a feature called “hand-guidance” was included in our project. This module permits the user to manually move the

entire robot structure, which can be particularly useful since it is the most intuitive and quickest method to conform the robot to the desired pose. The hand-guidance mode imposes virtual constraints to avoid driving any joint beyond their limits and takes into account the dynamics of the robot and any attached tool for gravity compensation.

The Connectivity FRI package provides the FRI (historically Fast Research Interface (Schreiber et al., 2010)) an interface via which the robot controller can communicate continuously and in real-time with an external system, Figure 5.6. It was design for laboratory experimentation with the purpose of developing new applications and algorithms to control the LBR iiwa robot. To operate the robot through FRI three

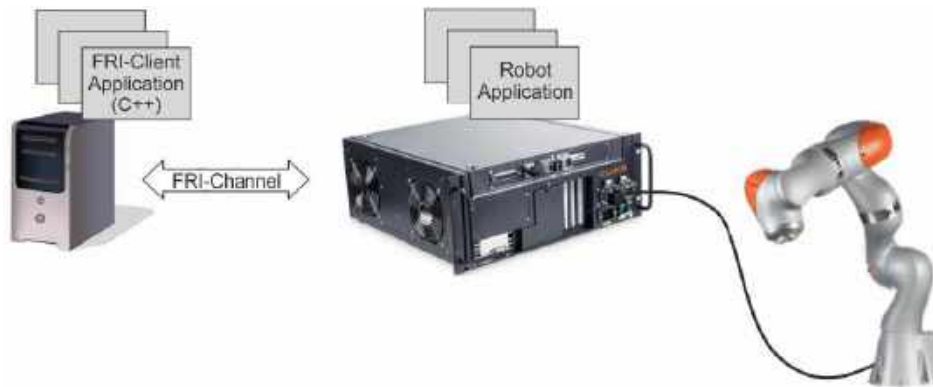


Figure 5.6: KUKA Sunrise.Connectivity FRI overview.

elements are needed: the robot application (server) coded in JAVA and executed directly in the Sunrise Cabinet, the client application coded in C++ and running in a remote host, and a FRI channel. This channel connects the remote host to the cabinet through a physical Ethernet connection using a UDP protocol for best-effort communication of binary data packages. The client of the FRI connection is described in more detail in subsection 6.6.1.

Communication via FRI is established via the KUKA Option Network Interface (KONI) that provides a rea-time capable Ethernet connection with clock rates up to 1ms. Once the connection is configured and established the server requires a continuous and steady communication with the client at the agreed frequency, hence the reason for the real-time requirements of the control architecture components. A mismatching latency, communication jitter or dropped communication messages deteriorates the connection quality, which is usually labeled as POOR, FAIR, GOOD or EXCELLENT. Connection state affects directly the session state, a parameter that defines if the FRI connection is in a monitoring state (MONITOR_WAIT or MONITOR_READY) or in a commanding state (COMMAND_WAIT or COMMAND_ACTIVE), Figure 5.7.

In a monitoring state, only server-to-client uni-directional communication is supported. While the FRI connection defaults to a monitoring state (independent from the connection quality), it can only transition to a commanding state if the quality rises to GOOD or EXCELLENT. In any session state, the client receives information about the connection state and quality; axis-specific coordinates such as, actual position, torque, setpoint position and torque, external torque; time stamps and other operation related variables. If the connection is at COMMAND_ACTIVE the client can then send new target joint positions for the robot to follow. Note that, the controller internally does the best effort to reach the target coordinates from the

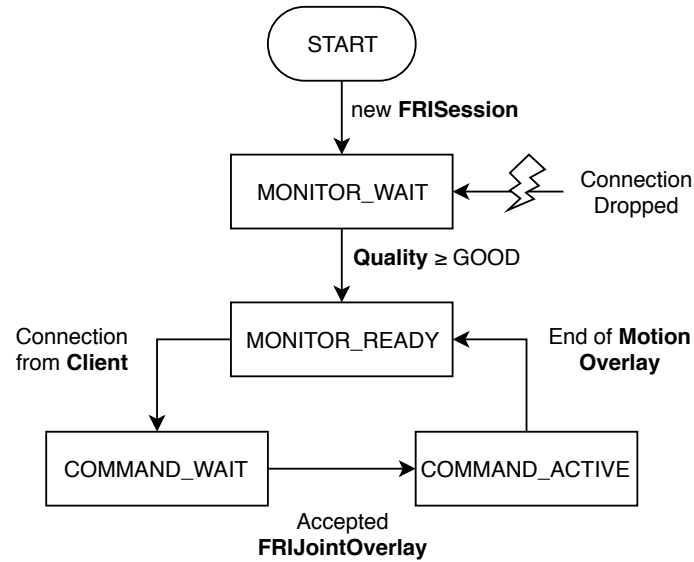


Figure 5.7: Connectivity.FRI session state diagram.

current position within a time-step disregarding the torques or velocities generated. Despite the velocity and acceleration limits imposed for the joint trajectory interpolation 6.3, a software limiter was implemented at the lowest level (immediately before sending the coordinates to the robot controller) to guarantee that the commanded joint position displacement always falls within a safe threshold.

Project Management Although the high-level control of the robotic system is handled by the control architecture, our solution still depends on the robotic controller⁷ to drive the joints and provide feedback about the system state. A robot application was developed to combine the remote real-time control through FRI connection with the certified hand-guidance mode to re-position the arm. The program was implemented as a state-machine with three basic states: Initial or Stopped state, Hand-guidance state and the Remote control state. The transition between states is event-driven and triggered either from the designed end-effector (refer to subsection 5.6.2) or directly from the pendant console. Transition to any of the actuation states (Hand-guidance or Remote) involves a continuous user-input, or a two-step confirmation. If any exception is caught or the communication lost, the robot defaults to the Stopped state.

The FRI Session parameters are configured during the initialization of the robotic application. During this step, the local and remote IP and port addresses are specified along with the time-step and the receiver multiplier⁸. The communication with the remote host is set to an update cycle of 5 ms with a multiplier factor of 1.

5.6.2 Other equipment: Micro-drive and Designed tools

To guarantee a correct fit between the surgical tools (e.g. the micro-drive and the drill fittings) and the robotic system, a custom end-effector was designed and created. The Leibinger micro-drive for stereotactic

⁷the Sunrise Cabinet in this case

⁸Factor that defines the rate at which the remote host should communicate with the controller, $\text{actual_rate} = \text{send_rate} \times \text{factor}$.

surgery was the surgical equipment selected to hold and place the electrodes, Figure 5.8. It has a rotating screw to control the advance of the electrodes with sub-millimetric accuracy.

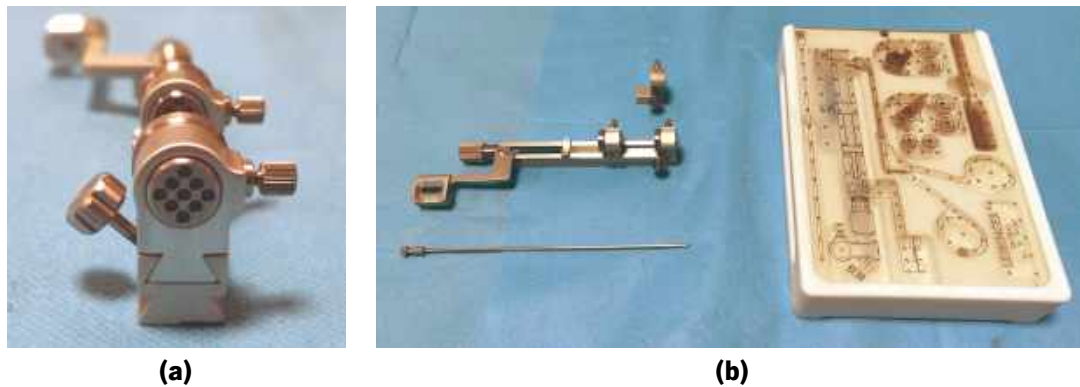


Figure 5.8: Leibinger electrode micro-drive.

We designed an end-effector tool to handle the micro-drive. The following requirements were considered:

- a chute to attach the micro-drive;
- a base that fits the KUKA LBR iiwa 7 R800 flange;
- unimpeded access to the micro-drive;
- dimensioned to keep the robot structure from being too close to the patient;
- configurable buttons to directly interact with the controller;
- a touch registration system.

The end-effector prototype was designed in a CAD software and its parts fabricated separately, Figure 5.9. The touch registration system is based on DME Round Tapered Interlocks (Milacron, Michigan,

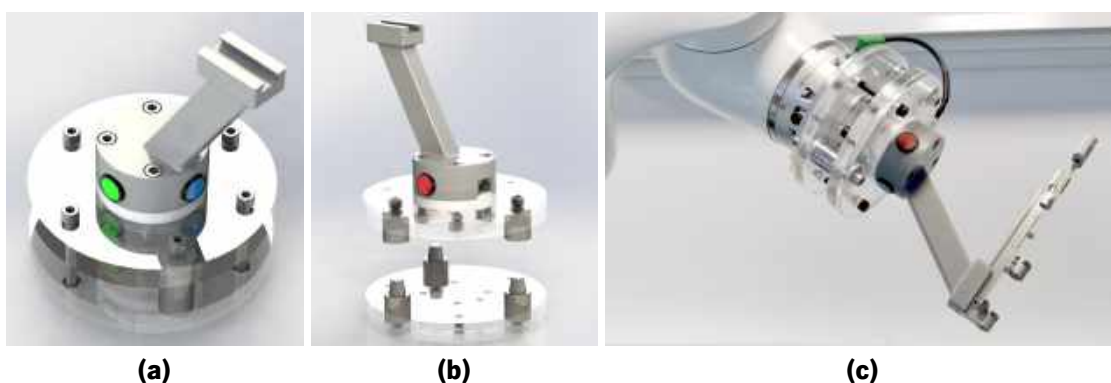


Figure 5.9: Designed end-effector for stereotactic surgery.

United States). The male part of the interlock system is fixed on an acrylic disk and attached to the robot flange. The female part is also attached to an acrylic disk and serves as a base for the surgical end-effector tools. The interlock interface permits the attachment on-the-fly of new end-effectors with guaranteed position and orientation alignment.

The end-effector part that holds the micro-drive includes 3 push-buttons. The red button halts any motion of the robot and switches the robot the Stopped state. The green button permits a switch to the Hand-guidance state where the user is able to manually drive the robot to the desired posture. In Hand-guidance state, the controller is configured to continuously monitor the velocity of each joint and the external forces exerted, as well as the position of each joint. If the applied velocity or external force overcomes the stipulated safety threshold, the robot switches to a safe state, and requires a new user-safety trigger to resume operation. Simultaneously, the controller applies a virtual constraint to each joint to avoid overshooting the joint position while manually driving the robot. Finally, the blue button can be pressed to notify the robot controller to switch to a Remote operation state. In this state, the robot can be driven with the implemented control architecture. We have designed a custom PCB as a button-controller interface to deal with the switch debounce, Figure 5.10.



Figure 5.10: 4-button debounce PCB.

5.7 Simulation Setup

To deliver a platform that facilitates the transition between concept to a final robotic application guided towards surgical applications, simulation stands as a fundamental tool. Perhaps the best evidence to support the claim to the usefulness of robotic simulators might be the quantity and quality of available solutions (Ivaldi et al., 2014).

The robotic simulator is first and foremost the tool to easily, yet closely replicate the robotic system as well as the conditions it will operate. The primary choice criteria were thus, the capacity to realistically emulate the real-system, the software performance and the completeness/extensibility of their API. As for any other software considered, the documentation and community support are also very important factors.

Amongst the full list of available robotic simulators, and keep in mind the novelty and fast-paced changing environment of these technologies, there are three major competitors when considering features and performance in the current scene (Pitonakova et al., 2018):

1. V-REP, a powerful general purpose simulator with an extensive variety of APIs / Programming approaches (Rohmer et al., 2013);
2. Gazebo, one of the first and most successful robotic simulators (Koenig and Howard, 2004);
3. ARGoS, a primarily multi-robot simulator focused on fast performance experiments involving swarm formations. (Pincioli et al., 2012).

In the end, we chose to work with the V-REP robotics simulator, as it is the software with the most complete API/features, documentation and community support, and just overall intuitiveness and ease of use.

5.7.1 Virtual Equipment Models

One of the advantages of V-REP is the facility to add customized graphical models and integrate their physical counterparts. Simulating the environment where the robot operates entails the design of a replica of an operating room, particularly, from the Neurosurgery Service of Coimbra University Hospitals. This was possible by integrating new and previously designed 3D models of medical equipment (Faria et al., 2013).

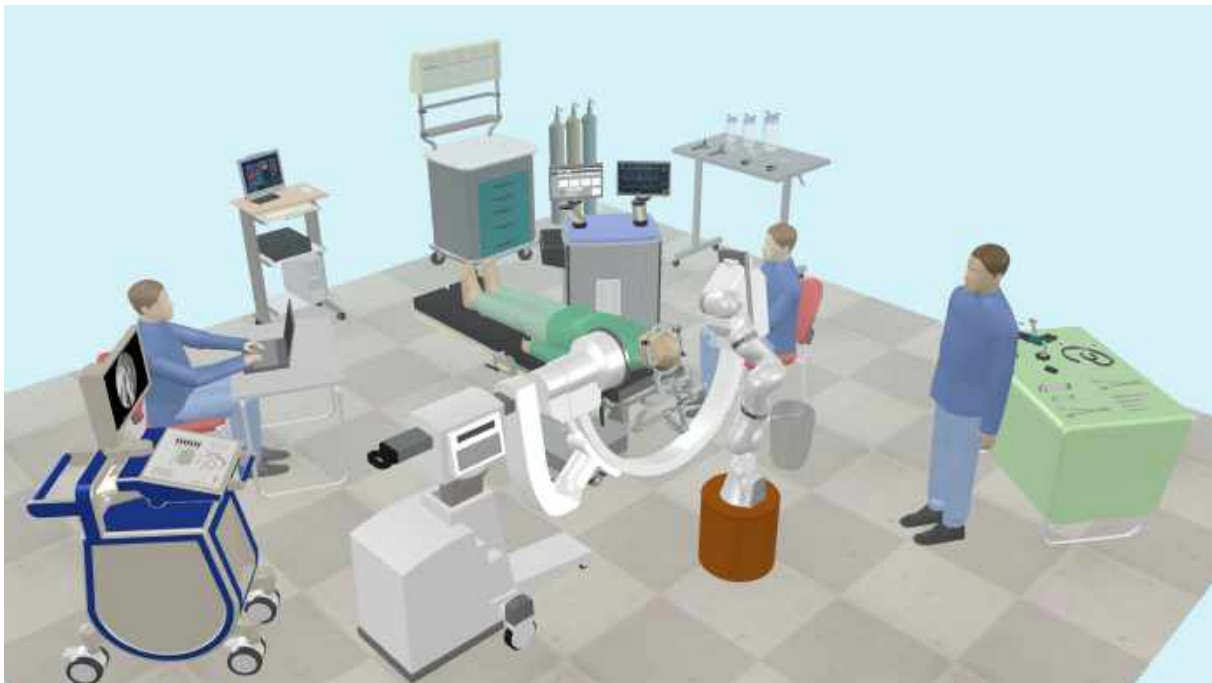


Figure 5.11: Operating room virtual scenario created in V-REP.

Each personalized model was designed in CAD software and one or more *.stl or *.obj files generated. These files are imported as triangular meshes into a shape. To differentiate between the graphical and

physical representation, each model contains at least 2 sets of shapes, the imported triangular meshes and the physical representation shapes, Figure 5.12.

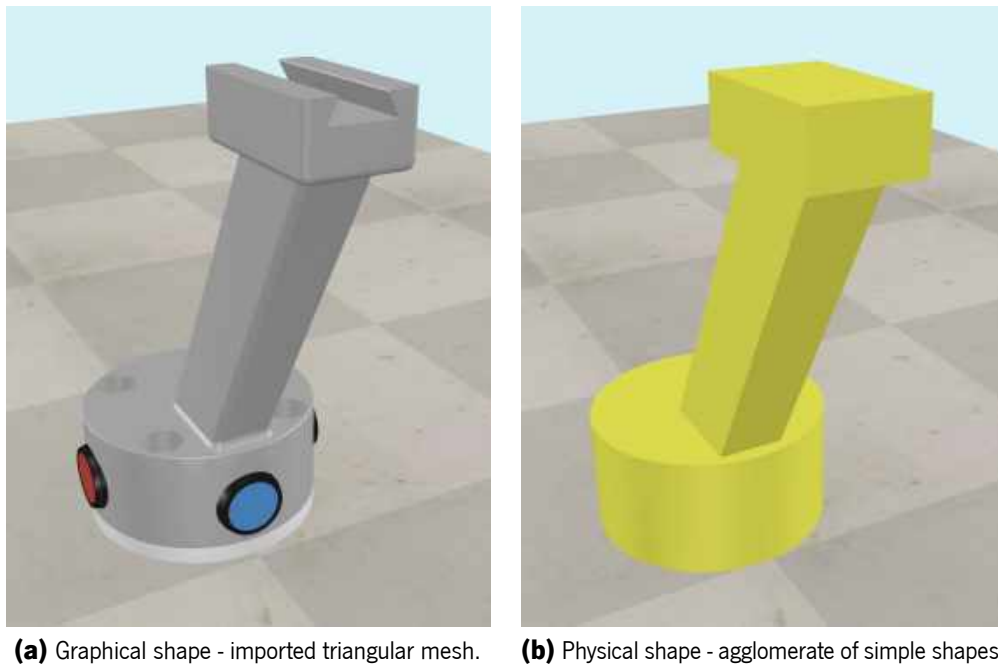


Figure 5.12: Virtual representation of the developed end-effector.

The physical shapes are usually allocated in a hidden layer and not drawn by the renderer (OpenGL). These shapes are used by the physical engine to process the interaction between simulated entities. The physical shapes of the integrated models were designed using a composition of simple shapes (cuboids, cylinders, spheres as seen in Figure 5.12b) or by generating a convex hull of the graphical shape. Additionally, to each physical shape, we assigned a mass, the inertial frame and the center of mass. This information is crucial, for instance, to dynamically simulate the robotic manipulator behavior.

V-REP has an extensive library of robots and models including the KUKA LBR iiwa R800 and R820. In these particular models, however, there is a mismatch between the shape of the virtual and the real flange, with the former being 15 mm shorter length-wise. We took this opportunity to introduce a new robot model with the correct flange shape as well as with corrected flange and link positions/orientations (see subsection 7.1), Figure 5.13.

To conclude this subsection, we will briefly refer to two new added features, the hollow robot and the inverse kinematics runtime functionality.

The hollow robot is no more than an instance of the same virtually controlled robot model but with no physical properties and with a semi-opaque graphical shape. This hollow representation fulfills three roles:

1. to provide instantaneous feedback on the robot posture when adjusting the robot relative to its redundant degrees of freedom;
2. to execute the exact same motions required of the real robot to provide a pre-visualization mechanic;
3. to provide instantaneous feedback on the final robot pose at the end of a movement.



Figure 5.13: KUKA LBR Med model added to V-REP.

These features add value to the simulation that can be applied as an alternative or complement to operating with the real robot. When operating with the real robot, one can use the simulation scene to pre-visualize the robot movements before forwarding the command to the robot controller.

As for the inverse kinematics runtime feature, we take advantage of the V-REP inverse kinematics module to provide the user with an interface to manually move the virtual robot posture in runtime using the object manipulation controls available in V-REP. This feature aims to emulate the 'hands-on' control implemented in the real KUKA controller.

5.7.2 Developed Plugin

As-is, V-REP allows the developer to interact with the simulation using six possible approaches: i) Remote API Clients, ii) Plugins, iii) Embedded Scripts, iv) Add-ons, v) ROS nodes, and vi) BlueZero nodes. Since we are working with the OROCOS framework and to avoid adding to the system's software dependencies, we ruled out the communication through ROS or BlueZero nodes. Out of the other possibilities, our solution had to guarantee the following requirements:

- customized client/server communication with no lag;
- access to the Regular API (most complete interface);
- fast code execution speed (compiled languages);

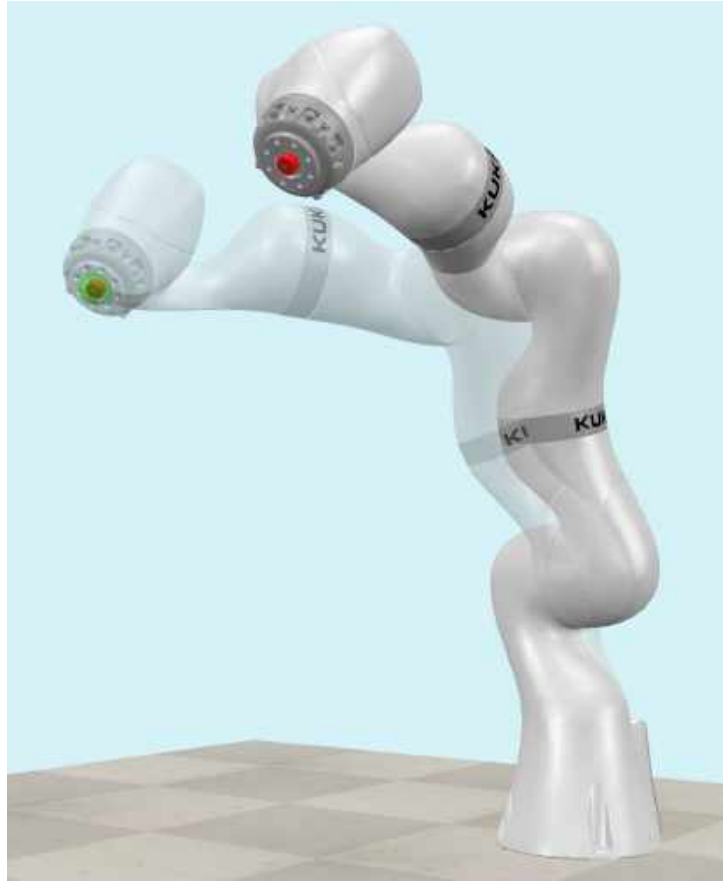


Figure 5.14: Image of the real and the hollow robot with a coincident base frame and with different poses.

With this conditions we aim to replicate as close as possible the conditions we have when interacting with the real robot, thus having a more reliable simulation. According to the requirements previously defined, we chose to develop a C++ plugin for V-REP with an integrated interface for OROCOS.

Plugins are extensions to the main application that are loaded at the program start-up. A plugin has no communication lag because they are handled directly by the main application execution thread. It is relevant to point out that the plugin only supports synchronous operation, i.e. the main application handles the plugin calls at each simulation cycle. At the same time, the plugin is responsible for the synchronous/asynchronous communication with the control architecture, and thus, a thread-safe a mediating mechanism was implemented based on concurrent queues.

The plugin hosts three OROCOS components that communicate directly with the control architecture through the standard communication routes (attributes, ports, operations), refer to subsection 5.2.3. The connection is handled by CORBA which means that the the control system and the simulation can seamlessly communicate when running in the same computer or in computers in the same network. These components fulfill three tasks, robot joint control, tool changing and control, and surgery trajectory management. In fact, the V-REP Plugin control in Figure 5.4, actually expands to the components in Figure 5.15.

Since these components are expected to be launched internally as a simulation starts, a deployer class was setup to properly launch and connect the plugin components to the control architecture. In each

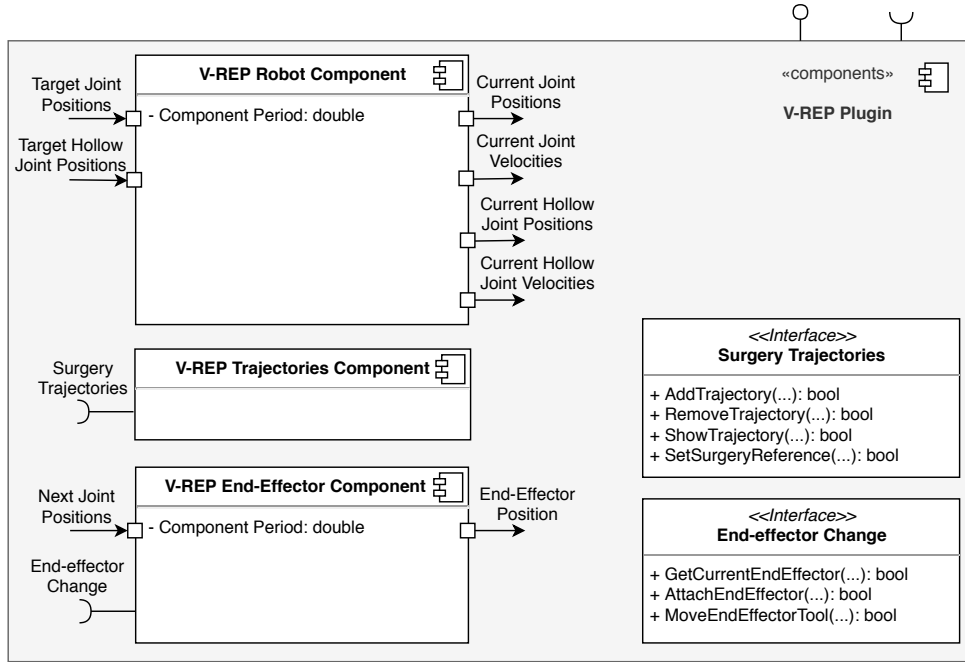


Figure 5.15: OROCOS components embedded in the developed V-REP plugin.

simulation cycle, the mediating concurrent queues that store the information from the control architecture are inspected, and the corresponding V-REP Regular API functions called from the main application running thread⁹.

The V-REP Robot component is responsible for exchanging information about the robot joint coordinates between the control architecture and the virtual robots. These messages are exchanged at a steady frequency, defined by the Component Period parameter, Figure 5.15. Similar to the case with the real KUKA robot, the information exchanged between the control and the machine is limited to joint coordinate data. The V-REP Trajectories component allows the user to create visual cues related to the surgical trajectories. These trajectories are pure-graphical and semi-opaque shapes generated from the target and entry coordinates relative to the surgery reference frame, Figure 5.16. Finally, several end-effector models were added to the scene, some can be actuated along their working axis, Figure 5.17 and 5.18. The V-REP End-Effector Component allows the user to switch between different end-effectors, as well as to control their tool movement (if actuated).

5.8 Task Description

As presented in chapter 2, stereotactic neurosurgery is inherently a strict, well-defined procedure in terms of surgical target positions and how to approach them. Objectively, the stereotactic surgical can be summarized to the guidance of tools along linear trajectories. This problem was previously discussed (see subsection 4.4) for the use of an industrial manipulator. The role of the robotic system is similar to the previously presented industrial robot adaptation. However, since the robot is for the most part controlled from the

⁹Calling Regular API functions from the communication threads results in faulty behavior.

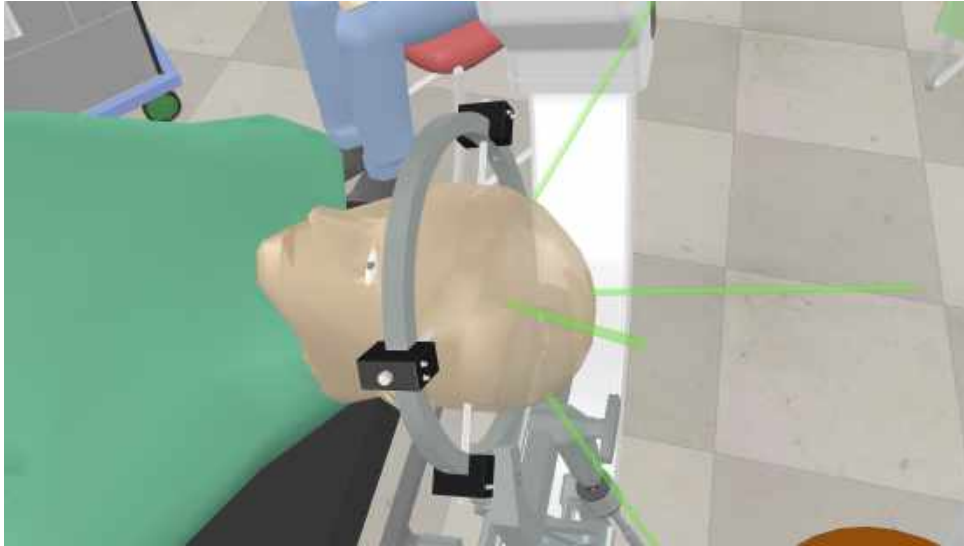


Figure 5.16: Virtual representation of the surgical trajectories in the V-REP scenario.

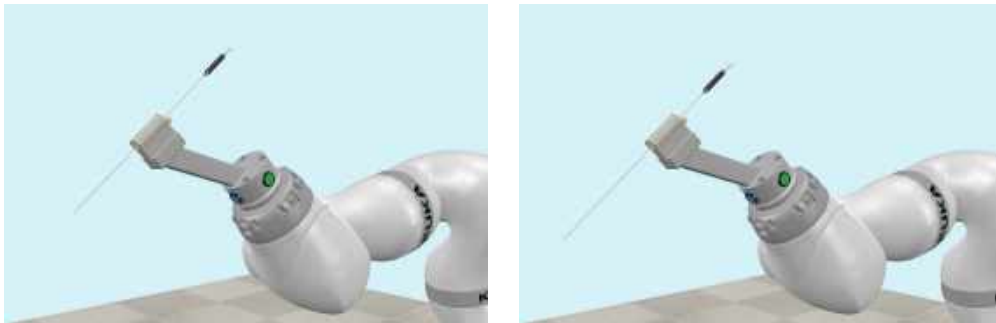


Figure 5.17: Concept of probe guide end-effector with actuated depth control.

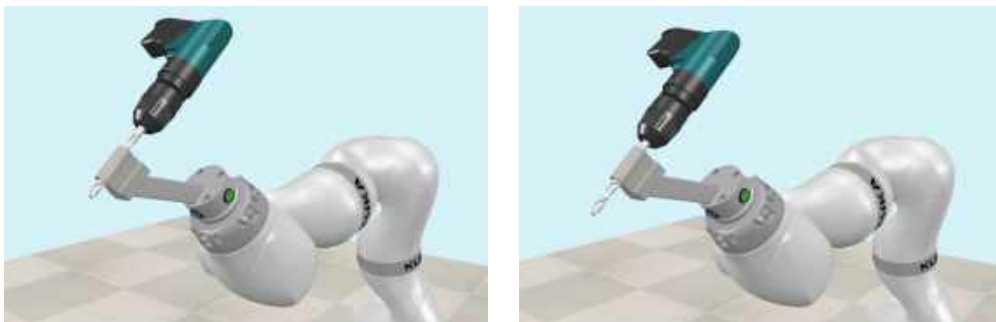


Figure 5.18: Concept of trepan guide end-effector with actuated depth control.

modular architecture, one has more thorough control over each action, the capacity to finely tune each action to the task requirements, and the ability to react in real-time and within a single control cycle to any event. Moreover, the flexible structure of the control architecture permits a seamless addition or replacement of components or processes and facilitates porting the solution to different actuators.

The robotic manipulator assistance in a stereotactic surgery can be broke down to two steps: i) the **Approach** to the selected trajectory, and the ii) **Tool Handling** - linear movements along the entry-target trajectory, Figure 5.19.

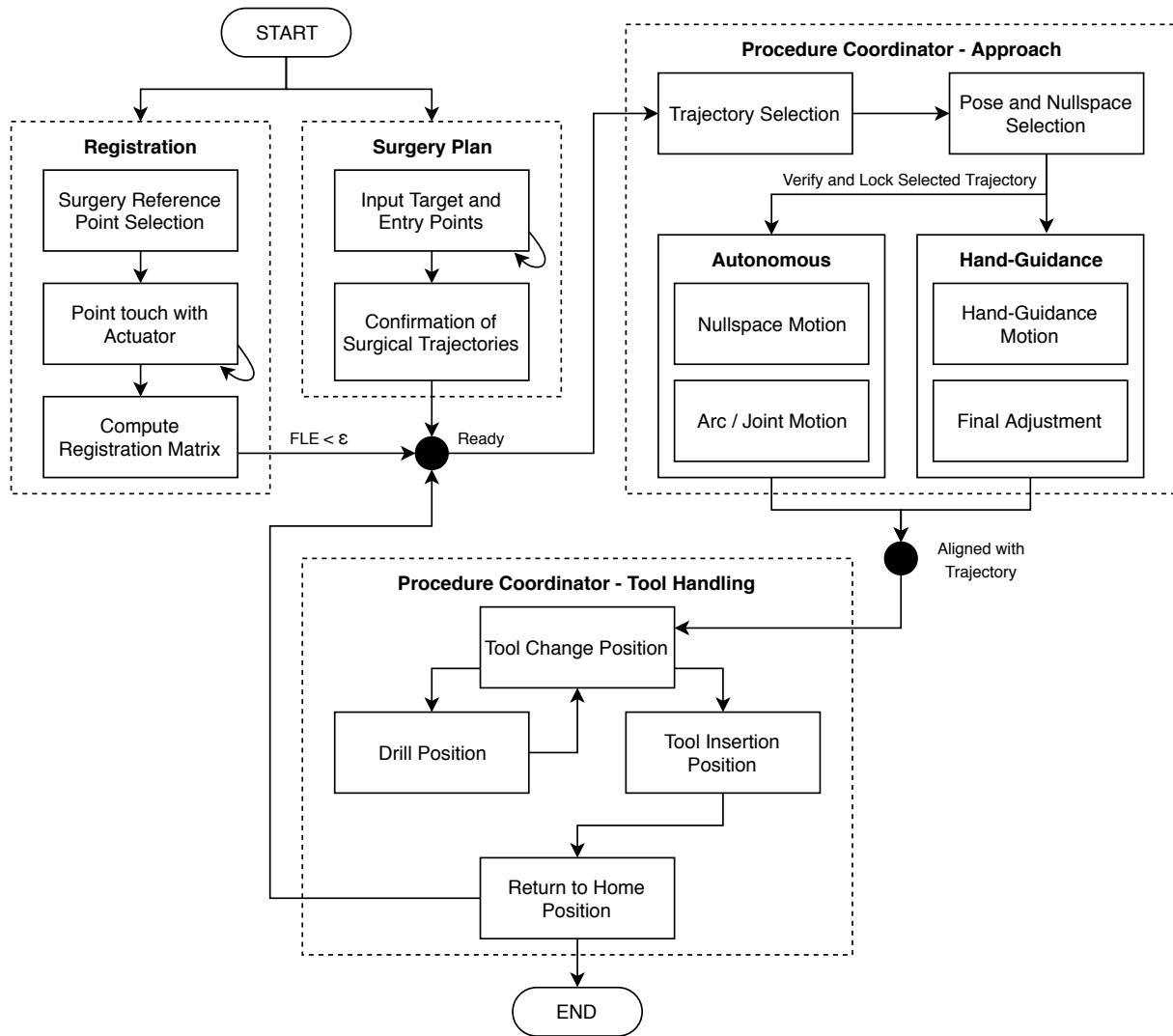


Figure 5.19: Diagram of Stereotactic Neurosurgery assistance tasks performed by the robotic system.

The user may choose to perform the **Registration** or input the **Surgery Plan** first, but both steps are required prior to initiating the surgical procedure. If the system operator chooses to perform the intra-operative registration first, then an interface is prompt to introduce the coordinates of the known registration points relative to the surgery reference frame. Once concluded, the robot changes to a hand-guidance mode and the end-effector is manually driven to touch the -registration points, all-at-once or each one sequentially. The two sets of registration points, relative to the surgery reference frame and the robot base frame, are processed by an algorithm that outputs the rigid transformation that relates both references. The transformation is accepted if the Fiducial Localization Error (FLE) is below an established error threshold (ϵ).

Since the current system has no integration with the surgery planning software, the user is required to manually introduce the surgery entry and target coordinates (relative to the surgery reference frame) in the system. When both steps are concluded the system shifts to the **Procedure Coordinator** mode.

For each surgical trajectory, the robot operation starts with the **Approach** step. When a trajectory is selected, the system performs a feasibility analysis to select the most appropriate pose and nullspace

configuration to approach the trajectory, guaranteeing the motion feasibility also expected for the tool handling step. The best fitting robot poses and configurations are reported to the user who can select the solution that better suits the workspace environment (pre-visualization in a virtual scenario). Once the robot pose and nullspace solutions are chosen for the selected trajectory, the robot attempts to move from its home position to the respective approach trajectory pose. The system permits two different interactions, autonomous or hand-guided motion. In autonomous mode, the robot first executes a nullspace motion to the selected nullspace solution and then moves to the final pose. Alternatively, the robotic system user may manually drive the manipulator to a pose proximal to the selected trajectory in task space, keeping an appropriate configuration. The robot then performs a small adjustment to exactly fit the end-effector to the linear trajectory defined by the entry and target points.

With the robot-end effector aligned with the surgical trajectory, the **Tool Handling** step begins. Since the surgical trajectory is described by a line in space, the relevant robot poses in space are completely described by the distance of the end-effector to a known point along the trajectory, the entry or target point, Figure 5.20. The approach point refers to the final position of the approach movement, i.e. the point in

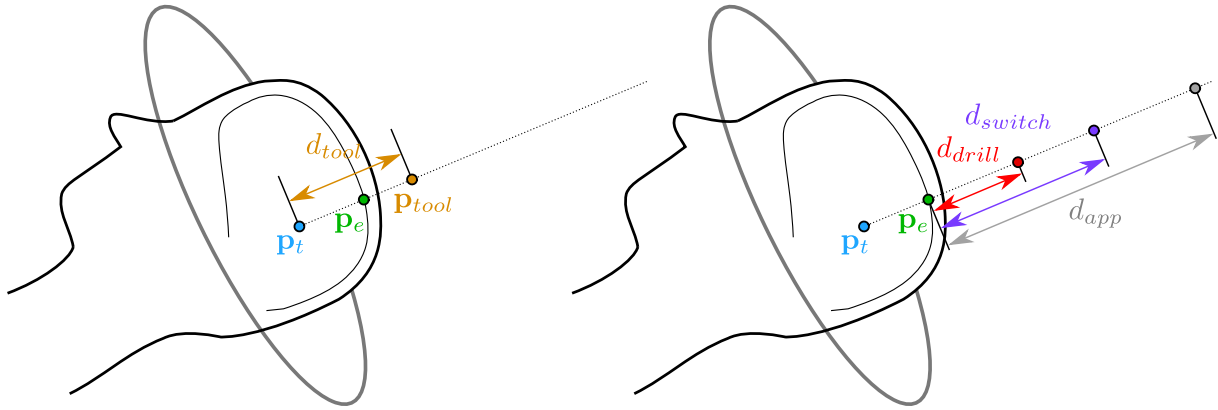


Figure 5.20: Relevant procedure points defined along the entry-target trajectory. In the target-entry direction and from the target point one finds the tool point at a distance of d_{tool} . In the target-entry direction and from the entry point one finds the approach point at a distance of d_{app} , the tool-switching point at a distance of d_{tool} and the skull drill point at a distance of d_{drill} .

task space where the end-effector will first align with the entry-target trajectory. This point is distanced of 200 mm from the entry point. The tool-switching point marks the position to where the robot retracts so the user may switch between surgical tools. This point is distanced from the entry point of 100 mm. The drill point refers to the point where the robot should hold its end-effector to passively assist the neurosurgeon during the scalp incision and skull drilling step. The position of this point is measured relative to the entry point and the distanced is affected by the length of the drill bit, in order to stop its advance at the interior wall of the skull. Finally, the tool point marks the position where the robot end-effector will be placed in order to allow the surgical tool holder to drive the probe, electrode, etc. to its final position. The position of the tool point is determined relative to the target point and depends on the length of the surgical tool used.

When the procedure for a single trajectory is concluded, the user is prompt to return the robotic arm to its home position. The robot operator may choose to let the manipulator move autonomously to its home position. If so, the robot executes a two-step motion, first a linear motion until reaching the approach

position and then the final arc/joint motion. Alternatively, the operator may manually drive the robot away from the surgical trajectory using the hand-guidance module. Before starting a new surgical trajectory procedure, the robot always starts from its home position. For a better understanding of the procedure workflow and user-interaction refer to the video at <https://youtu.be/011E5Cr4s1s>.

Chapter 6

Developed Components

In this chapter we list the different components that form the high-level control architecture. As previously mentioned, these components are split into two layers, the Core Layer that includes:

- Cartesian Trajectory Controller;
- Kinematics and Statics Module;
- Joint Trajectory Controller;
- Robot State and Transformations;
- Task Supervisor;
- Fast Research Interface.

The second layer - Application Layer - includes:

- Registration;
- Surgery Plan;
- Procedure Coordinator.

Finally, the Graphical User Interface is also described. Despite being an application, it includes OROCOS components that act as proxies to communicate directly and transparently with the high-level control architecture.

6.1 Cartesian Trajectory Controller

The Cartesian Trajectory Controller is the component assigned with generating timed trajectories according to the desired robot's end-effector motion in task space. It is based on the open-source Orocos Kinematics and Dynamics C++ Library - KDL. The component provides an interface to execute Cartesian space trajectories such as: i) linear, ii) arc, iii) composed, iv) nullspace motions if the robotic manipulator is intrinsically redundant or v) a sequence of the previous. The concept of trajectories should be understood as a combination of two concepts, the geometric path drawn in tri-dimensional space and the implicated velocity profile.

The component provides an interface to generate robot trajectories in task space. It outputs a vector with frames, generated at a fixed time sampling rate and the time-stamp of each frame the end-effector should pass through, Figure 6.1. The component's interface is mainly composed of operations that can be called to request a specific type of trajectory with the associated parameters. Once the trajectory frames are computed, the result is dispatched to the Kinematics and Dynamics Module. Thus, this component does not operate with real-time requirements.

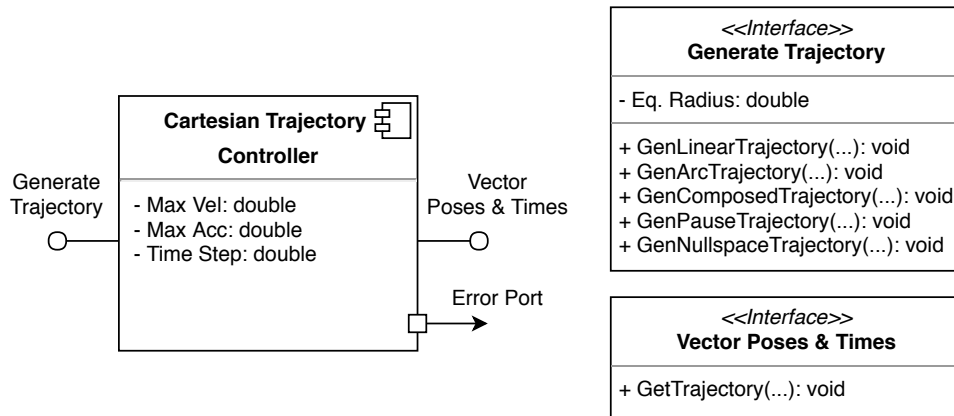


Figure 6.1: Cartesian Trajectory Controller component and interfaces.

6.1.1 Geometric Path

The required end-effector geometric path is either defined by a function describing its shape in space (point, linear, arc or nullspace motion) or by a set of 'way-points' (composed motion). In the end, both functions and way-points create a path in space that defines the end-effector position and orientation using poses. For clarity, a pose is henceforth referred to as a transformation frame, or simply frame.

6.1.1.1 Linear Path

The interface to create a linear path comprises a start and an end frame related to the same reference system, along with an object to mediate the rotation interpolation. If the linear path is a pure translation - the start and end frames share the same rotation sub-matrix - then it can be described by a vector in space relating the start and end frame positions, Figure 6.2a. On the other hand, if the linear path involves also a rotation, an interpolation algorithm is applied to rotate from the initial frame rotation to the final rotation. The algorithm takes both rotation matrices and determines the resulting axis-angle rotation, and then linearly interpolates the angle around the determined axis, Figure 6.2b.

6.1.1.2 Arc Path

Two alternative interfaces are provided to generate arc or circular paths in Cartesian space. The first option requires a start frame, an arc center position in the same reference system as the start frame, an arc direction vector that specifies the plane where the arc is inscribed, an arc angle scalar and an end rotation

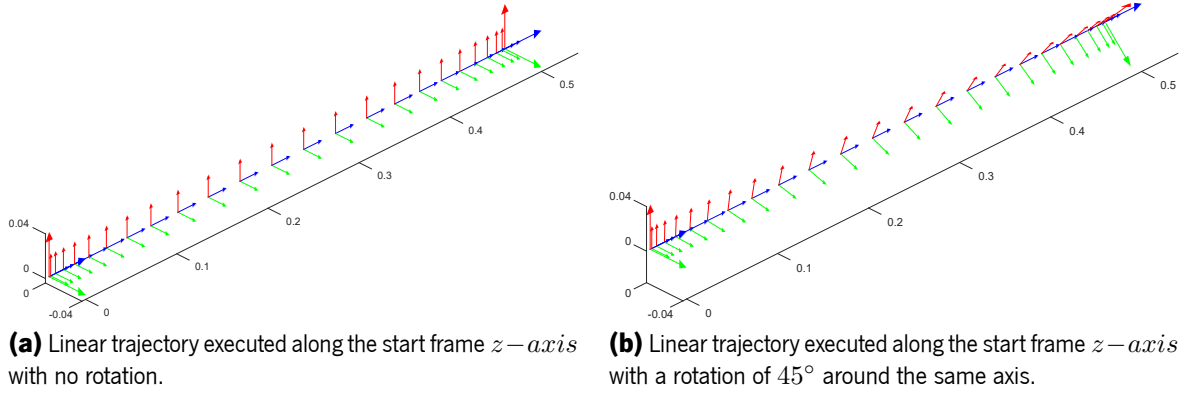


Figure 6.2: The required start and end frames are represented with larger coordinate frames. The smaller coordinate frames represent the intermediate frames which the end-effector should pass through.

matrix. The plane where the arc is inscribed ($\hat{\mathbf{v}}_{arc}$), is defined by the external product of the arc direction vector ($\hat{\mathbf{v}}_{dir}$) and the unit vector ($\hat{\mathbf{v}}_{cen}$) computed from the start frame position to the arc center position (Figure 6.3),

$$\hat{\mathbf{v}}_{arc} = \hat{\mathbf{v}}_{dir} \times \hat{\mathbf{v}}_{cen} \quad (6.1)$$

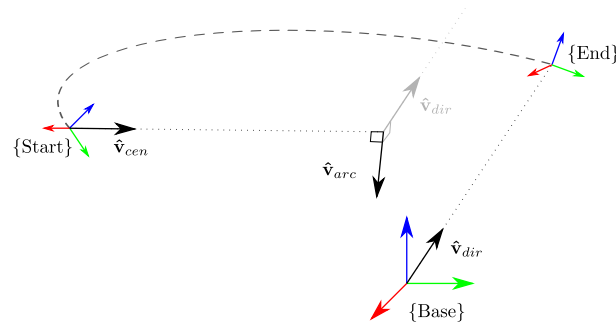


Figure 6.3: Example of arc generated with the variables specified in equation (6.1).

This option provides complete and explicit control over the arc geometry in Cartesian space.

The second interface requires the start and end frames, and a point non-colinear with the line that crosses both frame positions ($\mathbf{p}_{ref}$), the reference point, Figure 6.4. The position of the start, end-frames and the reference point define the plane where the arc is inscribed ($\hat{\mathbf{v}}_{arc}$). The center point of the arc is found along the line that bisects the segment from the start to the end frame positions, at a distance of the of the segment length from both points.

Similarly to the linear path, the rotation interpolation is applied if the start frame's rotation sub-matrix differs from the end rotation matrix, Figure 6.5.

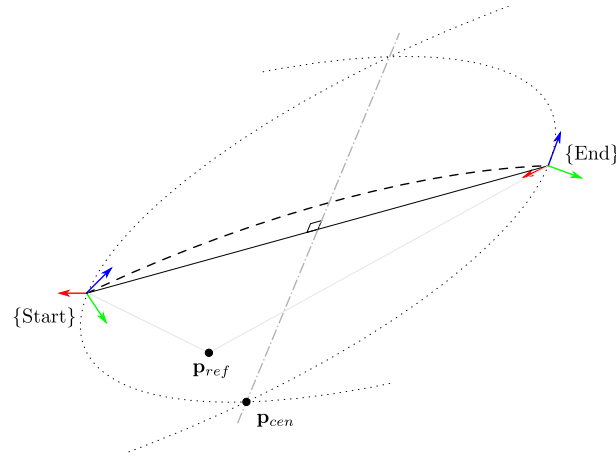


Figure 6.4: Example of arc generated with the start, end frame and the reference point.

6.1.1.3 Composed Path

The composite segment is a path generated from a set of ordered way-points. The component interpolates between the provided way-points following two different approaches. Depending on the motion required, one might specify: i) mandatory way-points, i.e. the generated path links linearly and inclusively the specified input frames (Figure 6.6a); ii) proximal way-points, i.e. the generated path makes an over-fly near a way-point to avoid sharp turns (Figure 6.6b). A path involving proximal way-points requires the user to provide an additional variable that codes the radius of the arc created to transition between line segments.

Lastly, there is a concept intrinsic to any of the path modalities previously presented that is fundamental to the determination of the velocity profile, the path length. The path length associated with translations is no more than the sum of the Euclidean distances between each segment start and end positions. While the concept might be straightforward when the motion is reduced to a pure translation, there is not a direct correlation between path length and the amplitude of a rotation or a nullspace motion. In order to maintain a coherent and safe time-frame for the generated path, a metric is used to quantify these non-translation movements into a quantifiable path length.

To account for the rotation, a concept named ‘equivalent radius’ is introduced. This variable is a parameter to the path function interfaces and establishes the radius of a hypothetical arc formed by the equivalent rotation (axis-angle notation). With this heuristic, it is possible to convert an angle into a representative path length, i.e. the arc distance. When the requested path transformation involves both translation and rotation, the final path length is equal to the greatest scalar between the Euclidean distance of the translation and the rotation equivalent arc distance. Two factors code the scale of both translation and rotation path lengths, in order for them to complete their paths synchronously. Thus, the largest path length part has a scale factor of 1, while the smallest counterpart has a scale that matches the ratio between path lengths.

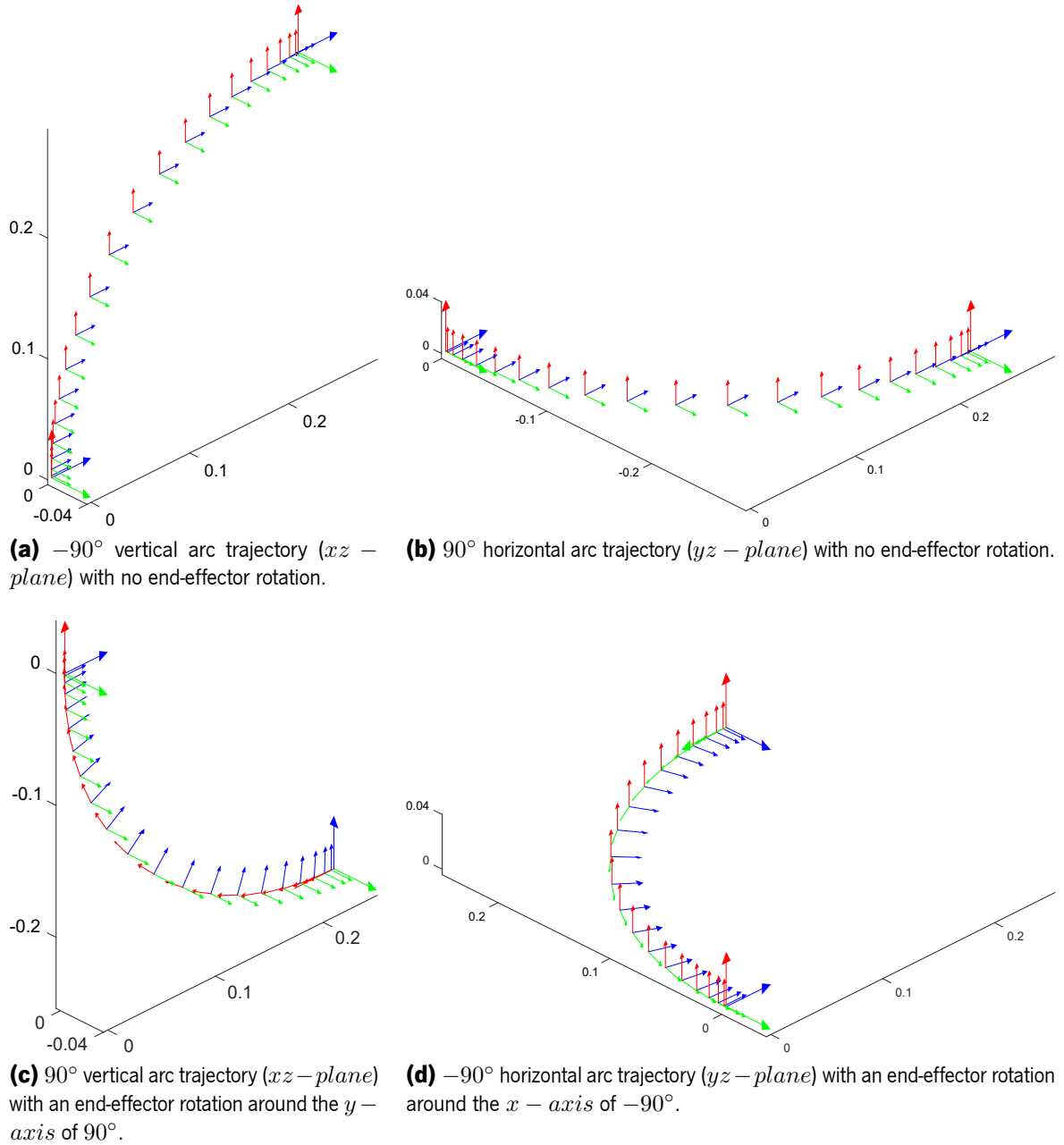


Figure 6.5: Arc trajectory examples in orthogonal planes for easier visualization. Both the arc radius, angle, plane and end-effector orientation can be finely tuned.

6.1.1.4 Nullspace Path

We call ‘nullspace motion’ to the joint motion of the robotic manipulator without altering its end-effector frame in relation to its base reference system, a property shared by kinematically redundant manipulators. Although at first it makes little sense to introduce the concept of nullspace as a part of the Cartesian Trajectory Controller component, in practice this possibility to adjust the robot posture without moving its end-effector will prove useful as a mechanism to handle secondary objectives¹. The nullspace motion is parametrized by a single coordinate (ψ) and a single end-effector pose that is maintained throughout the

¹We will further elaborate on the topic of ‘nullspace’ and its applications on the section 6.2.

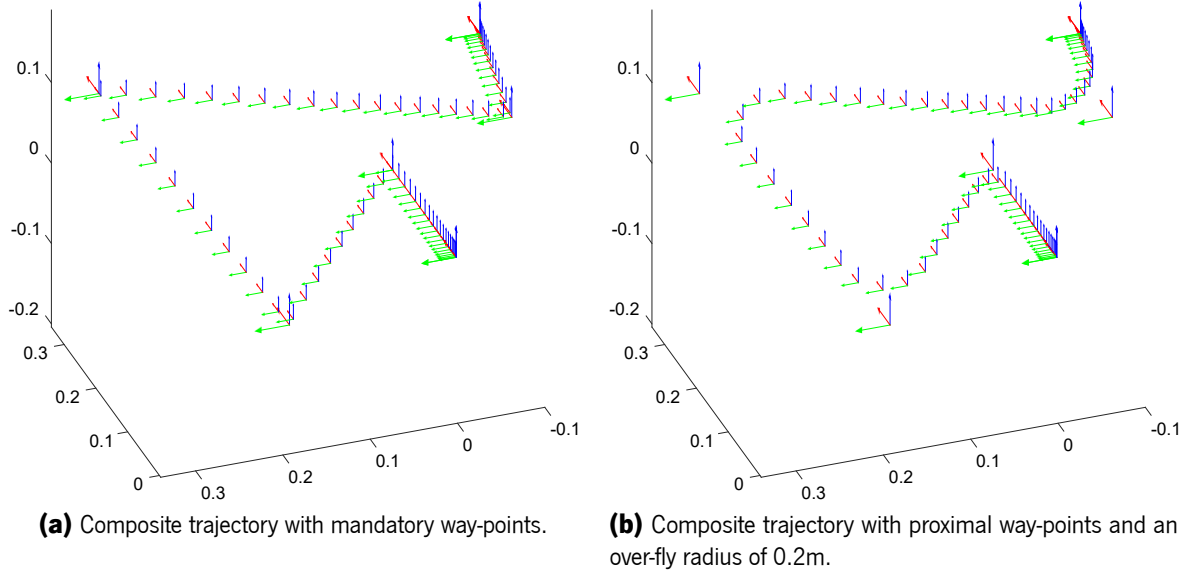


Figure 6.6: Example of a composite geometric path with way-points marked as larger frames.

trajectory segment.

6.1.2 Velocity Profile

A velocity profile defines how the end-effector moves through the geometric path (s). To guarantee a continuous velocity profile, the trapezoidal velocity profile is implemented. Here we define the maximum allowed velocity (v_{max}) and acceleration (a_{max}) for each path segment. Knowing the path segment length (L), one can extrapolate the time required to accelerate to (T_a) / decelerate (T_d) from maximum speed as well as the total time required to perform the whole trajectory (T). A similar strategy is applied to handle nullspace trajectories, where the geometric path (s) is mapped to the nullspace variable (ψ). A maximum ψ velocity and acceleration are also specified.

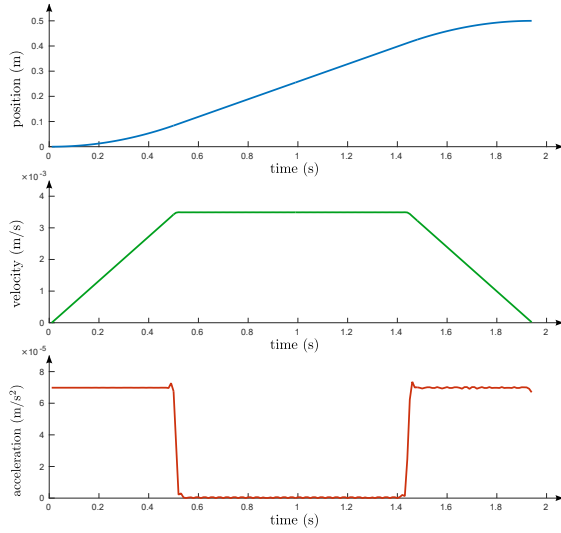
Since the path can be composed of multiple segments, different velocity profiles are applied to match the velocity requirements, Figure 6.7. If the path is composed of a single segment with null start and end velocity, it accelerates and decelerates during,

$$T_{a/d} = \frac{v_f - v_i}{a_{max}} \quad (6.2)$$

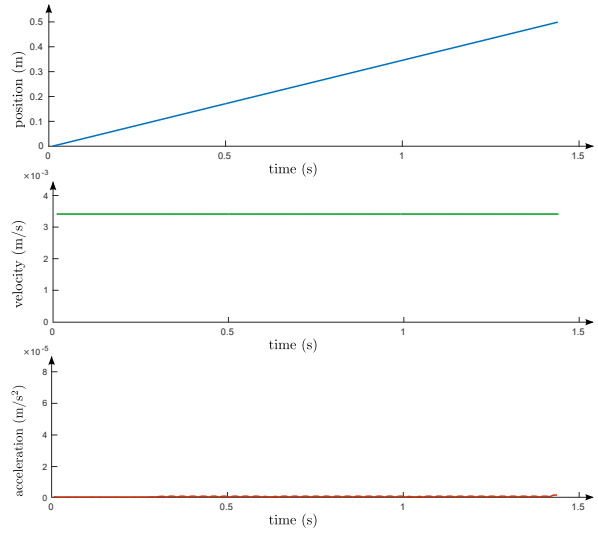
reaching a maximum velocity if,

$$L \geq \frac{v_{max}^2}{a_{max}} \quad (6.3)$$

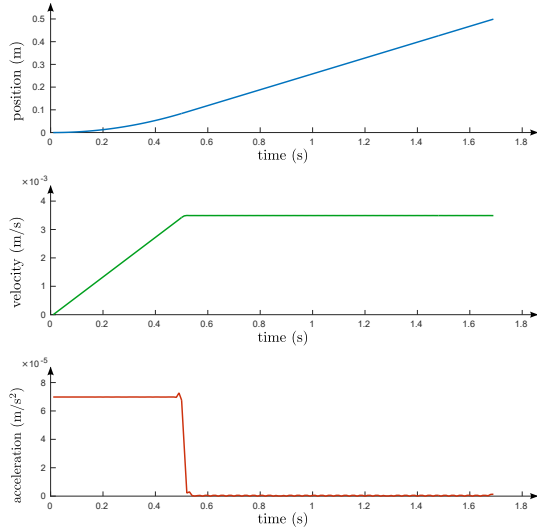
as depicted in Figure 6.7a. On the other hand, if the path is composed of 3 or more segments with the same maximum path velocity, then the first segment profile takes the shape of a left side trapezium Figure 6.7c, the mid-segment a rectangular profile Figure 6.7b and the final segment a right side trapezium as depicted in Figure 6.7d. These operations are all handled internally per trajectory request.



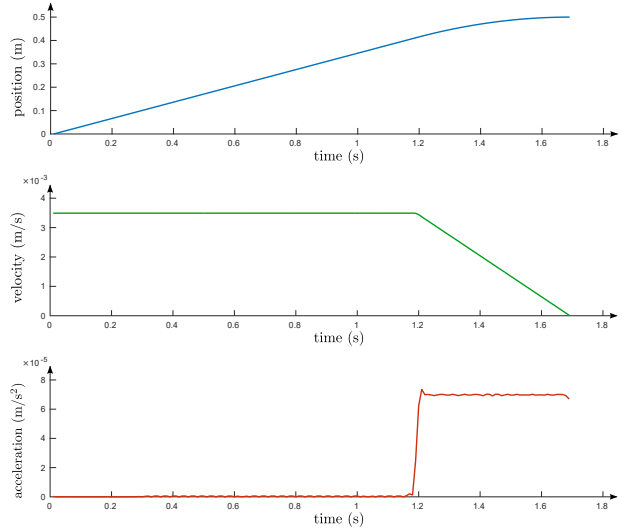
(a) Segment with different start and end velocities. Complete trapezium velocity profile shape.



(b) Segment with the same start and end velocities. Rectangular velocity profile shape.



(c) Segment with different start velocity and same end velocity. Left trapezium velocity profile shape.



(d) Segment with same start velocity and different end velocity. Right trapezium velocity profile shape.

Figure 6.7: The plots depict segments of the same path length (0.5 m) with different start and end velocity conditions, to show how the different velocity profiles adapt to create an overall trapezoidal shape.

6.1.3 Trajectory

The trajectory, as previously stated, combines both the geometric/nullspace path and the velocity profile. Upon calculating the path duration for each segment, the position of the end-effector along the segment can be determined as a function of time.

One of the Cartesian Trajectory Controller's properties is the time-step, which sets the time sampling interval of the trajectory. For the duration of the trajectory and at constant time intervals, the component generates new end-effector poses - containing both translation and rotation. The rate at which the translation and rotation are interpolated depends on the velocity profile as well as on the scaling factors used to match both transformations in path length/time duration. In the end, the component outputs two vectors: one

containing a sequence of frames to be followed in Cartesian space or in nullspace, and a vector containing the time stamp of each pose.

6.2 Kinematics and Statics Module

The Kinematics and Statics Module is the component in the architecture responsible for bridging the task space to the robot's joint space and vice-versa. The component is intimately linked to the robot structure, which in this case falls into the category of 7 DoF anthropomorphic manipulator for the reasons listed in section 5.4. It is based on a novel analytic method to solve the redundant inverse kinematics problem (Faria et al., 2018).

The component provides two interfaces (Figure 6.8): i) the Kinematics with Forward and Inverse solving methods; ii) and the Compute Trajectory with available methods to calculate the joint space trajectory for a specified task space Cartesian or nullspace trajectory. The latter includes one method to analyze the feasibility of a trajectory, one to score the alternative available inverse kinematics solutions, and a method to determine the pose with the highest manipulability. The component requires the Vector Poses & Times interface from the Cartesian Trajectory Controller. Parallel to the offline trajectory computation, the component is able to operate in real-time receiving target frames to follow and generating the respective joint coordinates. Also, it comprises a statics solver to calculate external forces and torques applied to the end-effector from the measured external torques at each joint.

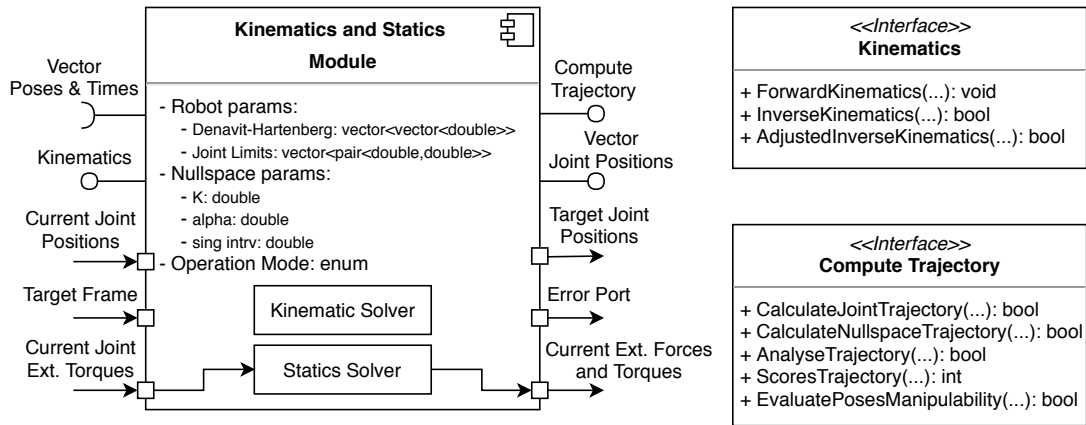


Figure 6.8: Kinematics and Statics Module component and interfaces.

6.2.1 Kinematics Solver - Problem and State-of-the-art

The Kinematics Solver introduces a novel analytic method to uniquely solve inverse kinematics of 7 DoF manipulators while avoiding joint limits and singularities. Two auxiliary parameters are introduced to deal with the self-motion manifolds: the global configuration (GC), which specifies the branch of inverse kinematics solutions; and the arm angle (ψ) that parametrizes the elbow redundancy within the specified branch. The relations between the joint angles and the arm angle are derived, in order to map the joint

limits and singularities to arm angle values. Then, intervals of feasible arm angles for the specified target pose and global configuration are determined, taking joint limits and singularities into account. A simple metric is proposed to compute the elbow position according to the feasible intervals. When the arm angle is determined, the joint angles can be uniquely calculated from the position-based inverse kinematics algorithm. The presented method does not exhibit the disadvantages inherent to the use of the Jacobian matrix and can be implemented in real-time control systems. This novel algorithm is the first position-based inverse kinematics algorithm to solve both global and local manifolds, using a redundancy resolution strategy to avoid singularities and joint limits.

6.2.1.1 Problem definition

The inverse kinematics for a redundant manipulator is an ill-posed and nonlinear problem. DeMers and Kreutz-Delgado (1993a,b) subdivide this problem into local and global ill-posedness, related to the extra-degree of freedom and the multiple solution branches (affects redundant and non-redundant manipulators alike), respectively. The self-motion problem was previously examined through a topological perspective (Burdick, 1989; Luck and Lee, 1993). It has been consensual among authors to split the problem into a finite set of manifolds in the configuration space (c-bundles), with the property that only one solution branch to the inverse kinematics problem may exist in each set. A c-bundle is homotopic and forward maps to the task space as a w-sheet. However, the pre-image of a w-sheet is a collection of c-bundles – as aforementioned – which are usually heterotopic with at least one co-regular surface delimiting its borders. Crossing a co-regular surface, to go from one c-bundle to the next invariably yields passing through a singular configuration. Avoiding singular configurations is often a requirement in task-space motion planning, a condition that motivates the algorithm proposed in this paper.

Within a c-bundle we find the self-motion manifold that derives from the extra DoF. Hollerbach (1985) was the first to propose a description to the manifold as the angle between a reference plane and the plane formed by the shoulder, elbow and wrist – later called “arm angle” (Kreutz-Delgado et al., 1992). Other methods to describe the self-motion were later introduced (Toudu, 2006; An et al., 2014). The redundant degree of freedom enables the manipulator to perform secondary tasks aside from reaching a specific position and orientation (Seraji, 1989), such as: singularities (Chiaverini, 1997; Yan et al., 2014; Nenchev et al., 2004) and joint limits avoidance (Chan and Dubey, 1995; Zhou and Nguyen, 1997; Nelson and Khosla, 1995), obstacle avoidance (Colbaugh et al., 1989; Nakamura et al., 1987), minimise energy dissipation (Suh and Hollerbach, 1987), enhance manipulability (Yoshikawa, 1985a) and human-like motions (Toudu, 2006; Asfour and Dillmann, 2003; Kim and Rosen, 2015).

The kinematic control of a redundant manipulator can be achieved by position- or velocity-based inverse methods, the latter being considered the standard approach to derive the inverse kinematic expressions of redundant manipulators (Nakamura and Hanafusa, 1986; Chang, 1987). However, velocity-based control exhibits several disadvantages when compared to position-based control: i) computationally more expensive, returning only one solution (An et al., 2014; Tolani et al., 2000); ii) difficult to map joint limits or configurations in the velocity domain (Shimizu et al., 2008); iii) subject to error instability due to cumulative

errors; iv) requires the pre-assignment of the Cartesian trajectory to follow (Lee and Bejczy, 1991); v) Jacobian inversion consequent singularities (Dahm and Joublin, 1997). Position-based closed-form solutions were deemed too difficult or impossible to derive due to the infinite number of joint configurations as a solution to the joint space manifold [26]. The introduction of auxiliary parameters to derive inverse kinematics expressions of redundant manipulators is a simple solution that greatly simplifies position-based methods (Kreutz-Delgado et al., 1992; Kuhlemann et al., 2016).

6.2.1.2 Related Work

Lee and Bejczy (1991) presented, in 1991, the first approach towards closed-form inverse kinematics to achieve position-based control. They introduced an algorithm to parametrize redundant joints, transforming the redundant manipulator into a non-redundant image of itself. Their case studies support the premise of position-control advantages over Jacobian-based methods. The proposed algorithm can be theoretically applied to any kinematic structure. Nevertheless, the quality of the solution heavily depends on a proper selection of the redundant joints and on the geometric and algebraic analysis of the non-redundant manipulator image.

The technical report by Dahm and Joublin (1997) describes an analytic solution for the inverse kinematics problem of a 7-DoF serial manipulator with no offsets. The authors introduced the use of the arm angle in the computation of position-based inverse kinematics according to its geometric representation. The authors make an initial approach to map joint limits into the arm angle, as a strategy for joint limit avoidance. However, only the process of computing the arm angles relative to the wrist joint limit is described.

Tolani et al. (2000) published in 2000 a set of inverse kinematic real-time algorithms for anthropomorphic limbs (7-DoF non-offset serial arm or leg). The authors present an interesting work using joint limit avoidance as a primary strategy for redundancy resolution. Two algorithms are introduced to address the inverse kinematics problem of under-constrained orientation and aiming, based on numerical approaches. However, the algorithms are not extended to a robotic structure nor the analytic expressions are derived.

The works of Tondu (2006), Asfour and Dillmann (2003), Kim and Rosen (2015) all report analytic inverse kinematic solutions, based on position-control. With small differences when estimating the arm angle, these works take advantage of the manipulator's redundancy to enhance the human-likeness of the movement. Virtual joint limits are imposed to better imitate the human arm. These limitations, however, also simplify the inverse kinematics solution since they condition the possible configurations. Ultimately, solutions developed for applications under these constraints cannot be directly extended to manipulators with wider working ranges. Also, only Kim and Rosen (2015) addressed the joint limits avoidance problem, by using an optimization routine that relies on estimated weight coefficients to distance from joint limits.

The works of Moradi and Lee (2005) and Shimizu et al. (2008) both describe a closed-form solution to inverse kinematics, and introduce a geometric and an analytic method to map the joint limits in the elbow self-motion manifold. Shimizu et al. developed a new concept to define the reference plane of the arm angle to avoid the algorithmic singularity, common to the technique of determining the reference plane

from an arbitrary vector. Shimizu et al. derived the relation between the arm angle and each joint angle to map joint limits and singularities in the elbow-redundancy circle. The authors also describe an optimization method for joint limit avoidance. Nonetheless, the functions relating the joint and arm angles are not explored in their full domain since the joint limits imposed mimic the human-arm. Also, neither Shimizu's nor Moradi's work propose a method to deal with the global configuration manifold.

Singh and Claassens (2010) and An et al. (2014) describe analytic inverse kinematics solutions for 7-DoF redundant manipulators with non-zero joint offsets. These methods address the elbow's self-motion manifold, by means of an additional parameter, either the arm angle or the theta-1 solution proposed in (Toussaint, 2006). Only Singh and Claassens discuss the problem of avoiding joint limits and provide a method to calculate the arm angle intervals related to the limits of the joints 2, 4 and 6. However, as the authors explain, these constraints are a subset of the solution, since the remaining joint limits were not considered.

Yan et al. (2014) present a closed-form inverse kinematics solution for redundant manipulators. They introduce a novel elbow-redundancy control approach based on dual-arm angle, to avoid situations where the reference plane degenerates to a line. No study was elaborated about the relation between arm angle and joint angles. On the other hand, the paper illustrates a novel approach towards the calculation of the reference plane, and how it affects the configuration control.

Kuhlemann et al. (2016) have recently presented a robust inverse kinematics algorithm that controls the elbow-redundancy and the global configuration. The authors emphasize the importance of consistent joint configuration control to avoid jumping between configurations in a simple trajectory. Their work however, only addresses the inverse kinematic computation, making no allusion to joint limit or singularity avoidance methods.

6.2.2 Analytic Kinematics

Seven-DoF non-offset anthropomorphic manipulators have a S-R-S (spherical-rotational-spherical) kinematic structure. One possible set of Denavit-Hartenberg (DH) parameters that describe the kinematic chain of a S-R-S serial manipulator are listed in Table 6.1. For more information about the DH convention and the determination of the real parameters refer to subsection 7.1.

Table 6.1: Classic Denavit-Hartenberg parameters of a 7-DoF anthropomorphic manipulator.

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	d_{bs}	θ_1
2	0	$\pi/2$	0	θ_2
3	0	$\pi/2$	d_{se}	θ_3
4	0	$-\pi/2$	0	θ_4
5	0	$-\pi/2$	d_{ew}	θ_5
6	0	$\pi/2$	0	θ_6
7	0	0	d_{wf}	θ_7

Each θ_i represents the i^{th} joint position variable, which is physically limited by the joint mechanical upper-limit θ_i^u and lower limit θ_i^l , $i \in \{1, \dots, 7\}$. The column d_i describes the kinematic link lengths

between: (b)ase, (e)lbow, (w)rist and (f)lange (see Figure 6.9).

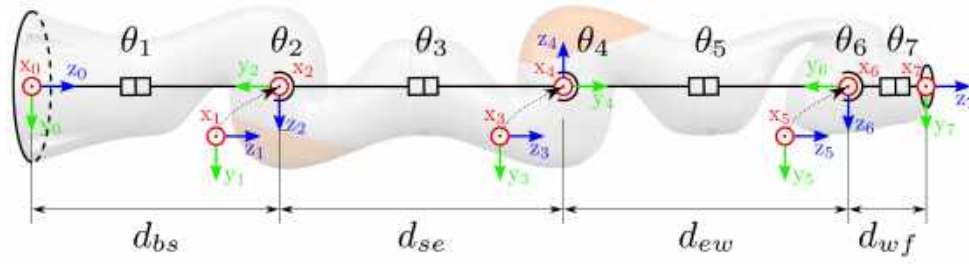


Figure 6.9: Manipulator generic structure, joint variables and DH frames assigned. The 7-DoF manipulator model of LBR iiwa 7 R800 from KUKA AG is used to depict the shape of an anthropomorphic arm without offsets.

The forward kinematics problem is easily solved once the DH parameters are determined. Four parameters are assigned to each joint, which convert to a transformation matrix that establishes the relation between one assigned reference frame ($i - 1$) and the next (i),

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6.4)$$

The product of these matrices from the base to the flange, ${}^0\mathbf{T}_7 = {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 {}^2\mathbf{T}_3 {}^3\mathbf{T}_4 {}^4\mathbf{T}_5 {}^5\mathbf{T}_6 {}^6\mathbf{T}_7$, returns the manipulator's pose in task space.

6.2.2.1 Self-Motion Parameters

To address the global and local self-motion manifolds, two additional parameters are introduced in the calculation of the inverse kinematics. The first parameter – Global Configuration (GC) – uniquely specifies the branch of the inverse kinematics solutions for the global configuration manifold. The second parameter – arm angle (ψ) – introduced by Hollerbach (1985), indicates the elbow position in the redundancy circle. Both the GC and ψ variables are directly determined in the forward kinematics problem from the joint angles, and are passed as parameters to the proposed inverse kinematics algorithm.

The Global Configuration GC_k parameter is split into 3 variables that express the sign of the joint angle coordinates at the shoulder (GC_2), elbow (GC_4) and wrist (GC_6). These variables subsequently exert control over the manipulator's arrangement in space. The GC_k is given by,

$$GC_k = \begin{cases} 1, & \text{if } \theta_k \geq 0 \\ -1, & \text{if } \theta_k < 0. \end{cases}, \forall k \in \{2, 4, 6\}. \quad (6.5)$$

The manipulator's arrangement in space is directly affected by the GC_k at the shoulder, elbow and wrist.

The arm angle, represents the angle formed by the shoulder-elbow-wrist plane (SEW) and the reference plane (SE^vW), as shown in Figure 6.10. The elbow redundancy depends solely on the structure of the manipulator and on the shoulder-wrist vector. The center of the ψ circle is located at half the distance

along the straight line connecting the shoulder to the wrist, and its curve is inscribed in the plane defined by the shoulder-wrist vector.

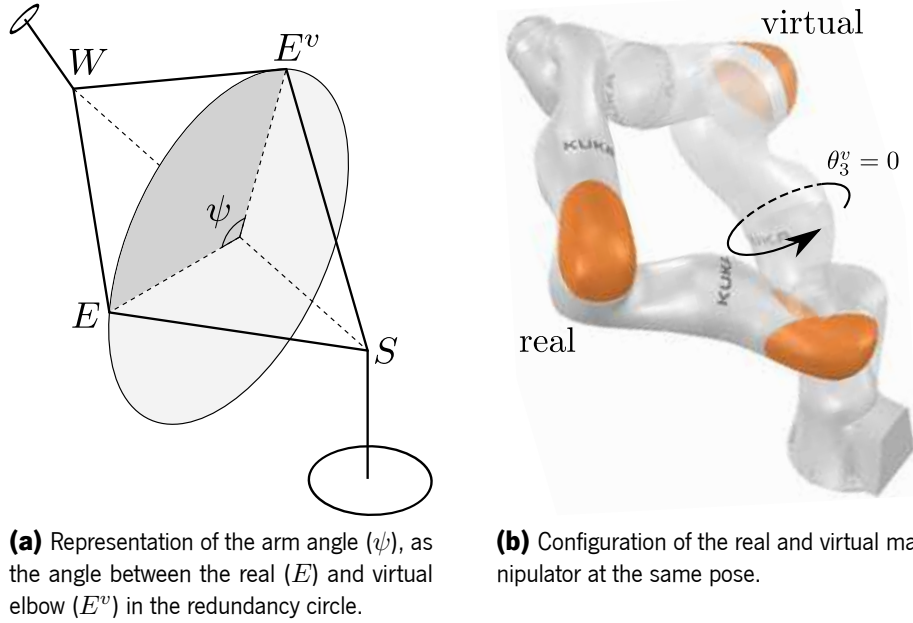


Figure 6.10: The “real manipulator” depicts a 7-DoF manipulator at an arbitrary pose. The “virtual manipulator” is the non-redundant (6-DoF) replica of the same manipulator reaching the same pose. The real manipulator is transformed into the virtual manipulator by blocking the 3th joint at zero ($\theta_3 = 0$).

The selection of the reference plane is not straightforward and has been a topic of discussion in several works (see e.g. (Shimizu et al., 2008; Yan et al., 2014)). The first approach (Kreutz-Delgado et al., 1992; Dahm and Joublin, 1997) was to define the reference plane based on the shoulder and wrist positions, and an arbitrary vector. Whilst this method is the easiest to compute and establishes a clear relation between the arm angle and the task space, it also raises algorithmic singularities that derive from the possibility of the arbitrary vector being collinear with the shoulder-wrist vector. In this case, the reference plane can no longer be determined since the vectors that describe the plane are linearly dependent.

Shimizu et al. (2008) introduced a workaround to the algorithmic singularity by defining the reference plan based on a virtual manipulator that is a non-redundant image of the original 7-DoF manipulator, with $\theta_3 = 0$. If no joint limits are imposed to the virtual manipulator, provided a target pose within its workspace, one can find a finite set of solutions for the inverse kinematics problem. These solutions, depend on the virtual manipulator configuration, which was not addressed in the referred paper.

Yan et al. (2014) presented the “dual arm angle” algorithm, which uses two perpendicular arbitrary vectors to define the reference plane, following the same approach as Kreutz-Delgado et al. (1992). The algorithm follows the principle that the shoulder-wrist vector cannot be collinear with both arbitrary vectors at the same time, so when the singularity occurs for one of the arbitrary vectors, they switch to the other one to generate the reference plane.

To develop an algorithm that solves global configuration self-motion along with the elbow redundancy, it should guarantee that: 1) the definition of the reference plane needs to account for the joint configuration,

the GC parameter; 2) the homotopic property within c-bundles should be extended to the nullspace definition – function of ψ . That is, any function in task-space should be continuously deformed into a function of the ψ parameter for a specific global configuration branch. The method from Yan et al. satisfies the first premise, but not the second. For this reason, we extended the method originally proposed by Shimizu et al. (2008) to consider the robot joint configuration.

6.2.2.2 Elbow redundancy

The arm angle, ψ , is the angle between the vectors normal to the real manipulator SEW plane and the virtual (non-redundant) manipulator SE^vW plane (Figure 6.10). There is however a small caveat to the statement, since the same plane can be defined by both its vector and its negative vector. This indetermination is relevant to our problem because the definition of the reference SEW plane vector depends on the global configuration of the virtual manipulator, i.e. if the elbow is upwards or downwards. To uniquely determine a reference plane vector, we associate the robot configuration variable GC to the calculation of the reference plane.

It is easy to prove that the shoulder and wrist positions are common to the real and virtual robot for the same target pose (Figure 6.10). Therefore, we only need the elbow position of the virtual manipulator to find the reference plane and consequently calculate ψ . Inverse kinematic expressions for the 6-DoF virtual manipulator are applied to discover its elbow position. The joint positions of the virtual manipulator will be represented by a superindex θ_i^v .

Let the target pose be represented by a transformation matrix ${}^0\mathbf{T}_7$, that combines a position component ${}^0\mathbf{p}_7 \in \mathbb{R}^3$ and a rotation component ${}^0\mathbf{R}_7 \in SO(3)$. To simplify the notation of the following equations, we list the vectors from: base to shoulder (${}^0\mathbf{p}_2$), shoulder to elbow (${}^2\mathbf{p}_4$), elbow to wrist (${}^4\mathbf{p}_6$) and wrist to flange (${}^6\mathbf{p}_7$) according to the DH parameters:

$$\begin{aligned} {}^0\mathbf{p}_2 &= \begin{bmatrix} 0 & 0 & d_{bs} \end{bmatrix}^T \\ {}^2\mathbf{p}_4 &= \begin{bmatrix} 0 & d_{se} & 0 \end{bmatrix}^T \\ {}^4\mathbf{p}_6 &= \begin{bmatrix} 0 & 0 & d_{ew} \end{bmatrix}^T \\ {}^6\mathbf{p}_7 &= \begin{bmatrix} 0 & 0 & d_{wf} \end{bmatrix}^T. \end{aligned}$$

The virtual elbow joint (θ_4^v) is the first to be calculated, since it only depends on the manipulator's kinematic structure and on the shoulder-wrist vector. The shoulder-wrist vector is calculated from:

$${}^2\mathbf{p}_6 = {}^0\mathbf{p}_7 - {}^0\mathbf{p}_2 - ({}^0\mathbf{R}_7 {}^6\mathbf{p}_7) \quad (6.6)$$

and θ_4^v is computed using the law of cosines

$$\theta_4^v = GC_4 \arccos \left(\frac{\|{}^2\mathbf{p}_6\|^2 - (d_{se})^2 - (d_{ew})^2}{2 d_{se} d_{ew}} \right). \quad (6.7)$$

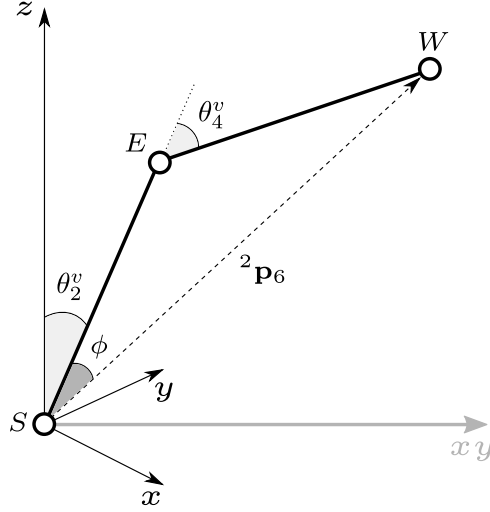


Figure 6.11: Representation of the virtual manipulator and variables required for the θ_2^v calculation.

The elbow configuration variable (GC_4) – the signal of the 4^{th} joint angle – is required to uniquely define the reference plane (6.7).

Since θ_3^v is 0, the shoulder-elbow vector (${}^2\mathbf{p}_4$), as well as the elbow-wrist vector (${}^4\mathbf{p}_6$) are aligned in the xy -plane. The joint θ_1^v is thus responsible for moving the virtual arm to the wrist x- and y- position coordinates.

However, if the shoulder-wrist vector (${}^2\mathbf{p}_6$) is aligned to the z -axis of joint 1 (${}^0\mathbf{R}_{1,z}$), then joint 1 is no longer defined and an algorithmic singularity occurs. Hence, the calculation of θ_1^v branches into,

$$\theta_1^v = \begin{cases} \text{atan2}({}^2\mathbf{p}_{6,y}, {}^2\mathbf{p}_{6,x}), & \text{if } \|{}^2\mathbf{p}_6 \times {}^0\mathbf{R}_{1,z}\| > 0 \\ 0, & \text{if } \|{}^2\mathbf{p}_6 \times {}^0\mathbf{R}_{1,z}\| = 0 \end{cases}. \quad (6.8)$$

The virtual shoulder joint θ_2^v is the last missing variable to find the virtual elbow position. As can be seen in Figure 6.11, ϕ can be calculated from the law of cosines:

$$\phi = \arccos\left(\frac{(d_{se})^2 + \|{}^2\mathbf{p}_6\|^2 - (d_{ew})^2}{2 d_{se} \|{}^2\mathbf{p}_6\|}\right), \quad (6.9)$$

and the value of θ_2^v , which depends on the elbow configuration, is:

$$\theta_2^v = \text{atan2}\left(\sqrt{({}^2\mathbf{p}_{6,x})^2 + ({}^2\mathbf{p}_{6,y})^2}, {}^2\mathbf{p}_{6,z}\right) + GC_4 \phi. \quad (6.10)$$

With θ_1^v , θ_2^v , θ_3^v and θ_4^v we can calculate the pose of the virtual elbow from the base (${}^0\mathbf{T}_4^v$) using the forward kinematics (6.4). The normal vector to the plane ($\mathbf{v}_{sew}^v$) is now calculated as the cross product of the unit vectors that link shoulder-elbow and shoulder-wrist:

$$\mathbf{v}_{sew}^v = \left(\frac{{}^0\mathbf{p}_4^v - {}^0\mathbf{p}_2^v}{\|{}^0\mathbf{p}_4^v - {}^0\mathbf{p}_2^v\|}\right) \times \left(\frac{{}^0\mathbf{p}_6^v - {}^0\mathbf{p}_2^v}{\|{}^0\mathbf{p}_6^v - {}^0\mathbf{p}_2^v\|}\right). \quad (6.11)$$

The normal vector to the real arm SEW plane ($\mathbf{v}_{sew}$) is calculated using the same formula with the position

of the real elbow instead,

$$\mathbf{v}_{sew} = \left(\frac{{}^0\mathbf{p}_4 - {}^0\mathbf{p}_2}{\|{}^0\mathbf{p}_4 - {}^0\mathbf{p}_2\|} \right) \times \left(\frac{{}^0\mathbf{p}_6 - {}^0\mathbf{p}_2}{\|{}^0\mathbf{p}_6 - {}^0\mathbf{p}_2\|} \right). \quad (6.12)$$

Let $\psi \in [-\pi, \pi]$ and in particular ψ be 0 at the reference plane. The sign of the arm angle parameter is determined as,

$$sg_\psi = \text{sgn} \left[\left(\widehat{\mathbf{v}_{sew}^v} \times \widehat{\mathbf{v}_{sew}} \right) \cdot {}^2\mathbf{p}_6 \right] \quad (6.13)$$

and ψ as,

$$\psi = sg_\psi \arccos \left(\widehat{\mathbf{v}_{sew}^v} \cdot \widehat{\mathbf{v}_{sew}} \right). \quad (6.14)$$

Since the arm angle, ψ , is now determined, along with the joint configuration (GC) and the final robot pose (${}^0\mathbf{T}_7$), there exists a full description of the robot configuration in task space, which maps to an unique manipulator configuration in joint space.

6.2.3 Inverse Kinematics

The described inverse kinematics algorithm is based on the standard approach of dividing the manipulator into the upper and lower arm, responsible for positioning and orientation respectively. In the same way that forward kinematics computes the final pose (${}^0\mathbf{T}_7$) and self-motion parameters (ψ, GC) from the joint positions, reciprocally, the inverse kinematics requires these three variables to generate a set of joint positions.

The first step consists of determining the shoulder-wrist vector (${}^2\mathbf{p}_6$). The same equation used for the virtual manipulator (6.6) can be applied for the real robot, since the shoulder and wrist positions are coincident. Moreover, because the shoulder to wrist vector is the same, also the elbow joint (θ_4) can be calculated from (6.7).

The θ_4 is the only joint angle that does not depend on ψ . In order to determine the other joint angles, we need to find the position of the real manipulator elbow. To do so, first we calculate the virtual elbow pose at the reference plane (${}^0\mathbf{T}_4^v$), following the algorithm described in section 6.2.2.2. The real elbow pose is no more than the virtual elbow pose, rotated around the shoulder-wrist axis (${}^2\mathbf{p}_6$) of ψ ,

$${}^0\mathbf{R}_4 = {}^0\mathbf{R}_\psi {}^0\mathbf{R}_4^v \quad (6.15)$$

which is equivalent to

$${}^0\mathbf{R}_3 = {}^0\mathbf{R}_\psi {}^0\mathbf{R}_3^v, \quad (6.16)$$

because θ_4 is the same in the virtual and the real manipulator, hence ${}^3\mathbf{R}_4^v = {}^3\mathbf{R}_4$.

The elbow redundancy rotation matrix (${}^0\mathbf{R}_\psi$) codes the rotation of the angle ψ around the shoulder-wrist

vector (${}^2\mathbf{p}_6$), Figure 6.10a. It is calculated using the Rodrigues's rotation formula in matrix notation,

$${}^0\mathbf{R}_\psi = \mathbf{I}_3 + \sin(\psi) [\widehat{{}^2\mathbf{p}_6} \times] + (1 - \cos(\psi)) [\widehat{{}^2\mathbf{p}_6} \times]^2 \quad (6.17)$$

where $[\widehat{{}^2\mathbf{p}_6} \times]$ is the cross-product matrix for the unit vector $\widehat{{}^2\mathbf{p}_6}$.

Substituting (6.17) into (6.16) and following the notation in Shimizu et al. (2008), ${}^0\mathbf{R}_3$ can be obtained in terms of three auxiliary matrices $\mathbf{A}_s$, $\mathbf{B}_s$ and $\mathbf{C}_s$,

$${}^0\mathbf{R}_3 = \mathbf{A}_s \sin(\psi) + \mathbf{B}_s \cos(\psi) + \mathbf{C}_s \quad (6.18)$$

where,

$$\mathbf{A}_s = [\widehat{{}^2\mathbf{p}_6} \times] {}^0\mathbf{R}_3^v$$

$$\mathbf{B}_s = -[\widehat{{}^2\mathbf{p}_6} \times]^2 {}^0\mathbf{R}_3^v$$

$$\mathbf{C}_s = [\widehat{{}^2\mathbf{p}_6} \widehat{{}^2\mathbf{p}_6}^T] {}^0\mathbf{R}_3^v.$$

The real values of θ_1 , θ_2 and θ_3 are now determined analytically by combining the elements² of the ${}^0\mathbf{R}_3(\theta_{1,2,3})$ matrix – derived from DH parameters – in trigonometric operations,

$${}^0\mathbf{R}_3(\theta_{1,2,3}) = \begin{bmatrix} * & \cos \theta_1 \sin \theta_2 & * \\ * & \sin \theta_1 \sin \theta_2 & * \\ -\sin \theta_2 \cos \theta_3 & \cos \theta_2 & -\sin \theta_2 \sin \theta_3 \end{bmatrix}. \quad (6.19)$$

The * symbol indicates omitted elements of the matrix, which are not required for joint position calculations.

It is important to note that these joint angles are subject to the global configuration parameter GC , thus:

$$\theta_1 = \text{atan2} (GC_2[a_{s22} \sin \psi + b_{s22} \cos \psi + c_{s22}], GC_2[a_{s12} \sin \psi + b_{s12} \cos \psi + c_{s12}]) \quad (6.20)$$

$$\theta_2 = GC_2 \arccos (a_{s32} \sin \psi + b_{s32} \cos \psi + c_{s32}) \quad (6.21)$$

$$\theta_3 = \text{atan2} (GC_2[-a_{s33} \sin \psi - b_{s33} \cos \psi - c_{s33}], GC_2[-a_{s31} \sin \psi - b_{s31} \cos \psi - c_{s31}]) \quad (6.22)$$

Once we know the rotation matrix relative to the shoulder joints (${}^0\mathbf{R}_3$), it is straightforward to compute the rotation matrix of the wrist joints (${}^4\mathbf{R}_7$)

$${}^4\mathbf{R}_7 = \mathbf{A}_w \sin(\psi) + \mathbf{B}_w \cos(\psi) + \mathbf{C}_w \quad (6.23)$$

²The notation m_{ij} is used to specify the element of the $\mathbf{M}$ matrix at the i^{th} line and j^{th} column

where,

$$\mathbf{A}_w = {}^3\mathbf{R}_4^T \mathbf{A}_s^T {}^0\mathbf{R}_7$$

$$\mathbf{B}_w = {}^3\mathbf{R}_4^T \mathbf{B}_s^T {}^0\mathbf{R}_7$$

$$\mathbf{C}_w = {}^3\mathbf{R}_4^T \mathbf{C}_s^T {}^0\mathbf{R}_7.$$

and now we can analytically extrapolate the joint positions of the wrist joints from the elements of the ${}^4\mathbf{R}_7(\theta_{5,6,7})$ matrix,

$${}^4\mathbf{R}_7(\theta_{5,6,7}) = \begin{bmatrix} * & * & \cos \theta_5 \sin \theta_6 \\ * & * & \sin \theta_5 \sin \theta_6 \\ -\sin \theta_6 \cos \theta_7 & \sin \theta_6 \sin \theta_7 & \cos \theta_6 \end{bmatrix}. \quad (6.24)$$

Respecting the global configuration parameter, we compute the remaining joint angles,

$$\theta_5 = \text{atan2} \left(GC_6 [a_{w23} \sin \psi + b_{w23} \cos \psi + c_{w23}], \right. \\ \left. GC_6 [a_{w13} \sin \psi + b_{w13} \cos \psi + c_{w13}] \right) \quad (6.25)$$

$$\theta_6 = GC_6 \arccos (a_{w33} \sin \psi + b_{w33} \cos \psi + c_{w33}) \quad (6.26)$$

$$\theta_7 = \text{atan2} \left(GC_6 [a_{w32} \sin \psi + b_{w32} \cos \psi + c_{w32}], \right. \\ \left. GC_6 [-a_{w31} \sin \psi - b_{w31} \cos \psi - c_{w31}] \right). \quad (6.27)$$

In conclusion, the joint angles are uniquely determined based on a target pose (${}^0\mathbf{T}_7$), and two auxiliary parameters, the arm angle (ψ) and the joint configuration (GC).

6.2.4 Limits and Singularities

Joint limits and singularity avoidance are two crucial conditions to guarantee a stable and feasible motion in task space. The described method maps joint limits and singularities as intervals of the elbow redundancy circle; these intervals specify whether ψ will drive the robot to a joint limit violation or singularity violation (see Figure 6.12).

Feasible intervals are labeled $\Psi_{i,j}$, where the index indicates interval j of the i^{th} joint. The final interval (Ψ_{all}) is the intersection of the feasible intervals of all joints,

$$\Psi_{all} = \bigcap_{i=1}^7 \Psi_i \quad \text{and} \quad \Psi_i = \bigcup_{j=1}^{n_j} \Psi_{i,j}. \quad (6.28)$$

The novelty of the proposed method has to do with the specification of the joint configuration when mapping the joint limits and singularities to the redundancy space. To determine these intervals, we will use the relation between joint angles and the elbow self-motion expressed in section 6.2.3. These relations are described in joints of pivot-type through a tangent function (6.20), (6.22), (6.25) and (6.27), and in joints of hinge-type through a cosine function (6.21) and (6.26). It is important to stress that the elbow

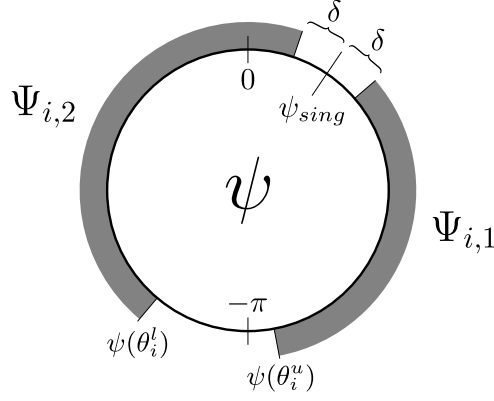


Figure 6.12: Example of joint limits and singularity intervals inscribed in the elbow redundancy circle for joint i . The $\psi(\theta_i^l)$ and $\psi(\theta_i^u)$ represent the arm angles where the respective joint limits are met, and ψ_{sing} is a singular arm configuration. An avoid interval is set next to the singular point with δ representing the unilateral spacing.

joint angle, θ_4 , does not depend on the arm angle and is therefore omitted from the following analysis.

The expressions that represent the joint position, θ_i , as a function of the arm angle, ψ , can be summarized in two generic shapes: one for the pivot joints ($i = 1, 3, 5, 7$) where³,

$$\theta_i(\psi) = \text{atan2}(GC_k [a_n \sin \psi + b_n \cos \psi + c_n], GC_k [a_d \sin \psi + b_d \cos \psi + c_d]) \quad (6.29)$$

and one for the hinge joints⁴ ($i = 2, 6$),

$$\theta_i(\psi) = GC_k \arccos(a \sin \psi + b \cos \psi + c). \quad (6.30)$$

where $k = 2$ for $i \in \{1, 2, 3\}$ and $k = 6$ for $i \in \{5, 6, 7\}$. Having two generic functions to represent the relation between joint angles and the arm angle simplifies the analysis of the $\theta_i(\psi)$ function. It is possible to determine the intervals of ψ that correspond to feasible joint angles by solving these equations as a function of the joint limits, θ_i^l and θ_i^u . This is explained next.

6.2.4.1 Pivot Joints

In order to determine the arm angle intervals that map to the joint limits, or the singular arm angle, we need to differentiate these expressions. Simplifying the notation of (6.29) let $\theta_i(\psi) = \text{atan2}(u, v)$, where

$$u = GC_k(a_n \sin \psi + b_n \cos \psi + c_n)$$

$$v = GC_k(a_d \sin \psi + b_d \cos \psi + c_d)$$

³ a_n, b_n and c_n are the signed coefficients of the first input of atan2 in equations (6.20), (6.22), (6.25) and (6.27), whereas index a_d, b_d and c_d represent the signed coefficients of the second input.

⁴ a, b and c are the signed coefficients of $\arccos$ in equations (6.21) and (6.26)

its total derivative⁵ is given by,

$$\frac{d\theta_i}{d\psi} = \frac{\partial\theta_i}{\partial u} \frac{\partial u}{\partial\psi} + \frac{\partial\theta_i}{\partial v} \frac{\partial v}{\partial\psi} = \frac{vu'}{v^2 + u^2} - \frac{uv'}{v^2 + u^2}. \quad (6.31)$$

After simplification we obtain the following expression,

$$\frac{d\theta_i}{d\psi} = \frac{a_t \sin \psi + b_t \cos \psi + c_t}{v^2 + u^2} \quad (6.32)$$

where,

$$a_t = GC_k(c_n b_d - b_n c_d)$$

$$b_t = GC_k(a_n c_d - c_n a_d)$$

$$c_t = GC_k(a_n b_d - b_n a_d).$$

Using the Weierstrass-substitution method, the stationary points (ψ_0) are given, if they exist, by,

$$\psi_0 = 2 \arctan \left(\frac{a_t \pm \sqrt{a_t^2 + b_t^2 - c_t^2}}{b_t - c_t} \right). \quad (6.33)$$

Depending on the value of $a_t^2 + b_t^2 - c_t^2$ three cases may arise:

- $a_t^2 + b_t^2 - c_t^2 > 0$, with stationary points;
- $a_t^2 + b_t^2 - c_t^2 < 0$, without stationary points;
- $a_t^2 + b_t^2 - c_t^2 = 0$, indeterminate joint angle.

With stationary points The generic profiles of the function (6.29), when it has stationary points, is depicted in Figure 6.13. Stationary points represent the maximums, minimums or inflection points of the $\theta_i(\psi)$ function. Shimizu et al. (2008) state that stationary points can be proved to be either global maximums or minimums, and elaborate on how the joint limits can be represented on the elbow redundancy circle, based on this premise (Figure 6.13a). This assumption, however, cannot be extended to a general purpose manipulator. It is common in industrial manipulators to feature actuators with ample working range, close or even beyond the interval of $[-\pi, \pi]$. When the target pose and joint configuration (GC) are specified, the resulting joint positions may be near to a joint limit. Thus, a change in the arm angle may lead to a joint angle variation such that, $\theta_i(\psi) < -\pi \vee \theta_i(\psi) > \pi$. This leads to a discontinuity because the atan2 function maps the arm angle to the joint angle in the $[-\pi, \pi]$ range. When the joint angle overshoots this limit, the returned joint angle goes from π to $-\pi$ or vice-versa (see Figure 6.13b). Hence, the stationary points can not be considered global maximums or minimums.

Without stationary points If the function is monotonic, there are no stationary points and there is a one-to-one correspondence between ψ and θ_i . The function $\theta_i(\psi)$ is not fully defined in the interval

⁵In points where both partial derivatives exist, the function $\text{atan2}(u, v)$ can be differentiated as $\arctan(u/v)$.

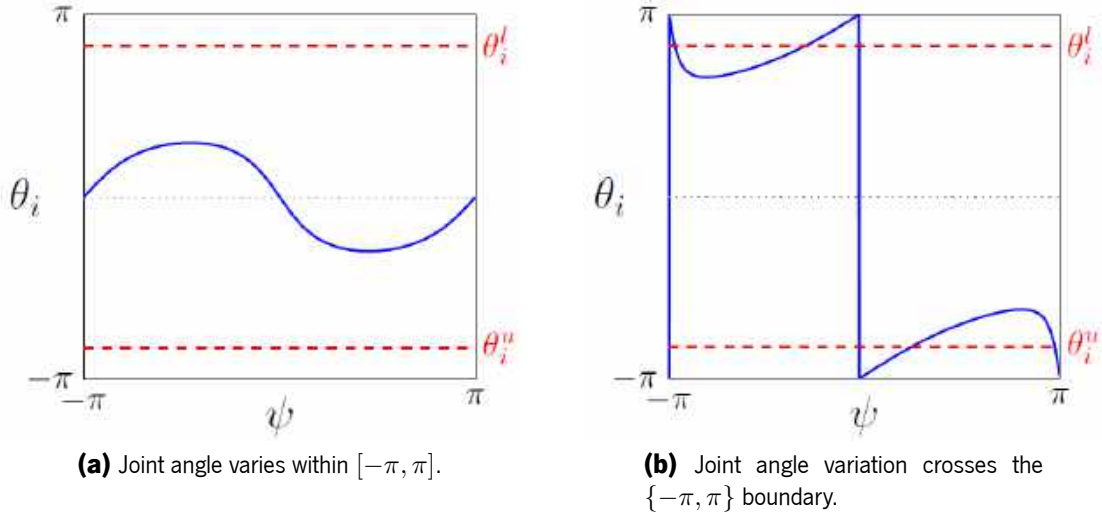


Figure 6.13: Pivot joint $\theta_i(\psi)$ profile with stationary points.

$[-\pi, \pi]$, due to the existence of a point of inversion, where the joint angle is discontinued (see Figure 6.14). This phenomenon has nothing to do with kinematic singularities, it derives from the range where atan2

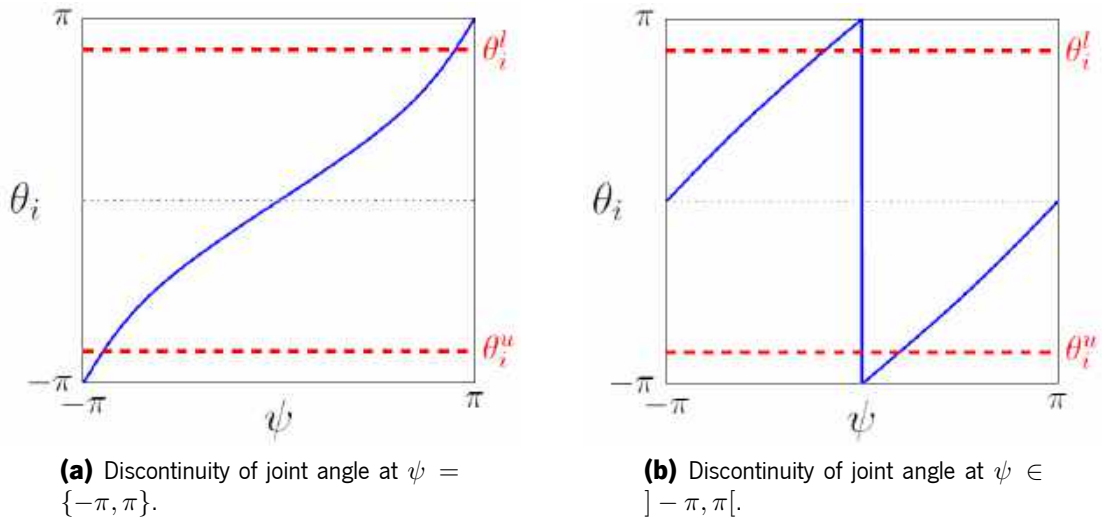


Figure 6.14: Pivot joint $\theta_i(\psi)$ profile without stationary points.

maps the arm to the joint angle. Since the joint limits are commonly found within the $[-\pi, \pi]$ interval, this inversion occurs from outside the joint limits and causes no trouble.

Indeterminate Joint Angle A singular point exists if the third condition is verified ($a_t^2 + b_t^2 - c_t^2 = 0$). The exact singular arm angle ψ_{sing} is directly determined from the stationary points of equation (6.33),

$$\psi_{sing} = 2 \arctan \left(\frac{a_t}{b_t - c_t} \right). \quad (6.34)$$

However, it cannot be labeled a stationary point because neither $\theta_i(\psi)$ nor its derivative is defined at that point. In a numerical approximation scheme it is not feasible to setup a condition that triggers on a single

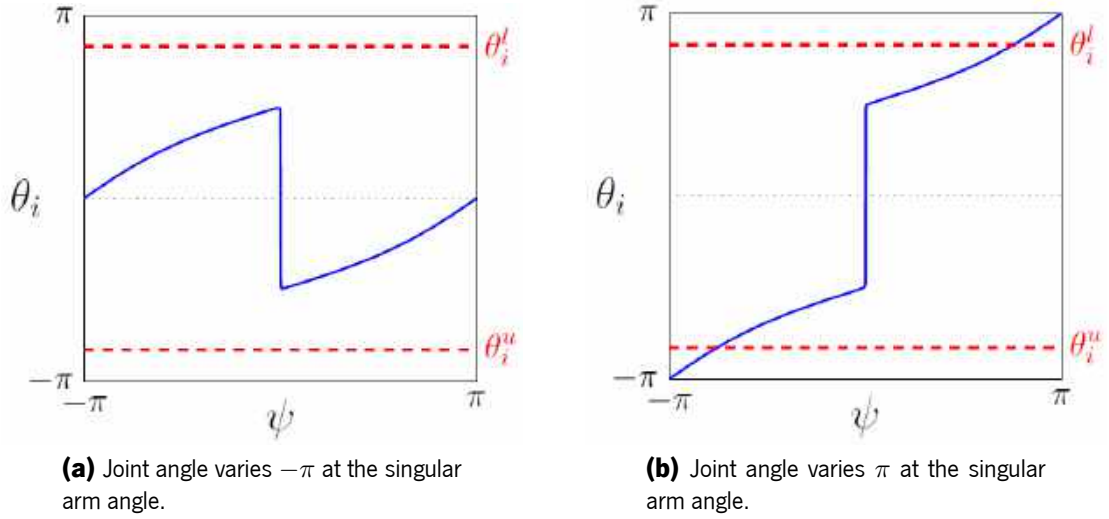


Figure 6.15: Pivot joint $\theta_i(\psi)$ profile when a singular arm angle exists.

value due to inherent numerical errors. That being said, a tolerance threshold is usually defined to trigger whenever the value of $a_i^2 + b_i^2 - c_i^2$ approaches zero. A logic strategy to avoid singularities consists in avoiding intervals that contain the singular arm angle, $[\psi_{sing} - \delta, \psi_{sing} + \delta]$. The δ represents a constant value or a variable that depends on the defined tolerance threshold.

Now that we have identified and proposed a method to deal with singular arm angles, we still need a method to map the joint limits to the arm angle. A simple and effective way to achieve that, consists of solving (6.29) as a function of the joint limits, θ_i^u and θ_i^l . Using the Weierstrass-substitution method we obtain,

$$\psi = 2 \arctan \left(\frac{-b_p \pm \sqrt{b_p^2 - 4a_p c_p}}{2a_p} \right) \quad (6.35)$$

where,

$$a_p = GC_k [(c_d - b_d) \tan(\theta_i) + (b_n - c_n)]$$

$$b_p = 2 GC_k [a_d \tan(\theta_i) - a_n]$$

$$c_p = GC_k [(b_d + c_d) \tan(\theta_i) - (b_n + c_n)].$$

One of three outputs may occur:

- $\exists \psi \in [-\pi, \pi] : \theta_i(\psi) = \theta_i^l \vee \theta_i(\psi) = \theta_i^u$
- $\forall \psi \in [-\pi, \pi], \theta_i(\psi) \in]\theta_i^l, \theta_i^u[$
- $\forall \psi \in [-\pi, \pi], \theta_i(\psi) \notin [\theta_i^l, \theta_i^u]$.

If the first case is met, the $\psi(\theta_i^l)$ or $\psi(\theta_i^u)$ values are found, but that information alone does not specify if the function is entering an out-of-limits zone or an allowed zone. To do so, the first derivative of the function (6.32) is analyzed at these limit arm angles. If $\theta_i(\psi)$ and its derivative have the same sign at $\psi(\theta_i^l)$ or $\psi(\theta_i^u)$, the next interval will be an out-of-limits zone; else, if the sign differs the next joint angles

will be within an allowed zone.

The last two cases occur if θ_i does not coincide with the joint limits, for any arm angle in the $[-\pi, \pi]$ interval. Assuming that $\theta_i(\psi)$ is defined in $[-\pi, \pi]$, any point along the function (6.29) will confirm if the whole interval is possible or not.

6.2.4.2 Hinge Joints

The same methodology used for pivot joints, is applied for hinge joints. The relation between joint and arm angle is represented by (6.30). Applying the derivative to $\theta_i(\psi)$ we obtain,

$$\frac{d\theta_i}{d\psi} = -GC_k \left(\frac{a \cos \psi - b \sin \psi}{\sin \theta_i} \right). \quad (6.36)$$

Using the Weierstrass-substitution again we calculate the stationary points,

$$\psi_0 = 2 \arctan \left(\frac{-b \pm \sqrt{a^2 + b^2}}{a} \right). \quad (6.37)$$

The function exists in either $[-\pi, 0]$ or $[0, \pi]$ depending on the value of the configuration parameter, GC , for that particular joint. Depending on the value of $a^2 + b^2$ two cases may arise:

- $a^2 + b^2 > 0$, two stationary points;
- $a^2 + b^2 = 0$, one stationary point.

If the first condition is fulfilled, the hinge joint has a profile identical to the one depicted in Figure 6.16. Two stationary points exist in this case, one being the global minimum and the other the global maximum. Depending on the required pose and configuration parameter, the joint angle variation may be close to the joint limit, hence the need to check it for the hinge joints as well.

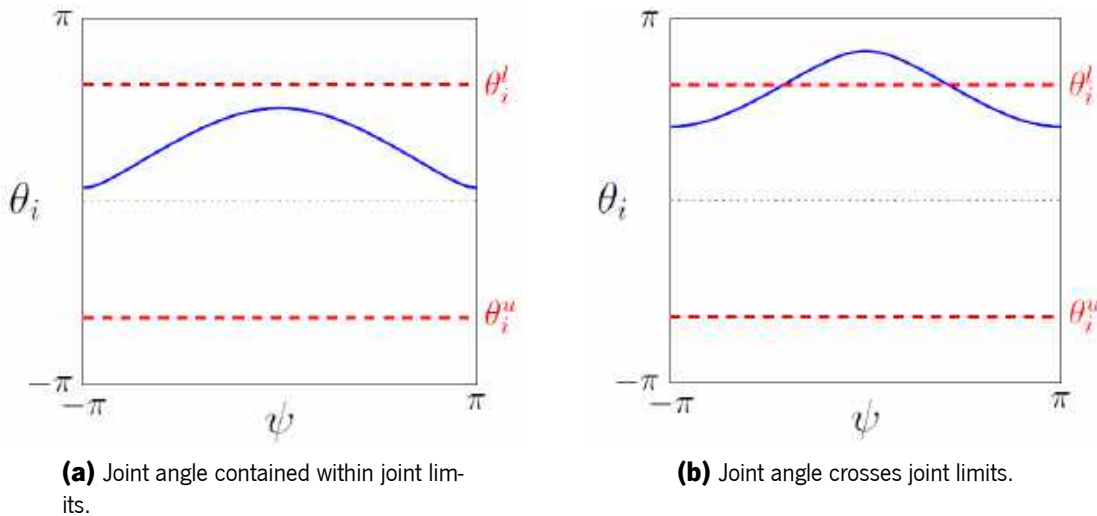


Figure 6.16: Hinge joint $\theta_i(\psi)$ profile when the configuration parameter is positive, function defined in $[0, \pi]$. If only the configuration is changed to a negative value, the function profile is mirrored relative to the x-axis.

The arm is in a singular configuration when $\theta_2 = 0$ or $\theta_6 = 0$ hold. In such cases, $\theta_i(\psi)$ is still defined at that particular ψ value but the first derivative is not. For that reason, there is only one stationary point defined at a singular point (Figure 6.17), given by (6.37). If the configuration parameter (GC_k) is positive, the stationary point is the global maximum, else it is the global minimum.

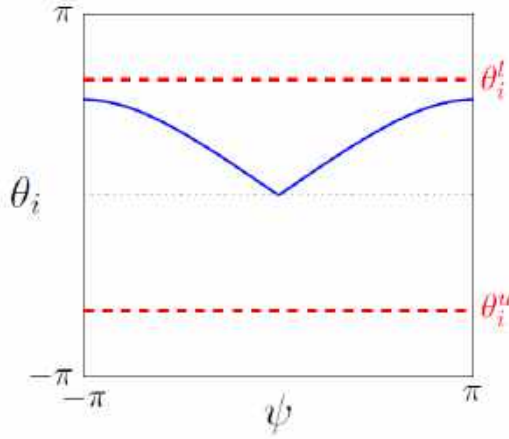


Figure 6.17: Hinge joint $\theta_i(\psi)$ profile when the arm is at a singular configuration.

Since $\theta_i(\psi)$ is continuous at a singular position (Figure 6.17), the singular arm angle in hinge joints does not need to be accounted for. Likewise to the pivot joints, equation (6.30) is solved as a function of the hinge joint limits to explore the associated arm angles.

$$\psi(\theta_i) = 2 \arctan \left(\frac{a \pm \sqrt{a^2 + b^2 - (c - \cos \theta_i)^2}}{\cos \theta_i + b - c} \right) \quad (6.38)$$

The same outputs possible for the pivot joints are true for the hinge joints. If limits are crossed, we can check if the robot is entering an out-of-limits or an allowed zone by following the same metric – of the $\theta_i(\psi)$ and its derivative signs at the arm angles relative to the joint limit. If no limits are crossed, then any θ_i value can be used $\forall \psi \in [-\pi, \pi]$ to determine if the function is contained within the joint limits or if it is outside.

6.2.5 Redundancy Resolution

The proposed inverse kinematics algorithm receives as parameters: the desired end-effector pose (${}^0T_7^d$), the global configuration of the manipulator (GC^d) and the arm angle (ψ). In this section, it will be proposed a redundancy resolution method to calculate the arm angle while avoiding joint limits and singularities.

The configuration control introduced in this section is convenient in scenarios where the control loop is continuously performing inverse kinematics calculations for the robot to follow a given pose. In many applications like for instance human-robot interactions it is highly desirable that the robot follows a smooth and stable trajectory without sudden jumps between different configurations or passes through singular configurations in joint space.

The problem has been formulated such that the joint limits and singularities are projected into the redundancy circle by means of feasible arm angle intervals. As denoted in equations (6.28), each joint has associated to it a union of n_j feasible intervals, $\Psi_{i,j}$ (Figure 6.12), and the global feasible intervals (Ψ_{all}) develops from the intersection of each joint intervals, Ψ_i . The calculation of the arm angle is thus simplified since the only hard-constraints to the problem are the defined intervals.

A plausible approach to calculate the new arm angle (block “Calculate New ψ ” in Figure 6.18) consists of applying a simple metric to the current arm angle, $\psi_{(t-1)}$, according to the current global feasible intervals. The current arm angle, can either be computed from the current joint positions θ_i^c (following the method explained in 6.2.2.2), or directly used from the preceding iteration.

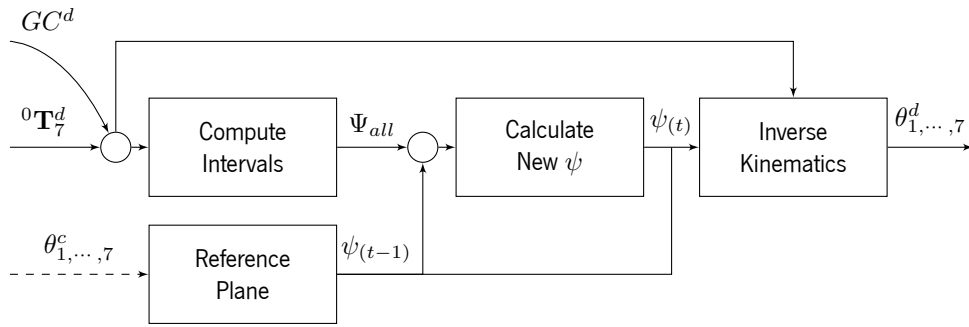


Figure 6.18: General control scheme for inverse kinematics resolution with joint limits and singularity avoidance.

First, we find the feasible interval that contains the previous arm angle
 $(\psi_{(t-1)} \in \Psi_{all,m}, \Psi_{all,m} = [\psi_m^l, \psi_m^u])$, (Figure 6.19). If the previous arm angle is contained in $\Psi_{all,m}$

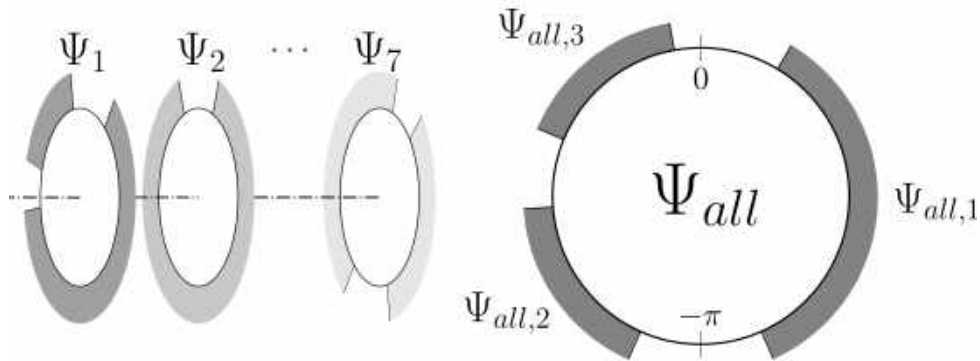


Figure 6.19: Example of intersection of feasible intervals from all the joints. The three resulting intervals are enumerated $m \in \{1, 2, 3\}$.

the new arm angle can be computed as follows:

$$\psi_{(t)} = \psi_{(t-1)} + K \left(\frac{\psi_m^u - \psi_m^l}{2} \right) \left[e^{-\alpha \left(\frac{\psi_{(t-1)} - \psi_m^l}{\psi_m^u - \psi_m^l} \right)} - e^{-\alpha \left(\frac{\psi_m^u - \psi_{(t-1)}}{\psi_m^u - \psi_m^l} \right)} \right] \quad (6.39)$$

where:

- $K \in [0, 1]$, is a constant that controls the strength of repulsion of the arm angle from the limits;

- $\alpha \in \mathbb{R}^+$, is a constant that controls the distance to the limits from where the arm angle starts to repulse.

In case the previous arm angle is not contained in any feasible interval, either the elbow is moved to the closest feasible range, or the motion cannot be continued. This phenomenon is prone to happen when the trajectory has sparse poses or if the α value is too large, which permits the robot to move closer to its joint limits.

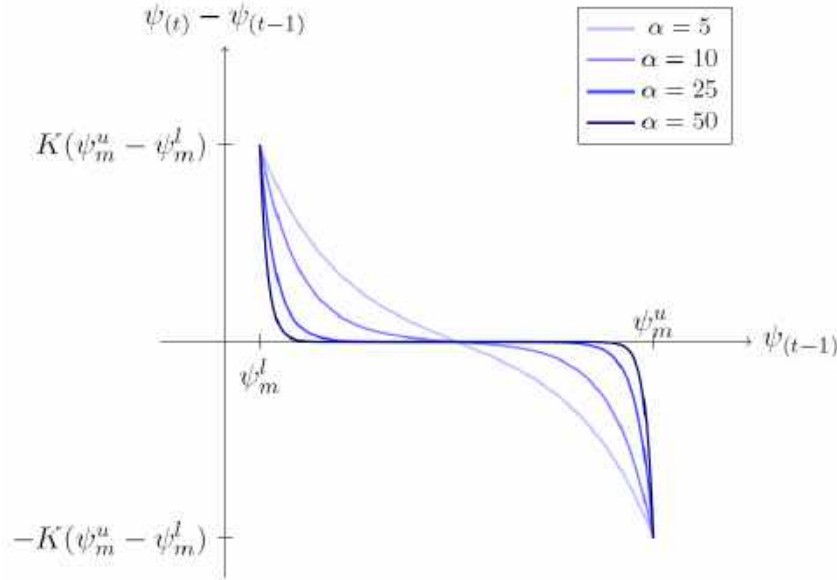


Figure 6.20: Change of the arm angle as a function of $\psi_{(t-1)}$, defined in the feasible interval $\Psi_{all,m}$.

By changing α (cf. Figure 6.20), we can control the tendency of the arm angle to move towards the interval center or to preserve its position as long as it is distant from the limits. The problem of computing the new arm angle can also be tackled from an optimization standpoint, depending on the task requirements.

Finally, the parametrization of the manipulator redundancy as an ellipsoid that describes the elbow motion, has a clear representation in task space. Hence, it is straightforward to impose additional interval limits on the redundancy circle that relate to task space obstacles.

6.2.5.1 Algorithm performance

The algorithm was implemented in MATLAB and C++ (Boost, Eigen) and the computational times were measured for the different parts of the above presented algorithm: i) the “Inverse Kinematics” algorithm alone; ii) the “Compute Intervals”, joint limits and singularities mapping to the redundancy circle; and the complete algorithm, Figure 6.18. Each algorithm was tested for a set of 10000 random and feasible poses. The performance of the algorithm was tested in a computer running Ubuntu 14.04 (Intel i7-4710MQ at 2.5GHz, 16GB of RAM).

Table 6.2: Descriptive statistics of computational times recorded of different parts of the algorithm, measured from 10000 samples, in microseconds (μs).

		Mean	Std. Deviation	Maximum
MATLAB	Inverse Kinematics	230.3	21.93	642.0
	Compute Intervals	822.3	88.61	622.0
	Full Algorithm	1132	77.65	3345
C++	Inverse Kinematics	2.247	1.228	32.00
	Compute Intervals	10.51	2.015	37.00
	Full Algorithm	12.74	2.103	43.00

6.2.5.2 Trajectory example

To demonstrate the redundancy resolution strategy, we created an example linear trajectory to be performed by the robot manipulator. The robot is purposely positioned at a configuration near its mechanical joint limits. The robot starts at the joint angles in degrees:

$$\theta^c = [-5.4101 \quad -26.4986 \quad -48.1542 \quad -61.6500 \quad 152.6198 \quad 114.4466 \quad 8.1812] \quad (6.40)$$

which correspond to the global configuration $GC = 3$, arm angle $\psi = 58.5882^\circ$ and pose:

$${}^0\mathbf{T}_7^c = \begin{bmatrix} -0.2634 & -0.9112 & -0.3166 & -0.1174 \\ 0.3014 & -0.3895 & 0.8703 & -0.1464 \\ -0.9164 & 0.1338 & 0.3773 & 1.0203 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.41)$$

The manipulator performs a linear motion in Cartesian space, keeping the same end-effector orientation but translating its position along the z-axis (${}^0\mathbf{R}_{7,z}$) of a distance of 0.25 m, ending at the target pose:

$${}^0\mathbf{T}_7^d = \begin{bmatrix} -0.2634 & -0.9112 & -0.3166 & -0.1966 \\ 0.3014 & -0.3895 & 0.8703 & 0.0712 \\ -0.9164 & 0.1338 & 0.3773 & 1.1146 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.42)$$

The path is interpolated and a new pose is passed to the redundancy resolution algorithm (Figure 6.18) every iteration. The global configuration remains unchanged throughout the trajectory, and the arm angle varies according to the parameters defined (α and K)⁶.

First, the impact of the parameter α is evaluated. Keeping the parameter $K = 0.1$ constant in all tests, we depict how different values of α affect the arm angle motion along the trajectory (Figure 6.21a). Then, the value of α is fixated at 20 and the effect of different values of K is demonstrated (Figure 6.21b). The evolution of the joint angles along the trajectory for each of the tests ran is depicted (Figure 6.21c and 6.21d).

⁶Motion comparison with different values of α and K in video format – (<https://youtu.be/mcMfWkMP1hQ>).

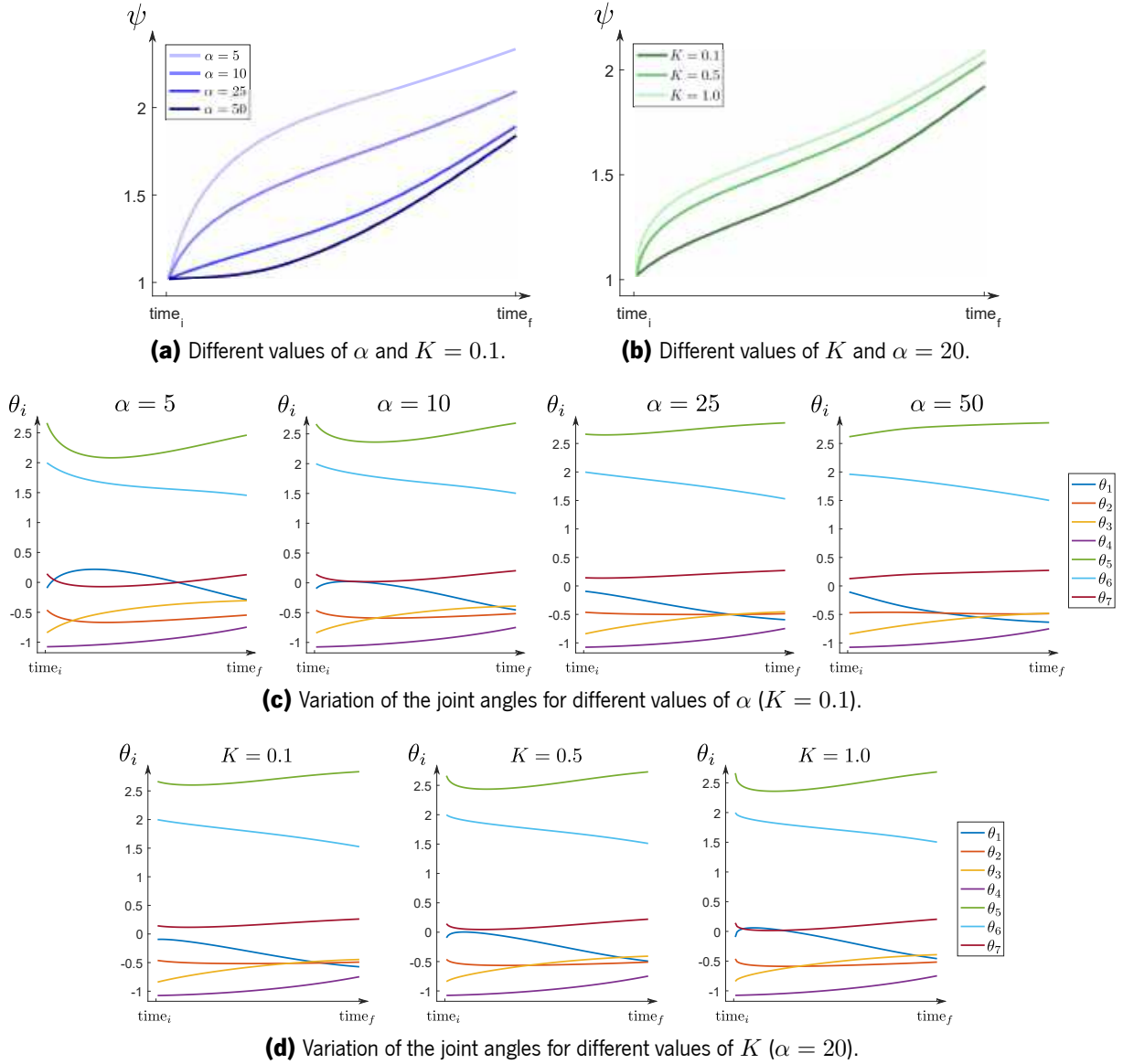


Figure 6.21: Variation of the arm angle (ψ) and joint angles along the trajectory.

In conclusion, the Kinematics Solver uses the concept of global joint configuration control in position-based inverse kinematics calculation and applies the same parameter to derive the expressions that map the joint angles to arm angle. This relation enables the mapping of the joint limits and singularities in the nullspace represented by the redundancy circle. It was shown a method to reliably compute intervals of feasible elbow positions to avoid joint limits and singularities. A simple metric to compute the arm angle from the interval of feasible arm angles was also discussed.

Being an analytic solution, the time required for each operation is limited and quantifiable, which makes this method adequate for real-time control systems. This solution relies on simple mathematical operations that are implemented and fully optimized in mathematical software libraries, making it a very light and fast algorithm. The proposed algorithm is specifically relevant in scenarios where the task execution requires continuous and predictable motion execution, avoiding jumping between configurations, passing singularities or reaching joint limits.

6.2.6 Component's Operation and Other Functionalities

The component operates in one of two different modes,

1. Trajectory Following Control;
2. Online Trajectory Generation.

In the Trajectory Following Control mode, the component receives a complete trajectory in task space frames from the required interface Vector Poses & Times (Cartesian Trajectory Controller) and processes it to generate a vector of corresponding joint positions. When the poses are received from the Cartesian Trajectory Controller component, the joint trajectory is processed when a call is made to the “Compute Trajectory” interface. It handles both the Cartesian space as well as the nullspace trajectories.

If the Online Trajectory Generation mode is active instead, the component operates in real-time receiving the target end-effector frames from the “Target Frame” input port and computing the next target joint positions that then sent through the output port. In this mode, the component also calculates the external forces and torques at the end-effector from the measured external joint torques, using the statics solver (subsection 6.2.7).

The “Kinematics” interface provides two methods to calculate Inverse kinematics. The Inverse Kinematics() requires the target end-effector frame as well as the two redundancy parameters GC and ψ . The Adjusted Inverse Kinematics() only requires the end-effector frame to calculate the joint position solution. The GC parameter is assigned to the current manipulator value and the ψ is calculated based on (6.39).

The CalculateJointTrajectory() and CalculateNullspaceTrajectory() methods from the “Compute Trajectory” interface generate a joint trajectory from the vector of end-effector frames / nullspace parameter, following a logic similar to the scheme in Figure 6.18. Four different outcomes are possible: i) the joint trajectory is feasible and is forwarded to the Joint Trajectory Controller; ii) the trajectory is not feasible from the current ψ (needs to be adjusted prior to trajectory execution); iii) the trajectory is not possible under no ψ of the current GC (requires a manipulator configuration change); iv) the trajectory is not feasible.

In order to guarantee that the robot is capable of performing the required trajectory with a minimum deviation from the current ψ , two additional methods are available. The AnalyseTrajectory() checks through each of the trajectory's frames arm angle intervals (for each configuration) Figure 6.19, and returns a set of possible ψ intervals for each configuration. The ScoresTrajectory() attributes a score to a discrete set of ψ values within the possible intervals, based on the distance from the current arm angle and on the need for a manipulator reconfiguration (change of GC), and returns a code with one of the four possible outcomes referred above.

Finally, the EvaluatePosesManipulability() method classifies a set of poses based on a metric involving the manipulability and the proximity to limits and singularities. The method evaluates all the input poses and, if inverse kinematic solution exist, it returns for each configuration the pose and arm angle that achieve the highest manipulability / distance to limits and singularities metric. More information about this method in subsection 6.2.6.4.

6.2.6.1 Interval Avoidance Strategy

To analyze the trajectory feasibility in terms of the four possible outcomes, we applied a backtracking algorithm that combines the sets of feasible intervals (Ψ_{all}) from the last trajectory frame back to the initial frame. This process is carried out in the `AnalyseTrajectory()` method and requires an additional parameter - $\psi_{dmax} \in \mathbb{R}^+$ - that codes the maximum arm angle displacement between frames / iterations. The algorithm returns a set of initial conditions that guarantee that the robot is capable of performing the received task space trajectory within the same configuration (GC) and with a maximum arm angle displacement (ψ_{dmax}) between iterations.

Instead of using the feasible intervals calculated directly for each frame, the algorithm combines the intervals of the current frame ($\Psi_{all}(i)$) with what we call the “shadow of the intervals” (Ψ_{shd}) from the next frame in time. The “shadow intervals” are no more than the feasible intervals of the next frame in time ($\Psi_{all}(i + 1)$) enlarged by ψ_{dmax} on each upper and lower bounds. Thus, the current frame feasible intervals are the intersection,

$$\Psi_{all}(i) = \Psi_{all}(i) \cap \Psi_{shd}, \quad (6.43)$$

which guarantees that if the arm angle of the current frame is in a feasible interval, it can proceed to a feasible interval of the next trajectory frame (Algorithm 1).

```

input : traInt a vector of trajectory intervals of size sz,  $\psi_{dmax}$ 
output: final feasible intervals finInt

// Create the 'shadow' intervals shdInt vector
1 shdInt  $\leftarrow$  vector(sz, interval_set(empty));
2 shdInt[sz] = traInt[sz]
3 for i  $\leftarrow$  sz - 1 to 0 do
4   foreach interval in traInt[i + 1] do
      // Truncate intervals between  $[-\pi, \pi]$ 
5     lo  $\leftarrow$  max(interval.lower() -  $\psi_{dmax}$ ,  $-\pi$ );
6     up  $\leftarrow$  min(interval.upper() +  $\psi_{dmax}$ ,  $\pi$ );
7     shdInt[i]  $\leftarrow$  shdInt[i]  $\cup$  interval_set(lo, up)
8   end
   // Intersect 'shadow' with current intervals
9   shdInt[i]  $\leftarrow$  shdInt[i]  $\cap$  traInt[i]
10 end
11 finInt = shdInt[0]
```

Algorithm 1: Algorithm to adjust feasible intervals of a trajectory.

6.2.6.2 Nullspace Scoring

From the trajectory analysis, one obtains a set of feasible arm angle intervals for the different global configurations. In order to select the appropriate initial pose for the robotic manipulator to execute the

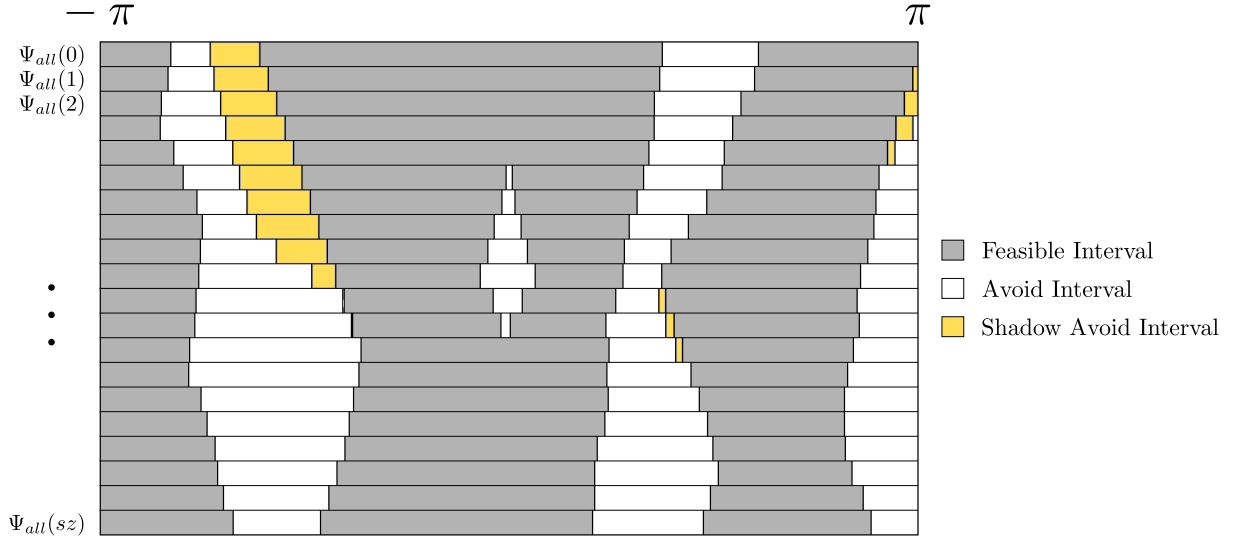


Figure 6.22: Example of standard trajectory with feasible intervals computed from the method presented in subsection 6.2.4, and after applying the interval adjustment algorithm.

trajectory, we created an algorithm that scores the feasible nullspace intervals (ψ_m). The goal is to assign the lowest score to the redundancy parameters better suited to perform the trajectory - minimize the distance to the current parameters and maximize the distance to joint limits and singularities. These intervals are initially discretized into evenly-spaced arm angle samples with the default separation step of $\Delta s = 1^\circ$.

$$\psi_s = n \Delta s + \psi_m^l, \quad n = 0, 1, \dots, N \quad (6.44)$$

where,

$$N = \left\lceil \frac{\psi_m^u - \psi_m^l}{\Delta s} \right\rceil \quad (6.45)$$

Then, the score of each sample ($\sigma(\psi_s)$) is calculated as a variation of (6.39),

$$\sigma(\psi_s) = \|\psi - \psi_s\| + K \left(\frac{\psi_m^u - \psi_m^l}{2} \right) \left[e^{-\alpha \left(\frac{\psi_s - \psi_m^l}{\psi_m^u - \psi_m^l} \right)} - e^{-\alpha \left(\frac{\psi_m^u - \psi_s}{\psi_m^u - \psi_m^l} \right)} \right] \quad (6.46)$$

thus, the lowest the score, the best arm angle alternative.

The scoring mechanic was implemented so that the end-user can ultimately alter the arm angle to a position that better fits the workspace while still accounting for a robot configuration that can accomplish the trajectory.

6.2.6.3 Nearing Singularities

Another question is raised when the manipulator moves along a trajectory composed of a sequence of frames or continuously follows an online trajectory with short time intervals and approaches a singular configuration.

As defined in subsection 6.2.4, the strategy to handle singular arm angles - ψ_{sing} - detected in pivot joints, is to set an avoiding interval, $[\psi_{sing} - \delta, \psi_{sing} + \delta]$, to surround the singular point. The singular point is detected when $a_t^2 + b_t^2 - c_t^2$ from the coefficients of 6.33 is zero, or below a tolerance value in real systems. If the singularity interval is created only when the condition verifies, the manipulator's current arm angle might be near or already within the singularity interval. If that is the case, the manipulator does not have time to adjust the arm angle to avoid the interval and pass near the singular point, with the consequent position discontinuity.

While this phenom can be predicted in trajectories known beforehand, it is unavoidable when the component is in the “Online Trajectory Generation” mode following an end-effector frame that is updated in short time intervals. To cope with this situation, we adjust the singularity interval (δ) with the proximity to the singular point, which occurs when $a_t^2 + b_t^2 - c_t^2$ tends to zero. Consider the variable $\varepsilon \Leftarrow \|a_t^2 + b_t^2 - c_t^2\|$ that codes the “nullspace proximity” to a singular point, and two additional variables:

- $\delta_{max} \in \mathbb{R}^+$, parameter that codes the maximum δ ;
- $d_\varepsilon \in \mathbb{R}^+$, parameter that codes the border proximity ε of a singular point vicinity;

The δ is then calculated according to the following quintic polynomial function,

$$\delta = \begin{cases} \delta_{max} \left[10 \left(\frac{\varepsilon - d_\varepsilon}{d_\varepsilon} \right)^3 - 15 \left(\frac{\varepsilon - d_\varepsilon}{d_\varepsilon} \right)^4 + 6 \left(\frac{\varepsilon - d_\varepsilon}{d_\varepsilon} \right)^5 \right], & \text{if } \varepsilon < d_\varepsilon \\ 0, & \text{if } \varepsilon \geq d_\varepsilon \end{cases}, \quad (6.47)$$

which generates the smooth transition curve (Figure 6.23), that allows the manipulator to gradually avoid

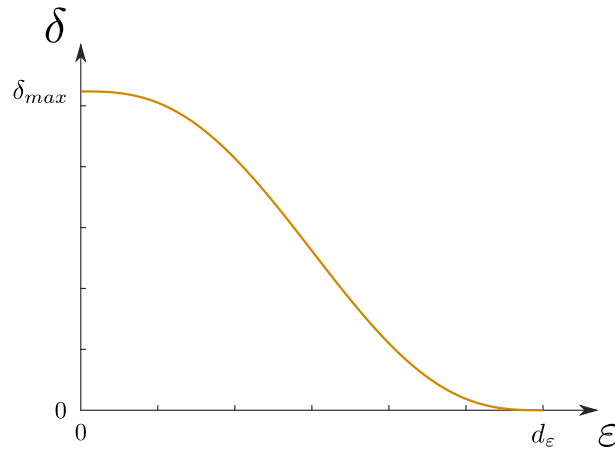


Figure 6.23: Singularity avoid interval as a function of the “nullspace proximity” parameter.

going near singularities.

6.2.6.4 Evaluating Poses

Up to this point, we were capable of mapping the limits and singular points of the robot mechanism to the nullspace of the inverted kinematic expressions. By doing so, we can take advantage of the robot's redundant structure to navigate through full-defined task space poses or trajectories. This method implements an

optimization function to solve extra degrees of redundancy, for example, whenever the task does not completely define the position and orientation.

The proposed pose evaluation method takes as inputs the pose variations relative to one or more redundant coordinate(s) (position and/or orientation), and grades the inputs based on a metric that involves the manipulability score and the distance to limits and singular points that will henceforth be named **modified manipulability** for simplicity, see (6.54). As for the applicability of the algorithm, consider the problem of selecting the manipulator pose to align with the linear electrode insertion trajectory (target to entry point). The axial symmetric property of the electrodes leads to a redundant orientation coordinate, around the axis that aligns with the trajectory.

Manipulability. Introduced by Yoshikawa (1985a,b) it directly relates to the “easiness of changing arbitrarily the position and orientation”. It is commonly chosen as a metric to evaluate the robot posture in the workspace to perform a given task. Based on the geometric Jacobian matrix ($\mathbf{J}(\mathbf{q})$ or simply $\mathbf{J}$), which maps the joint velocities to the end-effector linear and angular velocities, the original manipulability formula measures the distance to singular postures,

$$w_{std} = \sqrt{\det \mathbf{J} \mathbf{J}^T}. \quad (6.48)$$

The standard manipulability (w_{std}) can be rewritten as the product of the singular values of the Jacobian matrix found by Singular Value Decomposition (SVD). This analysis yields the singular vectors of the Jacobian matrix, which span the manipulability ellipsoids, a common graphical representation task space, Figure 6.24. As previously noted, anthropomorphic robotic manipulators are usually split into the upper arm

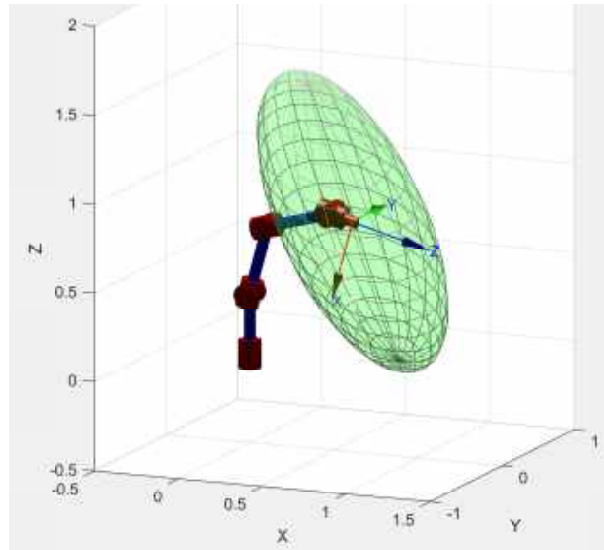


Figure 6.24: Manipulability ellipsoid of a KUKA LBR iiwa R800 kinematic structure at $\theta^c = [0.0, 15.0, 0.0, -60.0, 0.0, 30.0, 0.0]$ degrees. The shape of the ellipsoid represents the relation between the joint and end-effector velocity. It is more elongated along the axis where the same joint velocity translates to a higher end-effector velocity.

and the wrist, respectively associated with the translational and rotational parts. In 1991, Yoshikawa (1991) revised the total manipulability formula (6.48) to consider the translational and rotational manipulability

measures. To represent the new manipulability formula for general serial-link manipulators, which we apply in our algorithm, two new representations are important. First, consider the Jacobian matrix, which can be parted in the translational ($\mathbf{J}_P$) and rotational component ($\mathbf{J}_O$),

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix}. \quad (6.49)$$

Second, the inverted Jacobian matrix of any manipulator that is determined from the right pseudo-inverse ($\mathbf{J}^\dagger$),

$$\mathbf{J}^\dagger = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1}. \quad (6.50)$$

Now, the general case manipulability is determined as the product of the translational and rotational manipulabilities,

$$w = w_{P,s} w_{O,w}. \quad (6.51)$$

where,

$$w_{P,s} = \sqrt{\det \mathbf{J}_P (\mathbf{I} - \mathbf{J}_O^\dagger \mathbf{J}_O) \mathbf{J}_P^T} \quad (6.52)$$

$$w_{O,w} = \sqrt{\det \mathbf{J}_O \mathbf{J}_O^T}. \quad (6.53)$$

The translational manipulability factor ($w_{P,s}$) is determined as a “stricter” definition of the term, i.e. considering the rotational velocities input to be null ($\boldsymbol{\omega}_e = 0$).

The manipulability measure quantifies the proximity to singular postures but it does not account for mechanical joint limits (Vahrenkamp et al., 2012). Taking advantage of the method that maps these limits in the nullspace (refer to subsection 6.2.4) we were able to combine both the factors into a single metric, the modified manipulability (w_{alt}),

$$w_{alt} = w_g \left(1 - \left[e^{-\alpha(\psi - \psi_m^l)} + e^{-\alpha(\psi_m^u - \psi)} \right] \right). \quad (6.54)$$

With the classifier metric defined, one can evaluate and grade each input “redundant pose”. Similar to other redundancy resolution strategies presented, the received poses are evaluated per configuration, that is, solutions are only compared when they belong to the same GC . For each pose evaluated, the algorithm determines the feasible arm angle (ψ) interval. A spacing parameter is used to transform the feasible arm angle interval (see Figure 6.19), into a discrete set of ψ values. The modified manipulability is calculated for each arm angle of each pose tested. These scores are then processed and for each GC , the pose and arm angle that have the highest score are returned along with the respective score. The global configurations GC that have a feasible solution are ordered according to their respective scores as well.

The redundant poses may be evaluated for a specific or for all global configurations. Additionally, and as noted in subsection 6.2.5, arbitrary arm angle avoidance intervals may be added to further condition the feasible interval.

6.2.7 Statics Solver

The Statics Solver is the object responsible for determining the relationship between forces / torques applied to the end-effector and forces applied to the revolute joints. At the moment, the solver is simply tasked with the computation of the external forces and torques exerted on the robot's end-effector in task space.

Siciliano et al. (2009), used the principle of virtual work to demonstrate this relationship in mechanical manipulators at elementary displacements ($d\mathbf{q}$). The manipulator is at a state of static equilibrium when the elementary work (dW_τ) performed by the joint torques ($\boldsymbol{\tau}$) equals the elementary work performed by forces ($\mathbf{f}_e$) and torques ($\boldsymbol{\mu}_e$) actuating at the end-effector (dW_γ) with,

$$dW_\tau = \boldsymbol{\tau}^T d\mathbf{q} \quad (6.55)$$

and

$$dW_\gamma = \mathbf{f}_e^T d\mathbf{p}_e + \boldsymbol{\mu}_e^T d\boldsymbol{\omega}_e dt, \quad (6.56)$$

where $d\mathbf{p}_e$ stands for linear displacement and $d\boldsymbol{\omega}_e dt$ for angular displacement. According to the kineto-statics duality property, one can apply the geometric Jacobian matrix,

$$\mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad (6.57)$$

to reciprocally map the forces/torques in Cartesian space ($\boldsymbol{\gamma}_e$) to the calculate the joint torques,

$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q}) \boldsymbol{\gamma}_e \quad (6.58)$$

Inverting the equation yields,

$$\boldsymbol{\gamma}_e^T = \mathbf{J}^{-1}(\mathbf{q}) \boldsymbol{\tau}^T. \quad (6.59)$$

Due to the redundant nature of the manipulator, the Jacobian matrix can be inverted using the right pseudo-inverse. The torques measured at each joint motor (τ_m) are a combination of,

$$\tau_m = \tau_{grv} + \tau_{dyn} + \tau_{ext} + \tau_{err} \quad (6.60)$$

the torques originated from gravity (τ_{grv}), the torques originated from joint accelerations (τ_{dyn}), the torques resulting from external forces (τ_{ext}) and the torque disturbances caused by friction, measurement noise and modeling errors (τ_{err}). Knowing that the Robot FRI component provides the current joint positions as well as the current external joint torques, one can immediately compute the current external forces and torques applied to the end-effector.

6.3 Joint Trajectory Controller

The Joint Trajectory Controller is the component responsible for joint space trajectory planning (Figure 6.25). It builds on two libraries: the Reflexxes Type II (Kröger, 2010) (point-to-point - PTP Library) and a custom developed library to handle motion through a sequence of points (motion-through-points - MTP Library), each will be addressed below. This component generates smooth and time synchronous joint space trajectories within the specified velocity and acceleration limit values.

The Joint Trajectory Controller component communicates with the proxy component that connects to the real or virtual robot and operates in a real-time. The component has three basic operation modes: i) the PTP motion, where the robot receives the reference joint coordinates and executes a joint space motion; ii) the MTP, where the robot receives a vector of joint coordinates as well as the time law and interpolates between the points to execute a controlled robot motion in Cartesian space; iii) the position online trajectory generation or OTG motion, where the component continuously receives reference joint coordinates (at a lower sampling frequency) and interpolates between them at a higher frequency until reaching the reference point. The component only outputs the “Next Joint Positions” whenever a joint motion is being generated.

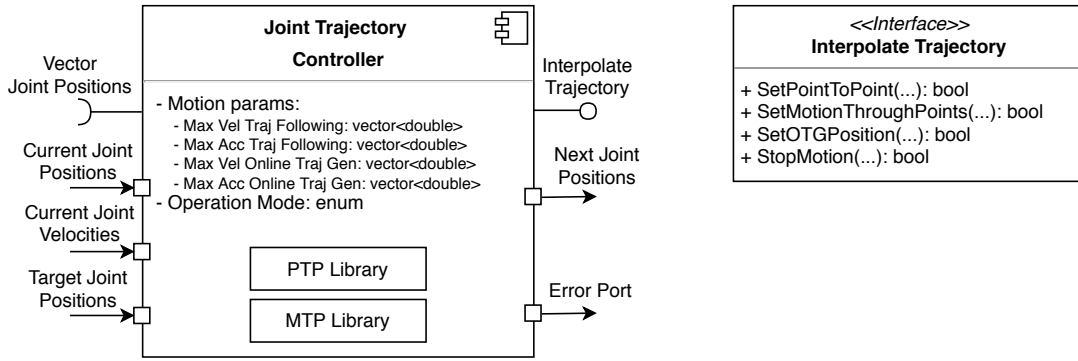


Figure 6.25: Joint Trajectory Controller component and interfaces.

6.3.1 Point-to-Point Library

The Point-to-Point motion in joint space is generated using the open-source Reflexxes Type II library. The Type II algorithm means that the position progression is described by polynomials up to the second order. It also guarantees a time-optimal and a possibility to set a time and phase synchronous joint trajectory. A phase synchronous joint trajectory translates to the generation of a homothetic trajectory, i.e. one-dimensional straight lines in multi-dimensional space (task space).

Intrinsic to the Type II algorithm is the concept of motion state. It is composed of the position ($\mathbf{P}_i$), velocity ($\mathbf{V}_i$) and acceleration ($\mathbf{A}_i$) values of one or more DoF at a specific time (T_i),

$$\mathbf{M}_i = (\mathbf{P}_i, \mathbf{V}_i, \mathbf{A}_i) \quad (6.61)$$

Adjacent to the concept of motion state, the kinematic motion constraints ($\mathbf{B}_i$) that define the maximum velocity ($\mathbf{V}^{max}$) and maximum acceleration ($\mathbf{A}^{max}$) constant values, are also specified.

$$\mathbf{B}_i = (\mathbf{V}_i^{max}, \mathbf{A}_i^{max}) \quad (6.62)$$

The algorithm is labeled memory-less due to the fact that it only generates the next motion state, i.e the motion state at T_{i+1} . It does so by taking the **input parameters** i) the current motion state, ii) the target motion state, and iii) the kinematic motion constraints, Figure 6.26.

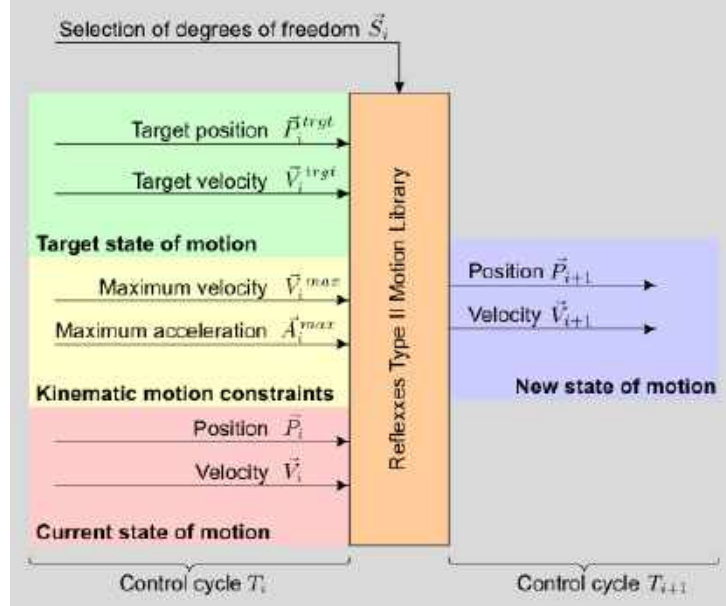


Figure 6.26: Type II Joint Trajectory Generation Algorithm with the **input parameters** at the left side (green, yellow and red) and the **output parameters** at the right side (blue).

The type II algorithm is structured in three distinct steps: i) the calculation of the synchronization time (t^{sync}), ii) the synchronization of the DoF and iii) calculation of output values.

In the first step and for each DoF, the algorithm determines the velocity profile that allows the actuator to move from the initial motion state ($\mathbf{M}_i^{ini}$) to the target motion state ($\mathbf{M}_i^{tar}$). This profile is selected from a finite set of possible profiles, by following a decision tree with boolean conditions related to the **input parameters**. With the profile determined, the minimum trajectory time for the DoF (t_i^{sync}) as well as the quadratic polynomial coefficients that code the DoF position progression can be calculated by solving a system of nonlinear equations. In order to generate a time-synchronous motion with a multiple DoF system, the trajectory times for each DoF must match. In type II algorithms, there may be a single inoperative time interval, for $t > t_i^{sync}$, which specifies an interval in time that the joint trajectory cannot reach the target motion state with the current **input parameters**. Thus, the joint trajectory synchronous time (t^{sync}) is, in fact, the minimum possible joint trajectory time for all DoF.

The second step initiates with new motion profiles being calculated for the remaining DoF. Between the different types of motion profiles, there is an infinite number of solutions. To keep the algorithm deterministic, the motion profile with the minimum integral of the acceleration module ($\int_{T_i}^{t^{sync}} |a_i(t)| dt$) is selected. Once all DoF velocity profiles are set, one can determine the polynomials that code the quadratic position progression likewise to first step.

The third and final step consists simply in solving the polynomial equations as a function of the time to obtain the joint position progression.

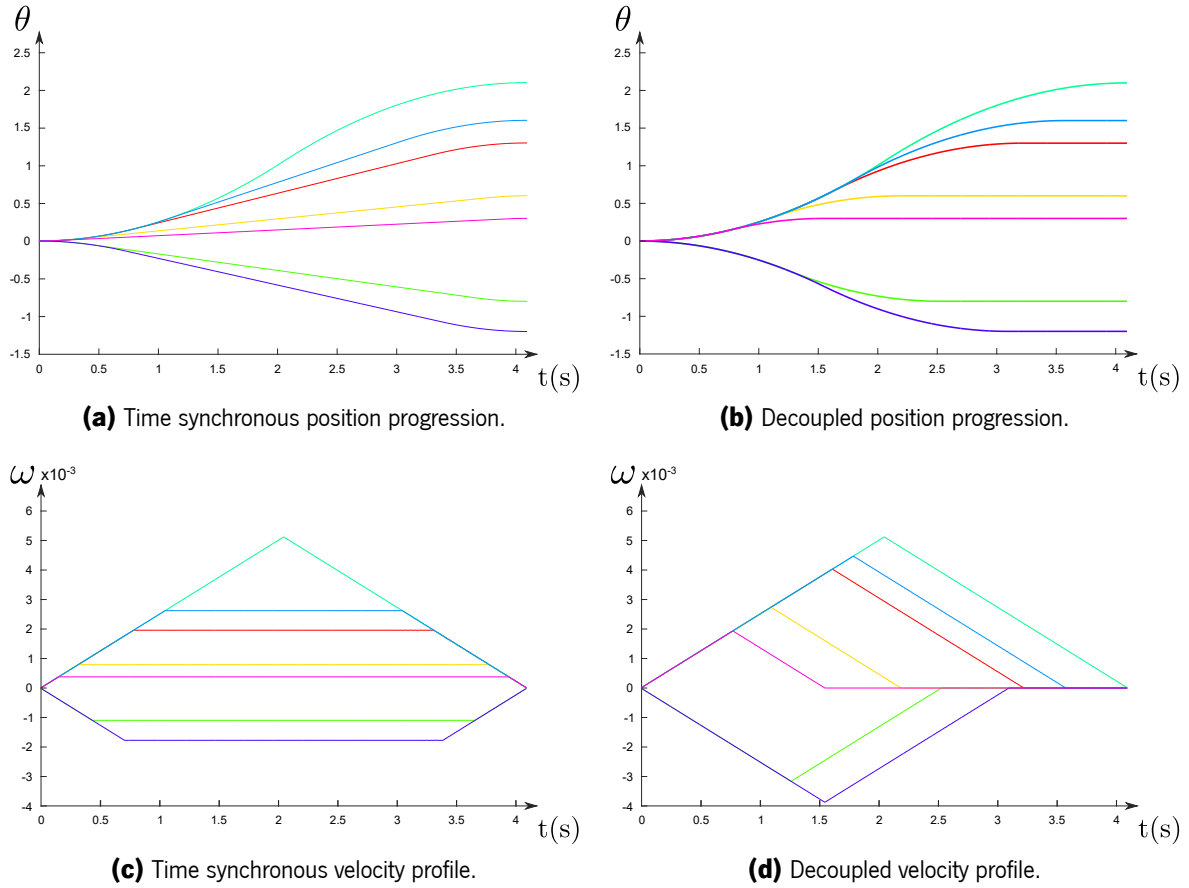


Figure 6.27: Effect of the time synchronous property exemplified with the position and velocity progression of the shoulder joints (θ_0 , θ_1 , and θ_2).

In conclusion, the type II joint trajectory generation algorithm is robust, deterministic and real-time capable, complying with the requirements defined for our robot controller architecture (refer to section 5.4). The available interface allows the user to set up a safety mechanism to handle unforeseen events within a single time step. At the moment, and whenever an error event is received, the “stop-motion” joint trajectory is generated to smoothly halt the robot motion (without violating the kinematic motion constraints).

Two foreseeable problems with the current implementation, have to do with the lack of control over the target acceleration as well as the lack of continuous acceleration profile. The lack of a continuous acceleration profile shortens the lifetimes of mechanical systems and leads to worse trajectory following behavior by enhancing the effects of the system’s structural natural frequencies. The Reflexxes Type IV algorithm includes the jerk-limitation feature and an enhanced API to permit the execution of a motion-through-points. However, this version of the library had a commercial paid license and is currently inaccessible since the company was acquired by Google Inc.

The PTP library is responsible for generating the robot motion with reference to a set of target joint coordinates (PTP) as well as to handle and continuously drive the robot arm in online trajectory following actions (OTG). Both control modes rely on the type II algorithm. A wrapper class was developed on top

of the Reflexxes library to facilitate the integration with the OROCOS component and to create distinct interfaces for each control modes. As the wrapper object is instantiated in the component configuration step, several parameters are passed including the number of joints to control, the time step and the kinematic motion constraints ($\mathbf{B}_i$). In this construction phase (non-real time), the members of the wrapper class are initialized and the algorithm flags are set (e.g. time synchronization, handling extreme motion states).

After construction, the PTP library can work in a real-time loop. Two distinct phases are involved in motion generation: i) initialization - where the **input parameters** are set, which translates to the steps 1 and 2 of the type II algorithm, ii) generation - where next motion states are generated at T_{i+1} , the 3rd step.

In the case of a PTP motion, the action is initiated by passing the **input parameters** that drive the robot towards the target motion state. It reaches the reference joint positions with null velocity in all joints unless an unforeseen event occurs that forces the robot to stop midway through the trajectory. The PTP motion generation method is called at every Joint Trajectory Component update cycle after initiation and until motion conclusion. In normal operation, it returns either a Working or a Final State Reached result. If an error code is returned instead, it is handled directly by the component that forwards it to the Task Supervisor.

Alternatively, if an OTG motion is requested, a new motion is initiated whenever new target joint positions are received at the component port (at a frequency that matches the update cycle of the Kinematics and Statics Module component). The target motion state position is composed of the reference positions received and a null joint velocity. Having a faster update cycle means that the component is capable of smoothly interpolating to the target joint positions received. If new target joint positions are continuously received, new motion states are initiated even though the last motion might not have reached the final state. This results in a trajectory blend from the previous to the current trajectory segment, creating a continuous motion flow.

6.3.2 Motion-through-Points Library

The Reflexxes Type II offers a limited API, not suited to execute motion-through-points sequences. To overcome this shortcoming and address the lack of a continuous acceleration profile we developed a software library capable of handling these cases.

As referred at the beginning of the current section, the component can operate in 3 different modes depending on the required joint motion. To execute an MTP, the component receives N sets of target joint positions and the correspondent times through the Vector Joint Positions interface. These vectors represent an end-effector trajectory initially defined in task space and translated to joint space. They follow a specific geometric path and velocity profile (refer to section 6.1).

The problem of generating a motion profile by moving through a sequence of points defined in time and space is well-known in robotics (Craig, 2005; Siciliano et al., 2009). It boils down to an interpolation problem, one which can be approached from different angles. The most common methods involve:

1. using a Lagrange interpolating polynomial function of order $N - 1$ that fits the N path points;

2. using linear functions to connect path points with parabolic blends;
3. using low-order polynomials functions at path points.

The first option has several problems. Since the order of the polynomial grows in a 1-to-1 ratio with additional path points, one might easily have to deal with a high order polynomial and all the associated drawbacks (increasing oscillatory behavior, numerical inaccuracies, and heavy computation). It is also not possible to set the initial and final trajectory velocities for a $N - 1$ polynomial.

As for the second option, using linear functions with parabolic blends to avoid velocity discontinuity at path points means that the trajectory does not necessarily pass through the path point. Also, the trajectory does not guarantee second derivative continuity.

In light of these limitations, we chose to implement an interpolation algorithm based on the third option. The cubic polynomials are the lowest polynomial order that guarantees a continuous second derivative ($\ddot{q}_k(t)$).

Consider the one dimensional problem with a set of path points ($q_k, \forall k \in \{0, \dots, N - 1\}$) and the associated times (t_k). The function of the final interpolated path ($\Pi(t)$) branches to a $N - 1$ number of cubic splines,

$$\Pi_k(t) = a_k + b_k(t - t_k) + c_k(t - t_k)^2 + d_k(t - t_k)^3, \forall k \in \{0, \dots, N - 2\}. \quad (6.63)$$

Thus, the interpolated path can be calculated by computing the polynomial coefficients for each path segment (a_k, b_k, c_k and d_k). Several conditions are set beforehand, such as:

- 1) the path with pass through the path points,

$$\Pi_k(t_k) = q_k, \quad (6.64)$$

- 2) the path will be continuous up to the second derivative,

$$\Pi_{k-1}(t_k) = \Pi_k(t_k), \quad (6.65)$$

$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k), \quad (6.66)$$

$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k), \quad (6.67)$$

- and 3) the path will be a clamped⁷ and natural cubic spline,

$$\ddot{\Pi}_0(t_0) = \ddot{\Pi}_{N-1}(t_{N-1}) = 0. \quad (6.68)$$

Let $\Delta t_k = t_{k+1} - t_k$. Based on these conditions and on (6.63), one can immediately determine a_k and a_{k+1} ,

$$a_k = \Pi_k(t_k), \quad (6.69)$$

$$a_{k+1} = a_k + b_k \Delta t_k + c_k \Delta t_k^2 + d_k \Delta t_k^3. \quad (6.70)$$

⁷ $\ddot{\Pi}_0(t_0)$ and $\ddot{\Pi}_{N-1}(t_{N-1})$ are given

On a similar note, the first derivative of (6.63) yields b_k and b_{k+1} ,

$$b_k = \dot{\Pi}_k(t_k), \quad (6.71)$$

$$b_{k+1} = b_k + 2c_k \Delta t_k + 3d_k \Delta t_k^2. \quad (6.72)$$

Repeating the process for the second derivative,

$$c_k = \frac{\ddot{\Pi}_k(t_k)}{2}, \quad (6.73)$$

$$c_{k+1} = c_k + 3d_k \Delta t_k. \quad (6.74)$$

Solving (6.74) as function of d_k and substituting into (6.70) and (6.72) yields,

$$a_{k+1} = a_k + b_k \Delta t_k + \frac{\Delta t_k^2}{3} (2c_k + c_{k+1}), \quad (6.75)$$

$$b_{k+1} = b_k + \Delta t_k (c_k + c_{k+1}). \quad (6.76)$$

Solving (6.75) as a function of b_{k+1} and replacing the k indices for $k - 1$ one obtains,

$$b_{k-1} = \frac{1}{\Delta t_{k-1}} (a_k - a_{k-1}) - \frac{\Delta t_{k-1}}{3} (2c_{k-1} + c_k). \quad (6.77)$$

The same index operation can be performed in (6.76),

$$b_k = b_{k-1} + \Delta t_{k-1} (c_{k-1} + c_k). \quad (6.78)$$

Substituting the two previous equations (6.77) and (6.78), one ends up with a equation involving the unknowns c_{k-1} , c_k and c_{k+1} , and the known coefficients for a_{k-1} , a , a_{k+1} , Δt_{k-1} and Δt_k ,

$$\Delta t_{k-1} c_{k-1} + 2c_k (\Delta t_{k-1} + \Delta t_k) + \Delta t_k c_{k+1} = \frac{3}{\Delta t_k} (a_{k+1} - a_k) - \frac{3}{\Delta t_{k-1}} (a_k - a_{k-1}). \quad (6.79)$$

At this point we have a system of $N - 2$ equations and $N - 2$ unknowns. Passing the system of linear equations to a matrix representation ($\mathbf{Ax} = \mathbf{b}$),

$$\mathbf{A} = \begin{bmatrix} \Delta t_0 & 2(\Delta t_0 + \Delta t_1) & \Delta t_1 & 0 & \dots & 0 \\ 0 & \Delta t_1 & 2(\Delta t_1 + \Delta t_2) & \Delta t_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \Delta t_{N-2} & 2(\Delta t_{N-2} + \Delta t_{N-1}) & \Delta t_{N-1} \end{bmatrix} \quad (6.80)$$

and

$$\mathbf{b} = \begin{bmatrix} \frac{3}{\Delta t_1}(a_2 - a_1) - \frac{3}{\Delta t_0}(a_1 - a_0) \\ \frac{3}{\Delta t_2}(a_3 - a_2) - \frac{3}{\Delta t_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{\Delta t_{N-1}}(a_N - a_{N-1}) - \frac{3}{\Delta t_{N-2}}(a_{N-1} - a_{N-2}) \end{bmatrix}. \quad (6.81)$$

Knowing beforehand (6.68), we solved the system of equations above using a QR decomposition method to calculate c_k or $\ddot{\Pi}_k(t_k)$, $\forall k \in \{1, \dots, N-2\}$. From here it is straightforward to obtain the remaining coefficients from (6.70), (6.72) and (6.74).

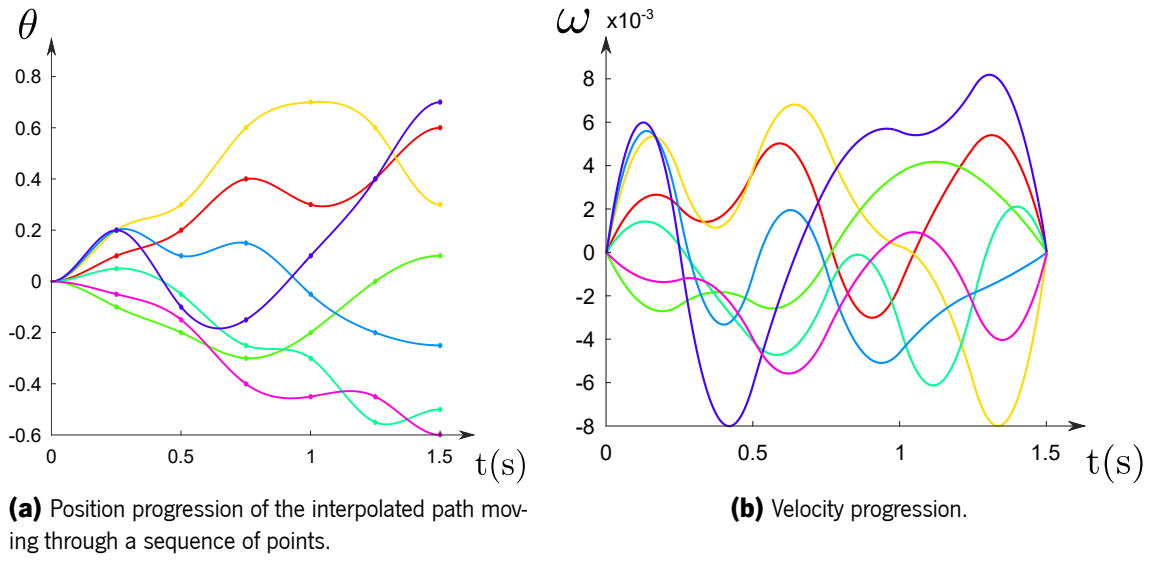


Figure 6.28: Example of the shoulder joints (θ_0 , θ_1 , and θ_2) following a trajectory generated from a sequence of points using the MTP library.

6.4 Robot State and Transformations

The Robot State and Transformations components are considered structural components to the architecture since they are mostly responsible for information storage and broadcast. The Robot State component is, as previously stated, a “server unit” that can communicate to other components in architecture or broadcast to other distributed components (Figure 6.29). It operates in real-time at the highest frequency matching the components that forward the interpolated joint positions.

The Robot State component receives the current joint positions and velocities from the Robot FRI and V-REP FRI components and broadcasts this information to all other components that require it. Additionally, the component includes the Kinematics Solver library so it can compute at each iteration the end-effector frame relative to the robot base as well as its arm angle and configuration. Finally, the robot state also broadcasts if the robot is in motion.

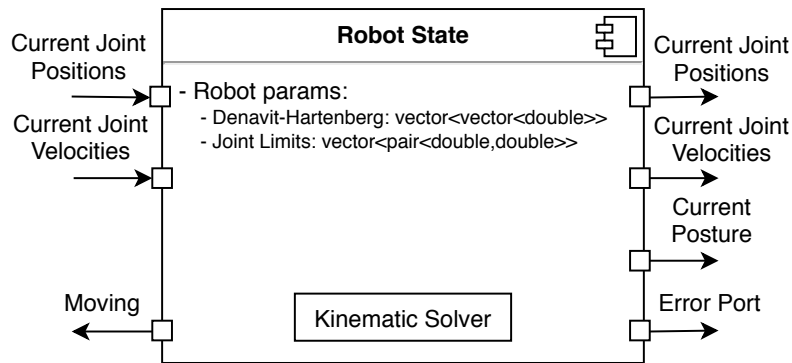


Figure 6.29: Robot State component.

The Transformations component stores two types of transformations, one that describes the tool frame relative to the robot's flange, and another that describes the work frame relative to the robot's base (Figure 6.30).

The "Tools" handled by the system are listed in a property file that stores their position and rotation relative to the flange. When a tool is attached to the robot, its transformation is loaded and the system recognizes a new end-effector frame. If the tool has a moving element such as the electrode or trepan, their translational axis is set to coincide with the tool frame z-axis and their position set to the moving element tip at the default position. Thus, the final position and orientation of the moving element can be calculated at any moment just by translating its increment along the tool's z-axis. At the moment, the system lists 4 types of end-effector tools: the empty tool interface, the interface holding the probe, the interface holding the trepan and a pointer element used for the registration step.

The "References" specify the frames about which the robot is to move. In other words, it codes the transformation of this reference frame relative to the robot's base frame, thus indicating where the task space trajectories are described. The user may specify different reference frames such as: i) base, ii) flange/tool and iii) work. Unlike the "Tools" transformations, these are not constant. For example, if the robot is to execute a linear motion of 100 mm along the x-axis of the work reference frame, the reference frame is set to the work frame and the tool frame set to whichever tool is currently attached. The trajectory coordinates are then transformed from the work's frame to the robot's base and the tool's frame to the flange so that the Cartesian Trajectory Controller can generate the end-effector motion relative to the robot base.

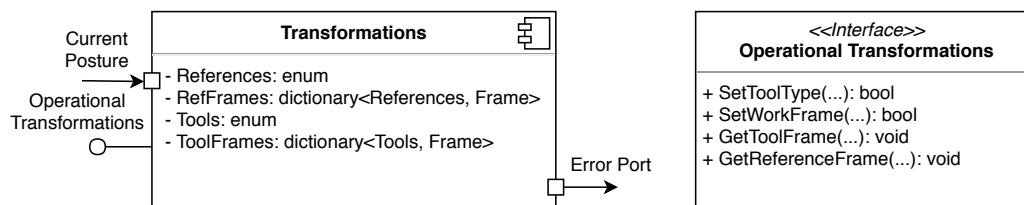


Figure 6.30: Transformations component.

6.5 Task Supervisor

Analogous to the central nervous system, the Task Supervisor component binds together the “Core” layer. It guarantees that each task is performed by the assigned components, that each component receives the required parameters so that the robot can perform the requested action. The component is also the “Core” layer connection to the outside world, providing and requiring interfaces to communicate with the “Application” layer or with other external components.

The Task Supervisor is also the component accountable for the peripheral components’ states and error handling (Figure 6.31). From a central management position, this component takes part in most of the decision-making to guarantee that all required conditions are met prior to carrying out the action, and that faulty states are properly handled to avoid compromising the system or the user’s safety.

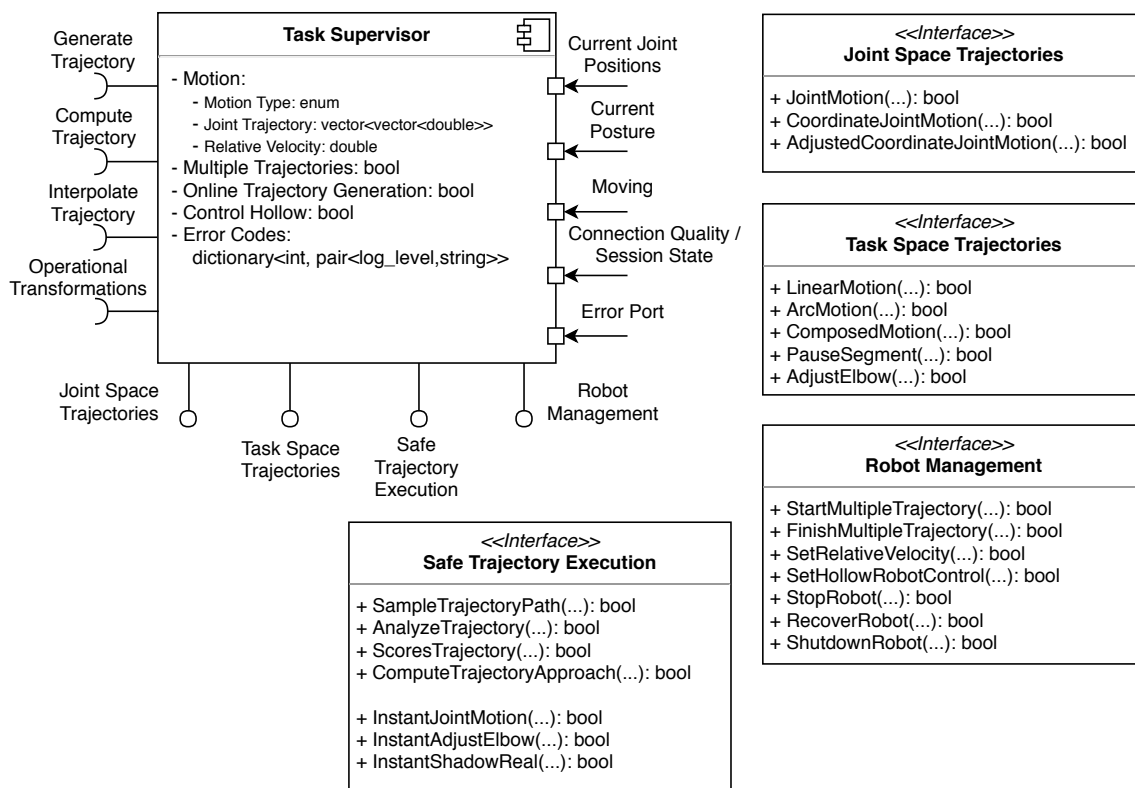


Figure 6.31: Task Supervisor component.

6.5.1 Action Interface

Any action required of the controller involves a coordinated effort of two or more peer components. This action is characterized by the properties and attributes loaded that span globally to every instruction and the instruction’s parameters. To guarantee a reliable and stable operation a set of properties and attributes related to the joint physical limits, nullspace navigation, task space constraints, tool properties, etc, are statically defined and loaded during the configuration phase (refer to previous components). These properties impose a number of constraints that affect every action performed by the component. Let Action Interface be the combination of Joint Space Trajectories and Task Space Trajectories interfaces. These

interfaces were designed such that the caller has an explicit control over all the variables aside from those pre-defined in the property files. It is important to note that no trajectory can be initiated if the robot is in motion or has other motions queued⁸.

Joint Space Trajectories refer to any robot action performed directly in the joint space, i.e. each robot joint moves synchronously along the shortest path from the starting to the reference point disregarding the robot body displacement in task space.

The JointMotion method displaces the robot from the current joint positions to the specified joint positions, a PTP motion (refer to subsection 6.3.1). The user specifies the target joint positions as well as the relative velocity (a factor that scales the motion execution maximum velocity). If the motion is possible according to the joint physical limits, a request is forwarded directly to the Joint Trajectory Controller's Interpolate Trajectory interface, Figure 6.25. The CoordinateJointMotion and AdjustedCoordinateJointMotion methods require a target frame to be reached by the robot's end-effector. The caller also needs to specify the reference frame where the target frame is defined as well as the currently attached tool. The CoordinateJointMotion method requires the target arm angle as well as the global configuration along with the final frame. On the other hand, the arm angle and global configuration parameters for the AdjustedCoordinateJointMotion are calculated using the redundancy resolution method introduced in subsection 6.2.5. Opposed to the JointMotion method, these involve the services of the Operational Transformations interface (Figure 6.30) to retrieve the related reference and tool frames, as well as the Kinematics and Statics Module's Kinematics interface to determine the correct set of joint positions. Then, if the motion is feasible, this set of joint positions is then forwarded to the Joint Trajectory Controller component.

Task Space Trajectories permit the caller to specify the desired end-effector trajectory in task space. They are initially generated using the Cartesian Trajectory Controller's Generate Trajectory interface (Figure 6.1), which computes a vector of end-effector frames discrete in space and their respective times, refer to section 6.1. After the task space trajectory is calculated, the vector of frames is forwarded to the Kinematics and Statics Modules's Compute Trajectory interface. If feasible, each frame is converted into a set of joint coordinates, which is then sent to the Joint Trajectory Controller along with the trajectory way-point times. These vectors are the input variables to generate and execute a Motion-Through-Points (subsection 6.3.2).

Across the different Task Space Trajectories methods, all require a frame of reference as well as the type of tool attached to its end-effector and, unless the robot is performing a multiple trajectory sequence, the requested motion always starts from the current robot pose. The interfaces for the LinearMotion, ArcMotion, and ComposedMotion mirror those of the Cartesian Trajectory Controller's Generate Trajectory (subsection 6.1.1). If the robot is planning a multiple trajectory, the user may add a pause segment where, for a limited time interval, the robot remains fixed in its pose. The AdjustElbow method will be addressed in subsection 6.5.5.

⁸The Task Supervisor allows queuing more than one trajectory only if the multiple trajectory mode is enabled.

6.5.2 Robot Management

The `StartMultipleTrajectory` and `FinishMultipleTrajectory` methods allow the component to enter into a multiple trajectory mode and accept one or more joint/task space trajectories. In multiple trajectory mode, the trajectories must share a common reference frame and tool type. If each trajectory is feasible, then the motion proceeds sequentially. Do not confuse with the `ComposedMotion` method that accepts a vector of poses and generates a linear interpolation path with parabolic blends at each way-point.

The `SetRelativeVelocity` method, as the name implies, allows the caller to set the robot's relative velocity scaling factor. If the controller is connected to the V-REP simulator, one may use the `SetHollowRobotControl` to preview the robotic manipulator motion or test its final posture. This command allows the user to control the pre-visualization 'hollow' robot using exactly the same control scheme used for the real platform.

When the `StopRobot` method is called the command is forwarded to the Joint Trajectory Controller that halts the transmission of 'Next Joint Positions'. In parallel, the queued trajectories are dropped and the multiple trajectory mode disabled if active. When the controller is in this stopped state, the robot is not allowed to move until a call is made to the method `RecoverRobot`. From this point, the controller resumes its normal operation. If the `ShutdownRobot` method is called instead, the robot halts its motion and the system terminates its operation. From a shutdown state, the robot cannot be recovered, requiring a complete boot.

6.5.3 Safe Trajectory Execution

Whenever a coordinated joint motion or a task space trajectory are requested from the controller, the action is verified at the Kinematics and Statics Module for feasibility. If the movement falls within the subspace of possible and determined solutions, the system generates the joint space trajectory obeying the redundancy resolution scheme proposed in subsection 6.2.5. Otherwise, either the trajectory is feasible from another set of redundancy parameters and the controller prompts the user to alter the arm angle or global configuration; or the trajectory is simply not feasible from any configuration, Figure 6.32.

The Safe Trajectory Execution interface extends this functionality by also providing methods to verify the trajectory feasibility prior to its execution. In stereotactic neurosurgery, it is common to have a list of pre-defined trajectories to guide medical instrumentation. Thus, the methods to sample, analyze, and score the trajectory in terms of the nullspace parameters allow the user to pre-position the robot in a configuration that guarantees the complete trajectory execution away from the mechanical limits and singularities.

The `SampleTrajectoryPath` allows the user to sample the geometric task space path to be followed by the end-effector into frames separated in regular, pre-defined steps.

When the `AnalyzeTrajectory` is called, the list of frames previously computed are sent to the Compute Trajectory interface from the Kinematics and Statics Module where the feasible redundancy intervals are calculated for each frame. The interval avoidance strategy is applied for the trajectory intervals and the initial/feasible arm angle intervals are calculated for all configurations (subsection 6.2.6 - Interval Avoidance Strategy).

The `ScoresTrajectory` method takes as input the feasible arm angle intervals of each global configuration for the analyzed trajectory and delivers a score to the available nullspace intervals based on the distance to

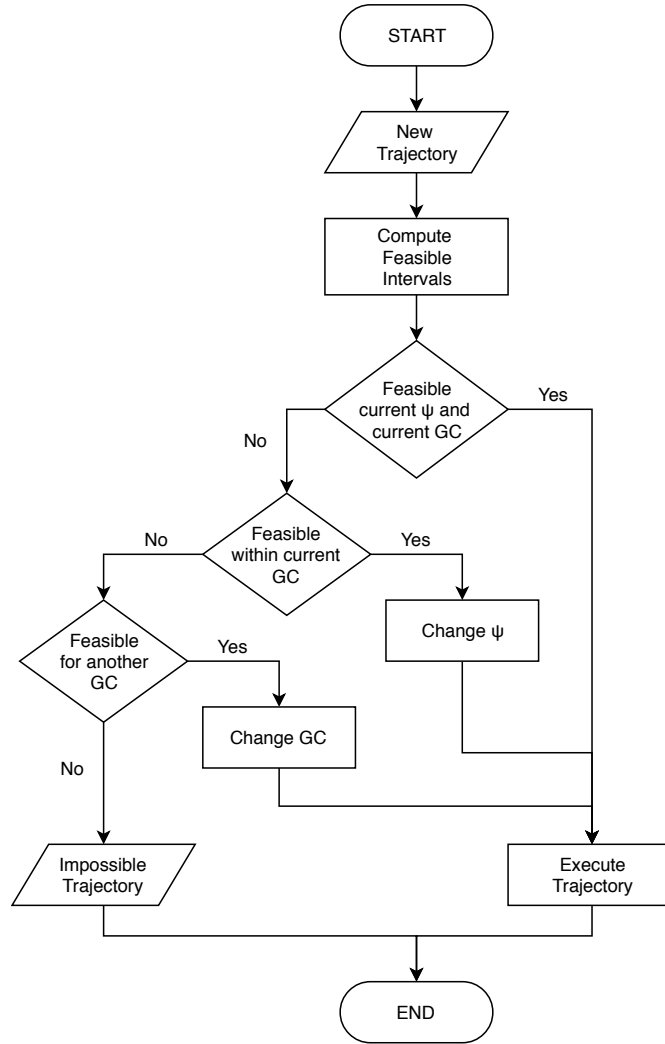


Figure 6.32: Decision process to guarantee a feasible trajectory.

the limits and the proximity to the current pose (subsection 6.2.6 - Nullspace Scoring).

The `ComputeTrajectoryApproach` method takes as input two points and a redundancy sampling step. The objective is to determine the best fitting full-defined pose (including the redundancy parameters GC and ψ) that aligns the end-effector z-axis (z_e) with the two input points vector. More about this functionality in the following subsection.

Finally, the instant motion methods provide visual feedback of the desired poses so the caller can instantly preview the final robot pose in the simulation environment. Contrary to the hollow robot control mode, which emulates the movement of the real one, the instant motion methods instantly position the robot at the final location: the `InstantJointMotion` at the target joint positions, the `InstantAdjustElbow` at the specified nullspace coordinates keeping the end-effector pose and the `InstantShadowReal` to the pose of the real robot.

6.5.4 Trajectory Approach

The Trajectory Approach feature was added to assist in the determination of the best pose to reach a given trajectory, in this case, defined by two points in task space. Since we have an unconstrained orientation variable, the manipulator becomes even more redundant with two non-defined task space variables, the nullspace. A new nullspace resolution strategy is proposed to address both degrees of redundancy, the undefined orientation coordinate and the intrinsic redundancy (GC and ψ).

First, nullspace's two-dimensional domain is sampled. Consider the points ($\mathbf{p}_1$ and $\mathbf{p}_2$) to be position vectors in task space relative to the robot base reference frame ($\{B\}$), and the sampling step, (κ), a scalar to sample the redundancy orientation interval into a discrete set, Figure 6.33. The set of “redundant poses”

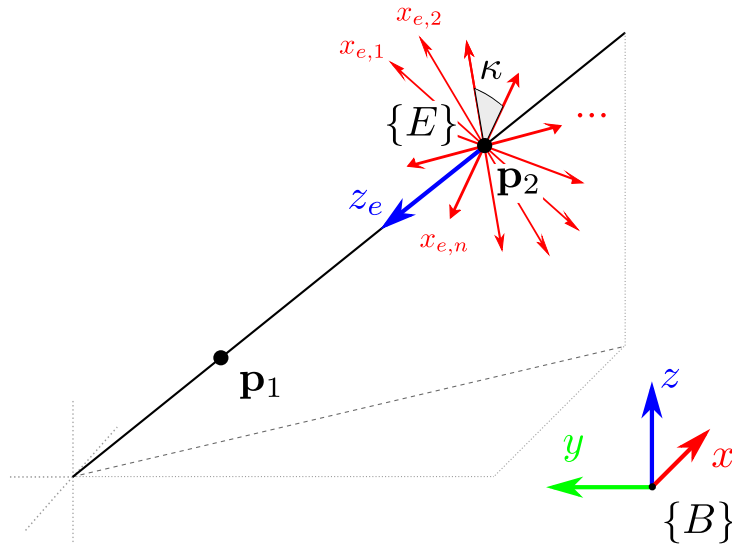


Figure 6.33: Set of poses that align the end-effector frame ($\{E\}$) z-axis with the vector created from the two input points ($\mathbf{p}_1$ and $\mathbf{p}_2$). The number of redundant poses n depend on the sampling step (κ). For clarity, only the z- and x-axis of the test poses are represented in the picture.

are forwarded to the Kinematics and Statics Module, Compute Trajectory interface and analyzed using the EvaluatePosesManipulability() method (refer to subsection 6.2.6.4). The subset of feasible nullspace solutions (see Figure 6.19) is determined for each pose. The solutions are grouped by global configuration branches. For each GC that contemplates feasible solutions, it is returned the pose and arm angle with the highest **modified manipulability**.

To guarantee that robot can move along the two-point trajectory from the initial approach pose without driving into a joint limit or singularity, further analysis is conducted. With the linear trajectory defined with the two points ($\mathbf{p}_1$ and $\mathbf{p}_2$) and the initial pose specified, the AnalyzeTrajectory and ScoresTrajectory methods are applied to determine first, if the trajectory is feasible from the initial pose, and second to score the feasible arm angle intervals.

The output of the Trajectory Approach comprises, for each GC branch with feasible solutions, the highest **modified manipulability** pose and arm angle and a score vector (refer to subsection 6.2.6.2).

6.5.5 Nullspace transitions

The possibility to adjust the robot arm angle and global configuration to guarantee the trajectory feasibility comprehends the transitions between the nullspace parameters, arm angle (ψ) and global configurations (GC), Figure 6.32, which involve two different sets of movements.

Adjusting the arm angle means to move in the elbow position without changing the end-effector pose or the global configuration. Unlike global configuration changes, the movement to change the arm angle can be entirely performed within the nullspace, if the nullspace path falls within single feasible interval. The motion is generated at the Cartesian Trajectory Controller by calling the `GenNullspaceTrajectory` method. While the other task space trajectory methods generate a vector of poses and times, the nullspace trajectory generates a vector of arm angle positions and the respective times since the current frame is maintained throughout the movement. A velocity profile similar to the one that describes the task space path is parametrized and associated with the “nullspace path”. It is then solved the inverse kinematics for each pose and arm angle pair, and the resulting joint space coordinates sent to the Joint Trajectory Controller as with any other trajectory.

If the redundancy parameter transition involves a global configuration change or an arm angle shift between two separated feasible intervals within the same configuration, the robotic manipulator is initially moved to an intermediate posture before reaching the final values. Generating a simple point-to-point motion in joint space to change the global configuration would potentially lead to ample and unpredictable movements, whereas shifting between feasible intervals involves passing near a joint limit or singularity. To cope with this situation, we separated such adjustments into three sequential sub-movements:

1. zeroing motion of pivot joints;
2. point-to-point motion of hinge joints to their final position;
3. point-to-point motion of pivot joints to their final position.

The robot starts by moving to a ‘twisted zero’ position by rotating just the pivot joints. At this point, the robot’s structure is straight and aligned normal to its base. In this position, the hinge movement causes the minimum displacement in the robot structure. The final point-to-point motion of the pivot joints causes the robot to finally reach the target global configuration with the same end-effector pose. This sequence guarantees that the transition between global configurations is predictable and relatively contained in the workspace.

6.5.6 Motion Queuing and Error Handler

The Task Supervisor is the component responsible for handling robot motion execution, which involves a combined effort from other components in the ‘Core’ layer. Additionally, the component handles external requests and error codes reported from peer components. To guarantee timely responses and deterministic behavior, the component operates in real-time.

The interface between the Task Supervisor and peer components can be divided into properties and attributes, ports and operations. Setting properties and attributes as well as communicating through

ports are real-time and thread-safe as long as the data transport is properly set up. Alternatively, calling and sending operations may compromise the real-time requirement since it implies memory allocation to collect the return values. Most of the memory allocation takes place during the configuration step, and the remaining variables are handled by the deployer's real-time memory allocator. On the other hand, calling operations blocks the Task Supervisor own thread of execution, which may cause it to fail the real-time constraint. Some of the operations requested by the Task Supervisor take a finite but undetermined amount of time considering the variable computational cycles that depend, for example, on the type and length of the trajectory generated.

As previously explained, the process of generating a robot movement involves a sequential execution of different components methods. However, the synchronization cannot be guaranteed by sequentially calling these methods, due to the risk of violating the real-time constraint. This situation is solved by implementing a finite-state-machine where each state relates to the different phases of motion generation, and by sending/collecting operations instead of making a blocking call.

If no motion is being requested the Task Supervisor can be in the monitoring or in the motion ready state. From its motion ready state and depending on the requested action it can transition to: i) an online trajectory generation mode which requires a continuous stream of robot poses or joint values, ii) a task space trajectory that involves the generation of the Cartesian trajectory, computing and interpolating the joint trajectory, and iii) a joint space trajectory execution (moving state), Figure 6.34.

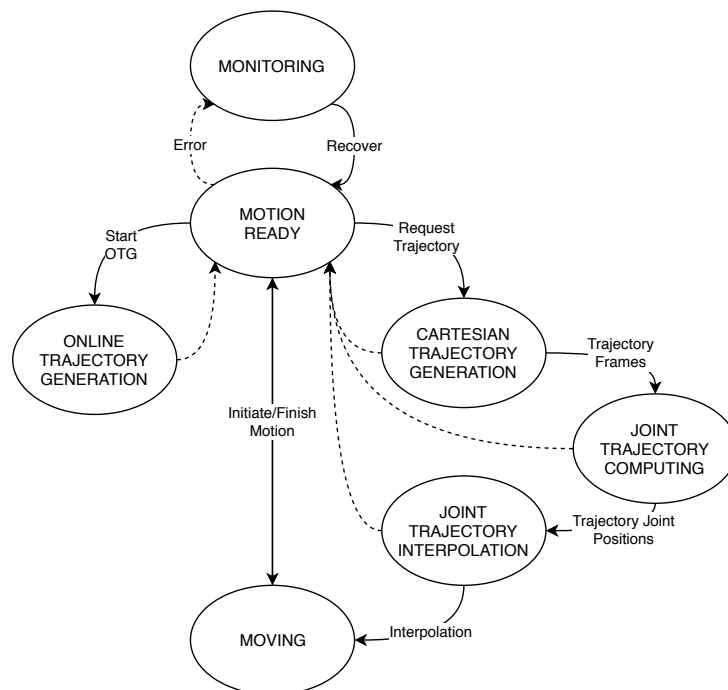


Figure 6.34: Task Supervisor motion state machine.

Motion requests are only accepted if the Task Supervisor is in the motion ready state. When a new motion is accepted, instead of issuing calls to peer components the Task Supervisor sends a method that is executed by the ExecutionEngine of the receiving component. Once per update cycle, the Task Supervisor component checks if the process is concluded to then collect the result and transition to the next state.

In case the robot is performing a multiple trajectory, the movements are processed in serial mode until the joint trajectories for each movement are determined. Once the multiple trajectory mode is terminated, the queued joint trajectories are sequentially executed by the Joint Trajectory Controller component.

The component is also responsible for handling error codes and faulty behavior. The error codes received from peer components include 3 parameters, a log level, an error unique code, and an error message. The “error log level” falls under one of the categories: i) Information, ii) Warning, iii) Error and iv) Critical.

The information level includes messages to notify the user of normal functioning events, successful process results, or correct connection with external devices/components. The warning level messages notify the user of events that do not compromise the robot operation but that require user-consent. Examples of warning error codes include events that require the user to adjust the arm pose to execute a trajectory, or to alert the user to create a work frame transformation prior to using it as a reference frame. The error level comprises messages related to exceptions or to errors occurred in peer components. Examples of error level messages include errors related to 3rd party libraries (KDL or Reflexxes), mismatch between expected and received parameters, mechanical limits violations, motions requested outside the manipulator’s workspace, etc. The critical level messages are reserved for hardware non-recoverable errors and communication failure within and outside the “Core layer” components.

Each error message is uniquely identified by a key code so that the system can properly react to each specific event or exception. Moreover, the Task Supervisor component is configurable to report different levels of error messages. For example, if the log level report is defined as warning level only warning and higher level messages are reported.

6.6 Fast Research Interface

The Fast Research Interface (FRI) components are the architecture components that interface with the robot controller hardware or the virtual robot controller. The FRI components were implemented to guarantee a transparent and seamless transition between controlling the real robotic platform or the virtual model (V-REP simulator). It is also possible to simultaneously control both platforms, to have the virtual robot acting as a pre-visualization tool to plan the real robot movements.

6.6.1 Robot FRI

As aforementioned in subsection 5.6.1, the FRI communication channel is explored to drive the robotic arm in real-time. Operating in a client-server typology, the client is typically a C++ application built on top on the provided SDK to communicate with the robot application running in the server side.

Instead of a stand-alone application, we developed an OROCOS component around the FRI client to smooth the hardware integration (Figure 6.35). In the configuration step, the connection IP addresses and ports of the local host and remote controller matching the server side configuration are marshaled from a property file along with data relative to the robotic joint limits. During the configuration step, the client

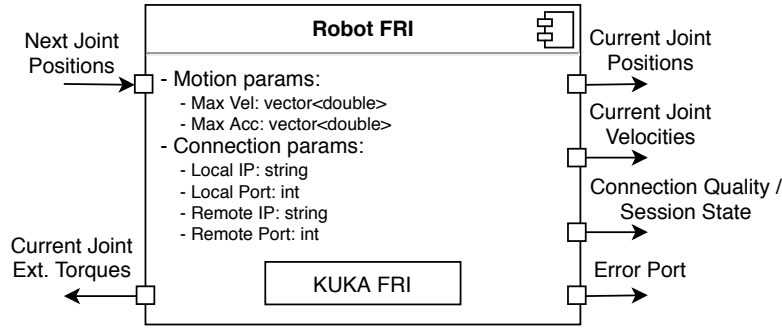


Figure 6.35: Robot FRI component.

object is instantiated and the connection established. In the component's update cycle, the client's step method is triggered to update both the measured joint data, the commanded joint data as well as other state variables. These variables were modified to have an atomic property and avoid possible concurrent access issues. Unlike the "next joint positions" port from the joint trajectory controller component, which only posts information whenever a new trajectory is being generated, the FRI module requires new commanded joint position data at each iteration. To handle this requirement, the component stores and continuously sends the last valid set of commanded joint positions to the server until a new trajectory is initialized.

The "Connection Quality" and "Session State" are two of the state variables read in each step (refer to subsection 5.6.1). These flags are monitored by the task supervisor component to avoid sending motion commands when the robot is not receptive.

The current joint positions, connection quality / session state and external joint torques are directly forwarded to the output ports. The current joint velocities, however, are not directly available in the FRI. A 5-point numerical differentiation formula was used to determine the derivative of the joint position as a function of time, i.e. the current joint velocity. Considering we have a continuous feed of joint positions at a regular time step and are interested in computing the derivative at the current time, we apply the formula to the end-point using the four preceding joint positions. Let h be the time step,

$$f'(x_0) = \frac{1}{12h} [25f(x_0 - 4h) - 48f(x_0 - 3h) + 36f(x_0 - 2h) - 16f(x_0 - h) + 3f(x_0)] + \frac{h^4}{5} f^{(5)}(\xi) \quad (6.82)$$

where $\xi \in [x_0 - 4h, x_0]$. The truncation error term ($\frac{h^4}{5} f^{(5)}(\xi)$) is considered 0 for $h = 0.005$.

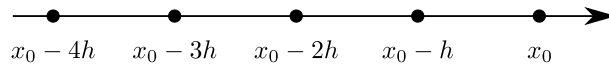


Figure 6.36: Representation of the last joint position (x_0) and the 4 preceding values.

A template library was also implemented to function as a circular buffer to store the last 5 joint positions, used as a parameter to calculate the joint velocity. The error code output port continuously updates the task supervisor component about the component's operation. If the connection to the server node is not successful, or the target joint positions are not reachable for the specified joint velocity limit (time step is fixed) an error command is issued.

6.6.2 V-REP FRI

One of the advantages of developing a high-level control architecture to move the robotic manipulator whose interfaces are mostly limited to the transmission of joint space variables is the possibility to test the solution in a virtual platform. The V-REP FRI component is the peer to the Robot FRI component that creates the bridge between the control architecture and the V-REP Plugin (refer to subsection 5.7.2).

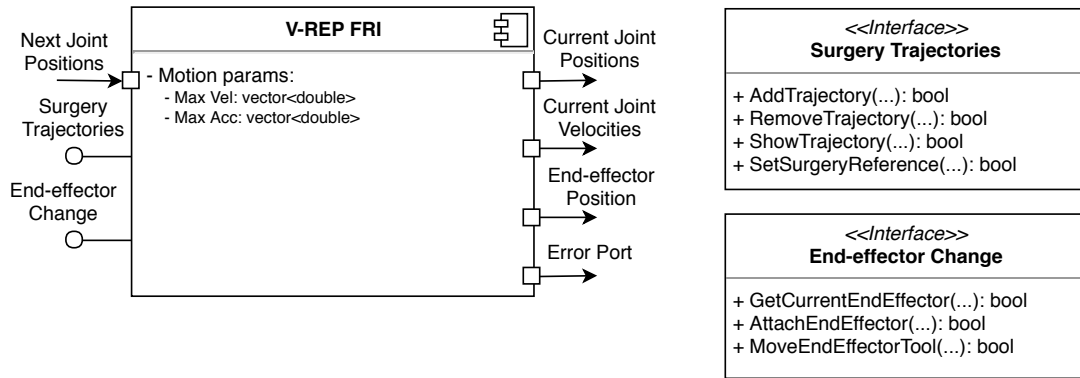


Figure 6.37: V-REP FRI component.

Dissimilar from the Robot FRI component that communicates with the KRC hardware through a Ethernet physical connection in a local network, the V-REP FRI connects to the V-REP plugin that hosts an OROCOS component using: i) inter-process communication (mqueue) if the simulation runs in the same machine as the control architecture or ii) distributed communication (CORBA middleware) if a remote computer in the local network is running the simulation. The other two differences to the peer component in terms of robot control relate to the lack of an external joint torque measurement from the virtual robot as well as the non-existing 'Connection Quality / Session State' port.

In Figure 6.37, only the interfaces with the control architecture are depicted. The interface with the V-REP plugin component mirrors⁹ the current joint positions, velocities and next joint positions ports. Internally, the component receives the 'Next Joint Positions' from the joint trajectory controller and continuously forwards either the updated target joint positions or the last valid values to the plugin. Conversely, the component receives the current joint positions and velocities from the virtual robot and forwards this information to the control architecture. The component also includes the interface for the 'Hollow Robot'.

Parallel to the interface that controls the robotic arm, the V-REP FRI component includes two additional interfaces to manage the surgery trajectories and the end-effector. The surgery trajectories allow the user to add, remove or show a particular working surgical trajectory while hiding the rest. The End-effector change interface enables the user to check which end-effector is attached to the virtual robot, to attach a new one, to move the tool (trepan, electrode, endoscope,...) and to set the surgery reference frame.

⁹input ports are output ports and vice-versa

6.7 Registration

The Registration component handles the process of intra-operative registration of the robot to the patient space (Figure 6.38). To limit the need for additional hardware, a touch-based calibration method was used. A number of points in the work space at known positions are used as anchor points. These points are

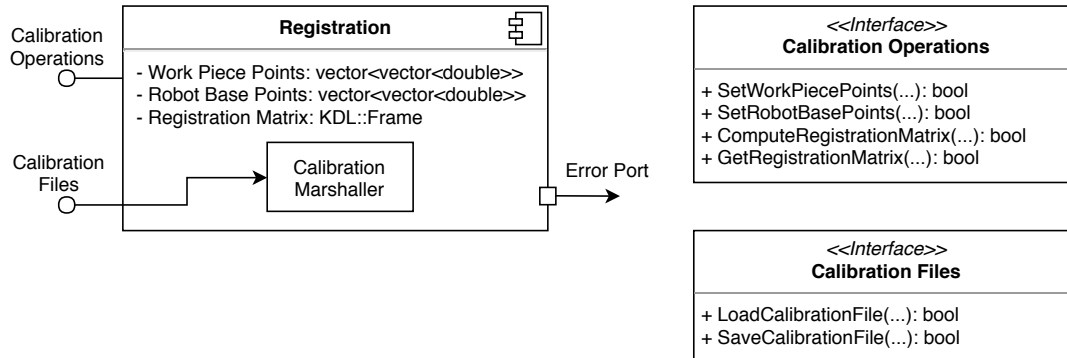


Figure 6.38: Registration component.

touched by a tool with known geometry attached to the flange of the robotic manipulator. The coordinates of these points in the robot space are recorded using Forward Kinematics, refer to subsection 6.2.2.

The two sets of points at the work and robot space are used as inputs to the Horn's closed-form method (Horn, 1987), which determines the rigid best-fitting transformation between spaces. The input datasets must match in size and in the order of points.

Finally, the registration process is only accepted if the fiducial localization error (FLE) is below a specified threshold. The FLE is determined by the sum of the squares of the Euclidean distances between matching points. Although, this measure does not correlate to the target registration error (Fitzpatrick, 2009), it does provide a measure to the quality of the calibration step.

The Calibration Marshaller package permits storing information about the registration process in files, and accessing those files to load details of a previous registration. The process of storing calibration data in non-volatile memory is particularly useful during the development and testing of new features.

Calibration files are saved in a pre-defined directory with a unique name containing the current data and hour. Information is saved in JSON format, a lightweight and readable format, according to the protocol:

```

{
  "wpts": [[3x1] ... [3x1]]
  "rpts": [[3x1] ... [3x1]]
  "cal_mat":
  {
    "pos": [3x1]
    "rot": [9x1]
  }
  "fle": "1x1"
}
  
```

The file stores the two inputs, i.e. the work piece points “wpts” and the robot base points “wpts”, along with two outputs, the rigid best-fitting transformation matrix “cal_mat” and the fiducial localization error “fle”.

6.8 Surgery Plan

The surgery plan component was created to handle the surgery directives. Geometrically, the probe / electrode insertion trajectories are described by a line segment in space with a direction, from the entry to the target point (Figure 6.39). The entry point refers to the position in Cartesian space at the inner skull surface, while the target point refers to the most distant point reached within the skull cavity, both relative to the surgery reference frame (see Figure 4.6).

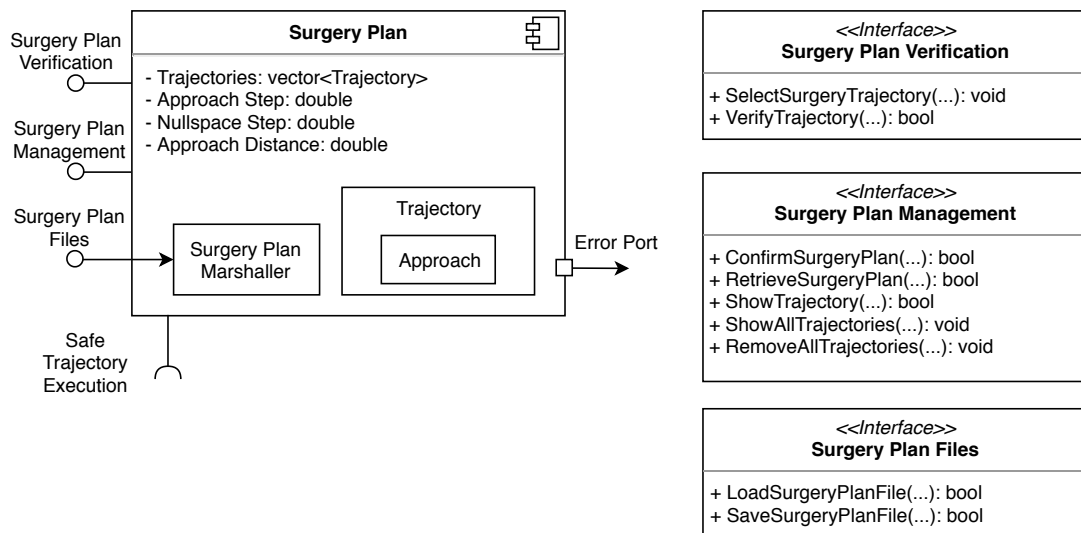


Figure 6.39: Surgery Plan component.

Internally, data about surgical trajectories are stored as “Trajectory” objects. Each “Trajectory” codes: i) the entry and ii) target positions relative to the surgery reference frame, iii) a unique identifier, iv) one flag to specify if it is selected for operation, v) another flag to identify whether the trajectory is visible (only used for virtual representation) and finally vi) a vector listing the approach solutions to the trajectory.

The approach solutions contains fully defined poses (including nullspace parameters) to position the end-effector collinear to the entry-target point trajectory while maximizing the **modified manipulability** (see subsection 6.2.6.4). Not only the best possible approach pose but also the feasibility of moving along the linear trajectory until the farthest point, usually the surgical electrode placement position, is evaluated, details in subsection 6.5.4. The approach solutions are stored as a vector of “Approach” objects, each containing a feasible solution for a single branch of global configuration (*GC*). The “Approach” object contains the highest **modified manipulability** approach frame and arm angle for a specific configuration, along with a score vector (defined in subsection 6.2.6.2).

The component provides three interfaces. The Surgery Plan Verification interface provides methods to process and collect data regarding the approach solutions. The method is split into the SelectSurgeryTra-

jectory operation, a non-blocking method that forwards information of a trajectory matching an identifier to the Task Supervisor, and the VerifyTrajectory method that collects the approach solutions data.

The Surgery Plan Management interface contains the ConfirmSurgeryPlan and RetrieveSurgeryPlan, functions responsible for receiving and storing data from the GUI and for sending back the surgery plan currently stored in the component, respectively. The RemoveAllTrajectories erases the surgery plan information. The ShowAllTrajectories and ShowTrajectory functions permit the visualization of the surgery plan trajectories in a virtual scenario (see section 5.7).

Similar to the Registration component, the Surgery Plan offers a file interface that allows the user to save and load files with surgery plan trajectory information, identifier, target and entry position. Data is stored in JSON format according to the protocol:

```
{
  "trajectory":
  {
    "id": "1x1"
    "target": [3x1]
    "entry": [3x1]
  }
  ...
}
```

6.9 Procedure Coordinator

The Procedure Coordinator component is responsible for coordinating the robotic system actions during the neurosurgery assistance task (Figure 6.40). The component is designed to be a server that interacts with the user-interface, translating the user-intention to a sequence of commands that are sent to the Core group components.

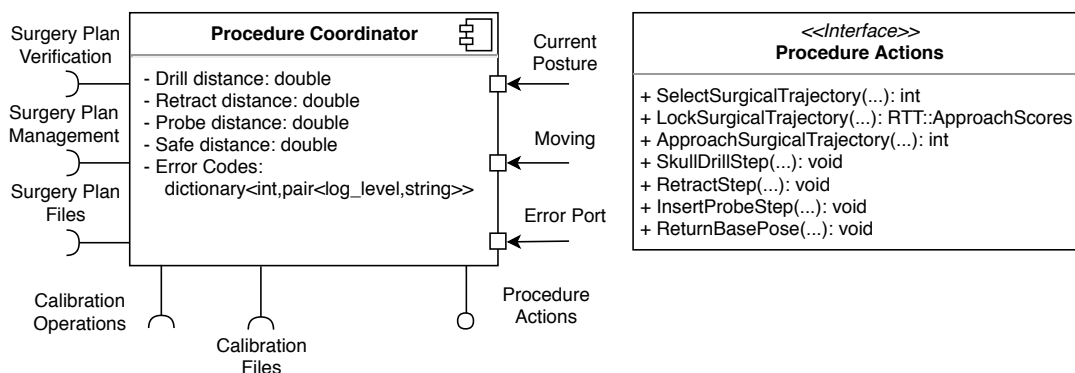


Figure 6.40: Procedure Coordinator component.

Being the server of the application layer components, the procedure coordinator acts as a proxy to the surgery plan and registration components' interfaces that are available to the user (e.g. the file

interfaces). Moreover, this component handles error messages from the peer components and provides an Error Handler mechanism similar to the one described for the Task Supervisor component (see subsection 6.5.6).

The design of the Procedure Coordinator component is strongly linked to the surgical task, see section 5.8. The Procedure Actions interface provides operations to move the robot during all stages of the surgery task. When a trajectory is selected, the information is sent to the Task Supervisor and the approach solutions processed (see section 6.8). At the same time, a visual representation of the surgical trajectory is shown in the simulation scenario, see Figure 5.16. Note that, the surgical trajectories information is only forwarded to the Task Supervisor once the appropriate end-effector is attached to the robot and the intra-operative registration matrix is known.

If the surgeon chooses to operate on the selected trajectory, the `LockSurgicalTrajectory()` operation should be called. This will prevent the system to execute actions outside the procedure for the selected trajectory. The previously processed approach solutions are now collected and presented to the robot operator. From the subset of solutions with the highest modified manipulability, the user may select the robot configuration and posture that better suit the task workspace. Selecting a solution will instantly reposition the virtual robotic arm (see Figure 5.14) in the simulation scenario to pre-visualize how it will fit the environment.

When the final approach pose is selected, the user may initiate the approach movement to align the robot end-effector with the surgical trajectory. As described in section 5.8, the robotic system operator may choose to automatically or manually maneuver the robot to the proximity of the approach pose. In both cases, the robot will need to make an adjustment to precisely align with the trajectory. The difference is that the autonomous motion will start from the home position whereas in the hand-guidance case, it starts from the pose where the robot was manually driven to. Whatever the option, the adjustment motion is split into two phases. First, the robot will perform a nullspace motion (see subsection 6.1.1.4) to match the selected elbow position. Then, the robot will attempt to move to the final approach pose with an arc motion in task space. Initially, the end-effector trajectory is defined from the start-, end-frame and a reference point (the surgical entry point), see Figure 6.4. The arc trajectory feasibility is analyzed in the Kinematics and Statics Module component (see section 6.2). If the arc motion is possible, it is executed. Else, the robot performs a point-to-point motion in joint space (refer to subsection 6.3.1).

After the approach step, the robot performs linear motions in task space to reach the tool-switching point, the tool and the drill point, see Figure 5.20. During these movements, the robot pose orientation is maintained. The robot global configuration is also maintained to avoid passing through singular configurations. The arm angle may execute small adjustments within the feasible and allowed interval to keep a distance from the joint physical limits and singular points.

When the procedure was completed for a single trajectory, the robot must necessarily return to the home position. The return to home position motion only begins after the robot linearly moves back to the approach pose (at a safe distance from the patient). Similar to the approach motion described above, the user may manually drive the robot away from the surgical trajectory, or let the robotic system handle the motion autonomously. Again, the nullspace motion is initially performed to conform to the home position

nullspace coordinates and then either the arc motion or a point-to-point motion is executed.

6.10 Graphical User Interface

The Graphical User Interface (GUI) is implemented using version 5 of the Qt Framework. It includes simple and straightforward dialogs to communicate user-intentions to the remaining architecture. The GUI integrates four components, one to communicate with each architecture server: i) a Task Supervisor proxy, ii) a Robot State proxy, iii) a V-REP FRI proxy and, a iv) Procedure Coordinator proxy. These proxy components will include the services, ports and properties to communicate with the interface of the controller architecture components.

The GUI is organized into 4 operation modes:

1. Direct Control, where the user can control the joint positions, the robot pose, nullspace or the attached end-effector;
2. Registration, the interface that guides the user through the robot-surgery registration process;
3. Surgery Plan, the interface to introduce the surgery coordinates from the image/planning software;
4. Procedure, where the user is guided through the steps of the surgical procedure.

In Figure 6.41, it is represented the window for direct setup of the robot parameters. By selecting the adequate checkboxes (top part of the window), the window will selectively display the widgets for joint position control, pose control, nullspace control or end-effector control. For the joint positions or target pose, the user may change the variable to an exact value by updating the corresponding field, or adjust the slider position. The robot motion can be triggered “offline” by specifying the target set of variables and then pressing the move button, or move the robot “online” as the sliders are displaced. The nullspace control widget permits the adjustment of the robot configuration (GC) and elbow angle (ψ). Around the wheel button, it is displayed in a color gradient the interval of feasible joint angles for the current pose, with a color correlation to the nullspace score (green to red). Finally, the end-effector control widget allows the user to change the end-effector virtually attached to the robot, and control the depth of the instruments (probes, trepans, etc).

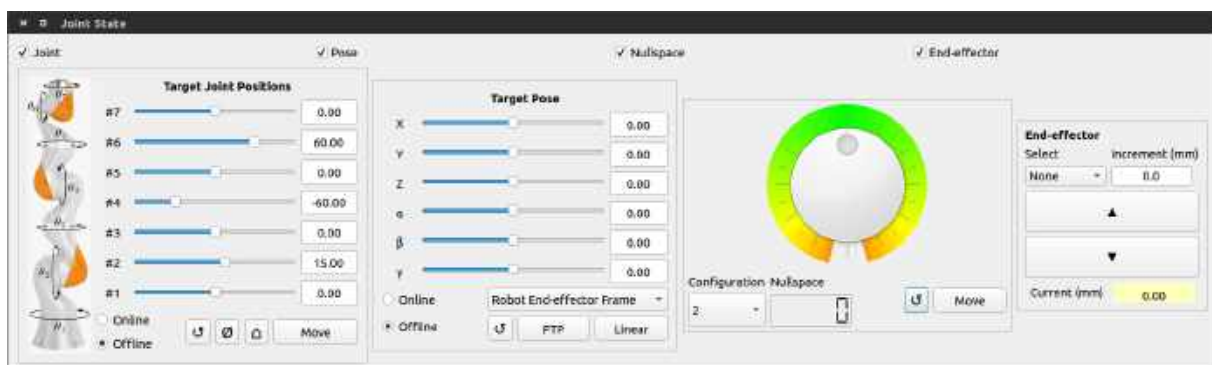


Figure 6.41: Direct control window.

The registration window displays the current end-effector pose relative to the robot's base reference frame continuously, Figure 6.42. When a new calibration process is started, the user may select a point-by-point or the interlock calibration process, both based on touch. As new points are selected, their positions are marked on the widget to the left and when the final robot-surgery transformation is determined both reference frames are depicted. The registration matrix is also visible in the Robot-Surgery Transformation widget along with the FLE.

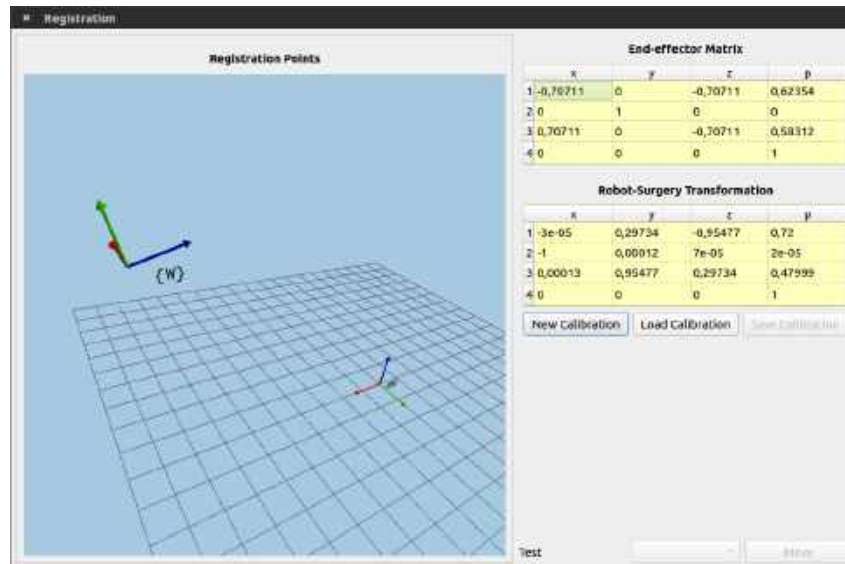


Figure 6.42: Registration window.

The surgery plan window (Figure 6.43) serves for manual introduction of the surgery coordinates, which are expressed in entry and target coordinates, refer to section 6.8.



Figure 6.43: Surgery plan.

Only when the registration process is completed and the surgery plan uploaded, the Surgical Procedure window is shown, Figure 6.44. The table at the top of the window lists the surgery plan trajectories. Every action executed in the procedure window interface requires second user confirmation, starting with the trajectory selection. Also, at any moment only the next logical actions in the surgical procedure chain

(Figure 5.19) are available to be selected¹⁰. When a procedure step is successfully completed, the led next to each button turns green.

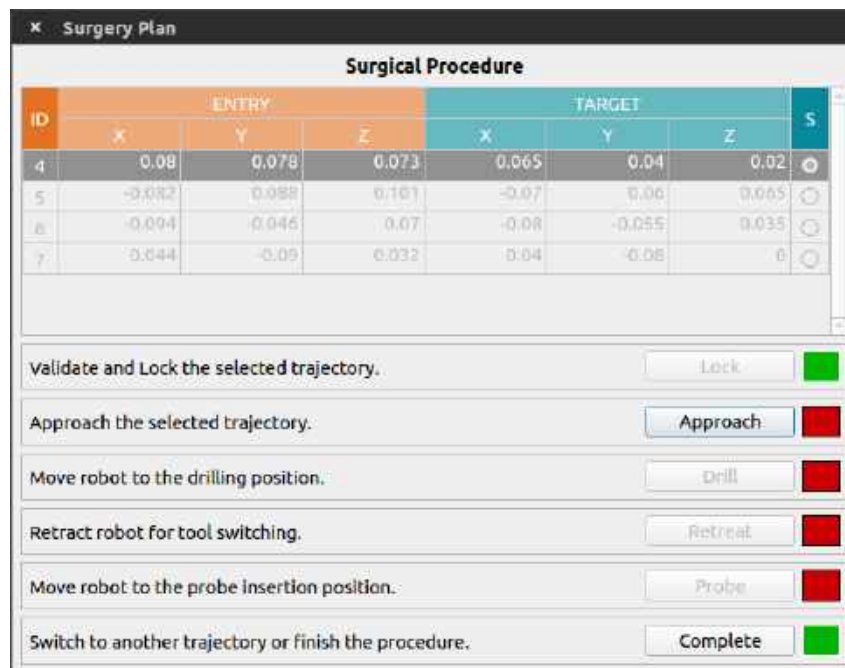


Figure 6.44: Procedure window.

For a better understanding of the Graphical User Interface refer to the video https://youtu.be/-Y_OADmaHJc.

¹⁰In Figure 6.44, the trajectory ID 4 has been selected and "Locked", the next logical step is the approach motion.

Chapter 7

Calibration and Application Accuracy

7.1 Kinematic Convention and Calibration

Kinematic identification refers to the determination of a minimal number of parameters that completely describe the position and orientation of a manipulator's structure as a function of its joint positions. Many models have been proposed to characterize a kinematic structure: the Standard and Modified Denavit-Hartenberg (DH) convention (Denavit and Hartenberg, 1955), the Hayati model (Hayati, 1983), the Stone and Sanderson's S-model (Stone, 1987) and recent models based on Product of Exponentials (POE) (He et al., 2010; Yang et al., 2014).

The Denavit-Hartenberg model is still the most used convention to represent the robot's kinematic structure. It provides a guaranteed minimal representation, an intuitive method to determine its parameters and most importantly, it works on straight-forward linear algebra whose matrices are computationally fast to solve.

Independent of the convention used, there is a consistent problem with robots, particularly serial structures that causes repeatable but inaccurate movements. This problem derives from manufacturing and assembly tolerances, wear and tear, or permanent bending due to fatigue. The listed error sources are reflected on the real kinematic model parameters, and on the gap to the nominal parameter values. Kinematic errors are especially impactful in serial manipulators where the parameter deviations propagate through the kinematic chain. Industrial robot calibration methods, particularly the ones based on kinematic parameters, are compartmentalized in four steps, i) modelling, ii) measurement, iii) identification, and compensation (Veitschegger and Chi-haur Wu, 1987; Wu et al., 2015). The proposed algorithm is primarily related to the parameter identification step as a sub-type of planar calibration methods (Khalil et al., 2009; Hage et al., 2011).

Another common problem to users that need to model kinematic structures of robotic manipulators relates to the inaccessibility of the model parameters, which are usually handled internally by the controller. Even if the robot is properly calibrated, the user has no access to the kinematic parameters other than the nominal values in the documentation.

In this section, we propose an algorithm for DH parameter identification based on geometry and dual

vector algebra for any type of serial robot. The algorithm splits into two parts, the first called “feature identification”. In this part, the robot’s end-effector position is acquired after sequential movements in each joint. The acquired set of points are processed to determine the motion axis of each joint, an idea originally explored by Stone (1987) to determine the S-model parameters. The second part, “parameter extraction”, applies dual vector algebra to calculate the intermediate coordinate frames between consecutive joints, and then to extrapolate the Standard DH parameters. The dual vector algebra concept was applied by Ketchel and Larochelle (2006) to detect collisions between robotic links modeled after cylindrical bodies.

To the best of our knowledge, the proposed algorithm is the first capable of correctly identifying a set of DH parameters for a wide range of differentiated robots, representing various kinematic typologies. The method was tested against serial robotic manipulators with: revolute and prismatic joints, intrinsically redundant and non-redundant structures, joints with aligned motion axis, shoulder and wrist offsets, as well as with curved wrists. It corrects and extends previous work by Hayat et al. (2013); Rajeevlochana et al. (2012), which fails to deliver a correct set of DH parameters for manipulators other than the industrial serial 6-DoF type. The source code of the algorithm here proposed is available at <https://github.com/neuebot/Kinematic-Calibration>.

The proposed method was tested with several robot models with variable structures and joint dispositions including a KUKA LBR iiwa R800, a Rethink Robotics Sawyer, an ABB IRB 140, a Universal Robots UR3, a KINOVA MICO and a SCARA-type Cobra 650 by Omron. A correct set of DH parameters was identified for all (Faria et al., 2019). The measured parameters for the KUKA LBR iiwa R800 were ultimately used for the kinematic model.

7.1.1 Features Identification

The objective of this section is to determine, for each joint of the manipulator, its motion axis: a) for a revolute joint - the plane and center of rotation; b) for a prismatic joint - the sliding vector and a contained point. These values constitute the input to the proposed algorithm.

To identify the motion axis of each joint, we move each robot joint separately and acquire the position of a point ($\mathbf{p} = (x, y, z)^T$) at the robot’s end-effector relative to the robot’s base reference frame. Either $\mathbf{p}$ is determined directly from the robot controller, which allows for the extraction of the controller’s intrinsic kinematic parameters, or $\mathbf{p}$ is determined using an external tracking sensor to achieve kinematic calibration. For each joint, from the base to the end-effector, a movement in joint space is executed, which translates to a circular or linear trajectory of the tracked point ($\mathbf{p}$) in Cartesian space depending on the type of the joint.

For consistent and reliable results, our method imposes the following requirements:

1. the reference frame attached to the first joint should be aligned with the base reference frame;
2. the joint motion must be in the positive direction during the motion acquisition;
3. the acquired position measurement step should be 3 to 4 orders of magnitude less than the nominal distance of the generated trajectory;

4. the acquired point must be contained in the plane defined by the normal to the last joint axis that contains the end-effector position, see Figure 7.1.

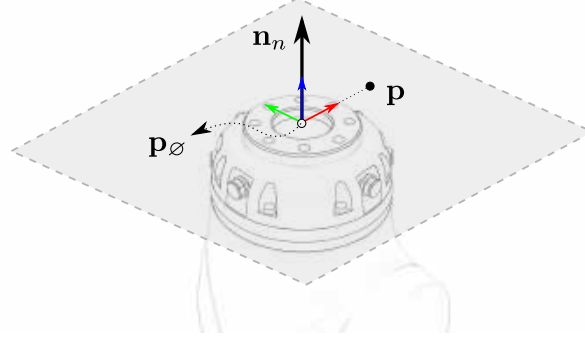


Figure 7.1: Example of acquired point ($\mathbf{p}$) contained in the plane defined by the last joint axis ($\mathbf{n}_n$) and the position of the flange. The acquired point must not be contained in the line that passes through the last joint axis, i.e. $\mathbf{p} \neq \mathbf{p}_\emptyset$.

The last condition assures that the tracked point trajectory generated from the last joint is not reduced to a single point. In addition, for the method here proposed, we recommend that:

1. each joint travels more than a third of its range;
2. each joint travels at identical and constant velocities - to guarantee a similar number of samples and uniform distribution of end-effector positions.

After acquiring the generated trajectories of $\mathbf{p}$ for each isolated joint, one should end up with n trajectories (n equates to the number of joints), each with m number of points corresponding to the tracked point measurements, $\mathbf{P} = \{\mathbf{p}_j \in \mathbb{R}^3, j \in \{1, \dots, m\}\}$.

Since the process is iterative to each joint, we explain in the following section how it applies to a generic prismatic and a revolute joint.

7.1.1.1 Prismatic Joints

A prismatic joint motion will generate a linear trajectory in workspace. The motion axis vector can be determined by finding the best fitting line to the set of trajectory points. Line and plane fitting are problems commonly addressed with Orthogonal Distance Regression. The goal is to minimize the distances between the set of points and the geometric element. Let the position of the best fitting line be represented by a point $\mathbf{c}$ belonging to the line, and let the vector $\mathbf{n}$ be the direction vector. The orthogonal distance (d_j) between each point ($\mathbf{p}_j$) and the line is,

$$d_j = \|(\mathbf{p}_j - \mathbf{c}) - ((\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n}) \mathbf{n}\|, \quad (7.1)$$

assuming,

$$\mathbf{c} = \frac{1}{m} \sum_{j=1}^m \mathbf{p}_j. \quad (7.2)$$

and without the loss of generality $\mathbf{n}$ to be a unit vector (i.e. $\|\mathbf{n}\| = 1$). For clarity, consider the vector norm $\|\bullet\|$ to be the Euclidean norm.

The best fitting line minimizes the square sum of orthogonal distances between the line and the points,

$$\min_{\mathbf{n}} \sum_{j=1}^m d_j^2$$

equivalent to,

$$\min_{\mathbf{n}} \left[\sum_{j=1}^m \|\mathbf{p}_j - \mathbf{c}\|^2 - \sum_{j=1}^m ((\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n})^2 \right]$$

that can be written as,

$$\min_{\mathbf{n}} \left(\|\mathbf{M}\|_F^2 - \|\mathbf{M}\mathbf{n}\|^2 \right),$$

where $\mathbf{M}$ is the matrix of $3 \times m$ mean-centered points, i.e.,

$$\mathbf{M} = \begin{bmatrix} \mathbf{p}_1 - \mathbf{c} \\ \vdots \\ \mathbf{p}_m - \mathbf{c} \end{bmatrix} \quad (7.3)$$

and $\|\bullet\|_F$ is the Frobenius norm. Thus, the problem translates to finding,

$$\mathbf{n} := \arg \max_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \|\mathbf{M}\mathbf{n}\|^2. \quad (7.4)$$

One efficient approach to deal with this problem relies on Singular Value Decomposition (SVD) to determine the dominant direction of data. Factorizing $\mathbf{M}$ into the product of three matrices $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices whose columns are orthonormal and $\mathbf{\Sigma}$ is a diagonal matrix with positive and real singular values listed in decreasing order ($\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$). It follows that,

$$\|\mathbf{M}\mathbf{n}\|^2 = \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{n}\|^2 \quad (7.5)$$

and given that $\mathbf{U}$ is an unitary matrix, the following is also true,

$$\|\mathbf{M}\mathbf{n}\|^2 = \|\mathbf{\Sigma}\mathbf{V}^T\mathbf{n}\|^2.$$

Considering $\mathbf{h} = \mathbf{V}^T\mathbf{n}$, the function (7.4) is equivalent to,

$$\mathbf{n} := \arg \max_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \left[(\sigma_1 h_1)^2 + (\sigma_2 h_2)^2 + (\sigma_3 h_3)^2 \right]. \quad (7.6)$$

Provided the decreasing order of singular values, the function is maximized for $\mathbf{h}_{\max} = [1, 0, 0]^T$, which follows that,

$$\mathbf{n} := \mathbf{V}\mathbf{h}_{\max}. \quad (7.7)$$

The tail-arrow direction of $\mathbf{n}$ is also important because it relates to the direction of the axis. As one of the imposed requirements for the calibration was to record a linear joint motion in the positive direction, one can guarantee the correct tail-arrow direction of $\mathbf{n}$ by,

$$\mathbf{n} = \begin{cases} \mathbf{n}, & \text{if } \mathbf{n} \cdot (\mathbf{p}_m \times \mathbf{p}_1) \geq 0 \\ -\mathbf{n}, & \text{if } \mathbf{n} \cdot (\mathbf{p}_m \times \mathbf{p}_1) < 0 \end{cases}. \quad (7.8)$$

7.1.1.2 Revolute Joints

One logical approach to calculate the plane and center of rotation is to determine the best fitting tri-dimensional circle to the trajectory, from which it is straightforward to extrapolate the plane and center of rotation. The first step consists in determining the best fitting plane to the set of points $\mathbf{P}$. To keep a consistent notation, let this plane be represented by its normal vector $\mathbf{n}$, and a plane contained point $\mathbf{c}$ determined from the centroid of the set of points, similar to the previous subsection. The orthogonal distances between the plane and the set of points are,

$$d_j = (\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n}. \quad (7.9)$$

The problem of finding the best fitting plane is similar to the one described in subsection 7.1.1.1,

$$\mathbf{n} := \arg \min_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \|\mathbf{M}\mathbf{n}\|^2. \quad (7.10)$$

which simplifies to,

$$\mathbf{n} := \arg \min_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} [(\sigma_1 h_1)^2 + (\sigma_2 h_2)^2 + (\sigma_3 h_3)^2]. \quad (7.11)$$

that is minimized for $\mathbf{h}_{\min} = [0, 0, 1]^T$,

$$\mathbf{n} := \mathbf{V}\mathbf{h}_{\min}. \quad (7.12)$$

Again and in a similar fashion to the previous subsection, the tail-arrow direction of the plane vector can be correctly determined using equation (7.8).

Now, since the best fitting plane for the points $\mathbf{P}$ has been determined, we can determine the circle that best fits these points in two dimensions, i.e projected into the plane defined by $\mathbf{n}$. First, the transformation of $\mathbf{P}$ to the best fitting plane coordinates, $\mathbf{n}$, is achieved with the Rodrigues' rotation formula,

$$\mathbf{p}_{\mathbf{n},j} = \mathbf{p}_j \cos \rho + (\mathbf{a} \times \mathbf{p}_j) \sin \rho + \mathbf{a}(\mathbf{a} \cdot \mathbf{p}_j)(1 - \cos \rho), \quad (7.13)$$

which rotates $\mathbf{P}$ around an axis $\mathbf{a}$ of an angle ρ ,

$$\mathbf{a} = \mathbf{n} \times [0, 0, 1]^T \quad (7.14)$$

$$\rho = \text{atan}_2 \left(\|\mathbf{n} \times [0, 0, 1]^T\|, \mathbf{n} \cdot [0, 0, 1]^T \right). \quad (7.15)$$

The projection of these points into the plane coordinates, XY -plane, is then simply accomplished by using the x and y coordinates.

With the projected points into the XY plane - $\mathbf{P}_n = \{\mathbf{p}_{n,j} \in \mathbb{R}^3, j \in \{1, \dots, m\}\}$, it is straightforward to determine the best 2D fitting circle. Consider a circle of radius r and center (x_c, y_c) , which is represented by,

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (7.16)$$

simplified as a function of x and y , one ends up with,

$$(2x_c)x + (2y_c)y + (-x_c^2 - y_c^2 + r^2) = x^2 + y^2. \quad (7.17)$$

We can now transform this into a system of linear first degree equations using the projected points $\mathbf{p}_{n,j} = (x_j, y_j, 0)^T$ to determine the circle center coordinates and radius. Converting (7.17) to matrix notation ($\mathbf{Ax} = \mathbf{b}$),

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_m & y_m & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 2x_c \\ 2y_c \\ -x_c^2 - y_c^2 + r^2 \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} x_1^2 + y_1^2 \\ \vdots \\ x_m^2 + y_m^2 \end{bmatrix}.$$

The system is solved as a function of $\mathbf{x}$ through a least-squares fitting approach. The goal is to minimize the square sum of residual errors $\|\mathbf{b} - \mathbf{Ax}\|^2$. It is now straightforward to find the center and radius of the circle once $\mathbf{x}$ is determined.

Having determined the best fitting circle in the plane defined by $\mathbf{n}$, it can be directly transformed to tri-dimensional coordinates by applying the inverse transformation to (7.13), i.e. inverting the axis and angle of rotation.

7.1.2 Parameter Extraction

With the vector $\mathbf{n}$ and the point $\mathbf{c}$ determined relative to the robot base reference frame and for each joint motion, one can extrapolate the Denavit-Hartenberg parameters using a geometric approach that consists of two steps:

1. determine the relative coordinate frames that relate each actuated joint;
2. identify the DH parameters from the set of frames.

The relative coordinate frames (7.18) can be partially constructed from the information gathered thus far.

$${}^i\mathbf{T}_{i+1} = \begin{bmatrix} \hat{\mathbf{x}}_{i+1} & \hat{\mathbf{y}}_{i+1} & \hat{\mathbf{z}}_{i+1} & \mathbf{p}_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7.18)$$

where

$$\begin{aligned}\hat{\mathbf{z}}_{i+1} &= \mathbf{n}_{i+1} \\ \hat{\mathbf{y}}_{i+1} &= \hat{\mathbf{z}} \times \hat{\mathbf{x}}\end{aligned}$$

Per definition, the z-axis should equate to the direction of the joint axis ($\mathbf{n}_i$). Similarly, the x-axis can be determined from the common normal to both direction axes, although this vector might not be uniquely determined. From the right hand rule the y-axis is directly determined once the z- and the x-axes are known. On the other hand, determining the position vector of each coordinate frame requires some Dual Vector geometrical operations to calculate the intersection/closest points between the axis lines.

Each pair of parameters ($\mathbf{n}_i, \mathbf{c}_i$) forms the base to represent a line in tri-dimensional space. Alternatively, the same information can be formulated in Plücker coordinates (S) using the notation of the moment vector ($\mathbf{k}$),

$$S = \begin{bmatrix} \mathbf{n} \\ \mathbf{c} \times \mathbf{n} \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ \mathbf{k} \end{bmatrix}. \quad (7.19)$$

Analogous to complex numbers, dual number notation can be expressed by a sum of two parts: the primary component or real part ($\mathbf{n}$) and the dual component or dual part ($\mathbf{k}$) (Fischer, 1998):

$$S = \mathbf{n} + \varepsilon \mathbf{k} \quad (7.20)$$

where $\varepsilon \neq 0$ and $\varepsilon^2 = 0$. This representation is especially useful to study the relationship of two straight lines in tri-dimensional space, Figure 7.2. Ketchel and Larochelle (2006) proposed an algorithm to classify this relationship based on dual vector representation for the purpose of collision detection. Using dual

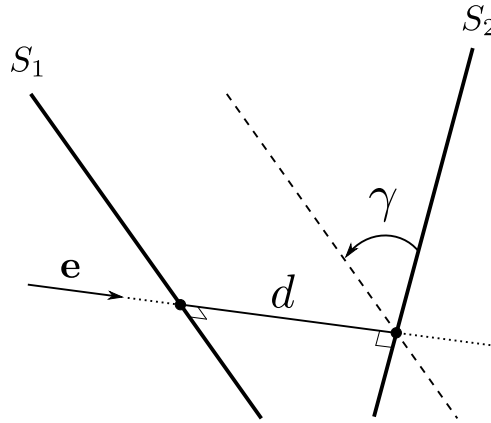


Figure 7.2: The relationship between two lines in space can be parameterized by the shortest distance between lines d (dual part), and the projected angle between the lines γ (real part) (Ketchel and Larochelle, 2006). The common normal is, $\hat{\mathbf{e}} = \|\mathbf{n}_1 \times \mathbf{n}_2\|$. The distance between lines is, $d = (\mathbf{c}_2 - \mathbf{c}_1) \cdot \mathbf{e}$. The angle between lines can be determined from, $\gamma = \text{atan2}(\|\mathbf{n}_1 \times \mathbf{n}_2\|, \mathbf{n}_1 \cdot \mathbf{n}_2)$.

vector algebra to determine the dot and cross products of S_1 and S_2 one can determine the distance (d)

and angle (γ) that relate them,

$$\begin{aligned}
\hat{S}_1 \cdot \hat{S}_2 &= (\mathbf{n}_1, \mathbf{k}_1) \cdot (\mathbf{n}_2, \mathbf{k}_2) \\
&= (\mathbf{n}_1 \cdot \mathbf{n}_2, \mathbf{n}_1 \cdot \mathbf{k}_2 + \mathbf{k}_1 \cdot \mathbf{n}_2) \\
&= \mathbf{n}_1 \cdot \mathbf{n}_2 + \varepsilon(\mathbf{n}_1 \cdot \mathbf{k}_2 + \mathbf{k}_1 \cdot \mathbf{n}_2) \\
&= \cos \gamma - \varepsilon d \sin \gamma
\end{aligned}$$

$$\begin{aligned}
\hat{S}_1 \times \hat{S}_2 &= (\mathbf{n}_1, \mathbf{k}_1) \times (\mathbf{n}_2, \mathbf{k}_2) \\
&= (\mathbf{n}_1 \times \mathbf{n}_2, \mathbf{n}_1 \times \mathbf{k}_2 + \mathbf{k}_1 \times \mathbf{n}_2) \\
&= \mathbf{n}_1 \times \mathbf{n}_2 + \varepsilon(\mathbf{n}_1 \times \mathbf{k}_2 + \mathbf{k}_1 \times \mathbf{n}_2) \\
&= (\sin \gamma - \varepsilon d \cos \gamma) \hat{\mathbf{e}}.
\end{aligned}$$

S_1 and S_2 are either: **intersecting**, **identical**, **parallel**, or **skewed**; best explained in the flowchart Figure 7.3.

7.1.2.1 Intersecting lines

The simplest case involves **intersecting lines**, as it logically follows that the position vector should be located at the intersection point,

$$\mathbf{p}_{\text{int}} = \begin{cases} \frac{\mathbf{k}_1 \times \mathbf{k}_2}{\mathbf{n}_2 \cdot \mathbf{k}_1}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_2 - \mathbf{n}_2 \cdot \mathbf{k}_1 \neq 0 \\ \frac{\mathbf{k}_2 \times \mathbf{k}_1}{\mathbf{n}_1 \cdot \mathbf{k}_2}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_2 - \mathbf{n}_2 \cdot \mathbf{k}_1 = 0 \end{cases}. \quad (7.21)$$

While (7.21) returns a solution for most cases, if any of the lines passes through the origin of the reference system, its moment vector becomes null ($\mathbf{k} = 0$) making it impossible to determine $\mathbf{p}_{\text{int}}$. Rajeevlochana et al. (2012) proposed a solution for this special case, which involves computing an auxiliary intersection point at an auxiliary coordinate frame and then applying the inverse transformation to obtain the real intersection point. The auxiliary coordinate frame (${}^1\mathbf{T}_{1'}$) is found by applying a pure translation of a unit distance from the reference frame along the common normal of the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$,

$${}^1\mathbf{T}_{1'} = \begin{bmatrix} \mathbf{I}_3 & \|\mathbf{n}_1 \times \mathbf{n}_2\| \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (7.22)$$

The computation of the auxiliary intersection point in the auxiliary reference frame is exactly the same as for the normal case (7.23), the only difference being the translation of the auxiliary center of rotations coordinates ($\mathbf{c}_{1'}$ and $\mathbf{c}_{2'}$) and consequently the auxiliary moment vectors ($\mathbf{k}_{1'}$ and $\mathbf{k}_{2'}$). From the auxiliary system, the actual intersection point is determined by translating the auxiliary intersection point back to the

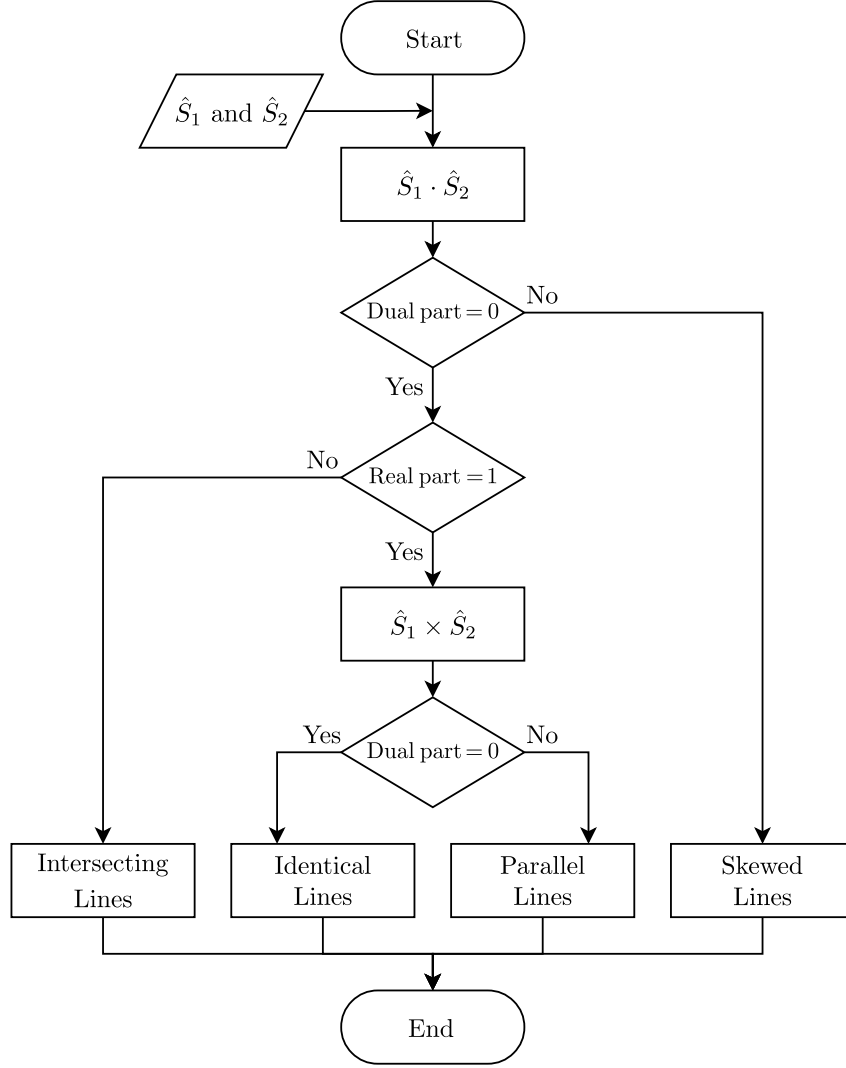


Figure 7.3: The line relationship categorization process.

original reference system, as given by:

$$\mathbf{p}_{\text{int}} = \begin{cases} ({}^1\mathbf{T}_{1'})^{-1} \frac{\mathbf{k}_{1'} \times \mathbf{k}_{2'}}{\mathbf{n}_2 \cdot \mathbf{k}_{1'}}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_{2'} - \mathbf{n}_2 \cdot \mathbf{k}_{1'} \neq 0 \\ ({}^1\mathbf{T}_{1'})^{-1} \frac{\mathbf{k}_{2'} \times \mathbf{k}_{1'}}{\mathbf{n}_1 \cdot \mathbf{k}_{2'}}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_{2'} - \mathbf{n}_2 \cdot \mathbf{k}_{1'} = 0 \end{cases}, \quad (7.23)$$

where

$$\begin{aligned} \mathbf{k}_{1'} &= \mathbf{k}_1 + (\|\mathbf{n}_1 \times \mathbf{n}_2\| \times \mathbf{n}_1) \\ \mathbf{k}_{2'} &= \mathbf{k}_2 + (\|\mathbf{n}_1 \times \mathbf{n}_2\| \times \mathbf{n}_2) \end{aligned}$$

The relative coordinate frame (7.18) is now fully defined for intersecting lines:

$$\begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_{\text{int}} \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_i \times \mathbf{n}_{i+1} \end{aligned}$$

7.1.2.2 Identical lines

Identical lines are perhaps the most uncommon of all cases (see the SCARA robot “Cobra 650” in section 7.1.3). No intersection point or common normal are uniquely determined and thus it is up to the algorithm to decide both. The relative coordinate frame (7.18) may be defined for identical lines according to:

$$\begin{aligned}\mathbf{p}_{i+1} &= \mathbf{p}_i \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_{1'}\end{aligned}$$

7.1.2.3 Parallel lines

If the lines are **parallel**, there is no intersecting point and there is no unique common normal. In this case, the algorithm imposes a point through which the common normal should pass through, for example, the center of rotation of the first joint ($\mathbf{c}_1$) in the studied pair. Using Dual Vector algebra, a line ($S_{1'}$) can be drawn from $\mathbf{c}_1$ to the closest point along S_2 ,

$$S_{1'} = \begin{bmatrix} \mathbf{n}_{1'} \\ \mathbf{c}_1 \times \mathbf{n}_{1'} \end{bmatrix} \quad (7.24)$$

where

$$\mathbf{n}_{1'} = \|\mathbf{n}_1 \times ((\mathbf{c}_2 - \mathbf{c}_1) \times \mathbf{n}_1)\| \quad (7.25)$$

It is now possible to determine the intersection point between $S_{1'}$ and S_2 with the method described for intersecting lines. Note that, if both lines are parallel, the common normal vector (specified as $\hat{\mathbf{e}}$ in Figure 7.2) is calculated as in (7.25). The relative coordinate frame (7.18) is now fully defined for parallel lines:

$$\begin{aligned}\mathbf{p}_{i+1} &= \mathbf{p}_{\text{int}} \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_{1'}\end{aligned}$$

7.1.2.4 Skewed lines

Finally, if the lines are skewed we have a two-part solution representing the closest points along the lines S_1 and S_2 to one another,

$$\mathbf{p}_{\text{int},1} = \left(\frac{\mathbf{k}_2 \cdot \mathbf{e} - \cos \gamma \mathbf{k}_1 \cdot \mathbf{e}}{\sin \gamma} \right) \mathbf{n}_1 + \mathbf{n}_1 \times \mathbf{k}_1 \quad (7.26)$$

$$\mathbf{p}_{\text{int},2} = \left(\frac{-\mathbf{k}_1 \cdot \mathbf{e} + \cos \gamma \mathbf{k}_2 \cdot \mathbf{e}}{\sin \gamma} \right) \mathbf{n}_2 + \mathbf{n}_2 \times \mathbf{k}_2. \quad (7.27)$$

The skewed lines case is specially relevant for robots that have offsets. Both points are used to compute DH parameters. The common normal in these cases is determined from the vector that passes through both points.

$$\begin{aligned}\mathbf{p}_{i+1} &= \mathbf{p}_{\text{int},2} \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_i \times \mathbf{n}_{i+1}\end{aligned}$$

Due to the fact that two solutions are possible for the common normal, accounting for the inverted vector, two conventions are suggested.

Whenever the common normal is aligned with the previous frame common normal, $\|\hat{\mathbf{x}}_{i+1} \cdot \hat{\mathbf{x}}_i\| = 1$, then the direction of the current common normal should match the previous, $\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i$. Else, when considering two inverted common normals with a non-zero y-component, $\|\hat{\mathbf{x}}_{i+1} \cdot [0 \ 1 \ 0]\| > 0$, we select the solution with a positive y-component.

7.1.2.5 Extracting DH parameters

Knowing $\mathbf{p}_{i+1}$ and $\hat{\mathbf{x}}_{i+1}$, we completely define the relative coordinate frame between actuated joints independent from the robot structure. To avoid ambiguities, the robot base frame is purposely identical to the system reference frame, implying that its origin is coincident with the origin of the first relative coordinate frame (joint 0), and its z-axis aligned with the first joint rotation axis. With the first relative coordinate frame assigned, the process of finding the remaining DH parameters is iterative for each $i \in \{0, \dots, n-1\}$. Note that the point $\mathbf{p}_n$ (position of the last joint) as well as $\hat{\mathbf{z}}_n$ are required. This problem is abbreviated by providing an end-effector position ($\mathbf{p}_n$) as input, and assuming $\hat{\mathbf{z}}_n$ to be parallel to the last joint rotation axis ($\hat{\mathbf{z}}_{n-1}$). If not, the final frame (${}^{n-1}\mathbf{T}_n$) is also required as input.

After calculating the relative coordinate frames from the base to the robot's end-effector, the four DH parameters used to describe each transformation are:

1. d_i , offset from the previous z-axis to the common normal,

$$d_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \hat{\mathbf{z}}_i \quad (7.28)$$

2. θ_i , angle between the previous to the current x-axis around the previous z-axis,

$$\theta_i = \text{atan2}((\hat{\mathbf{x}}_i \times \hat{\mathbf{x}}_{i+1}) \cdot \hat{\mathbf{z}}_i, \hat{\mathbf{x}}_i \cdot \hat{\mathbf{x}}_{i+1}) \quad (7.29)$$

3. a_i , distances between origins along the common normal,

$$a_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \hat{\mathbf{x}}_{i+1} \quad (7.30)$$

4. α_i , angle between the previous to the current z-axis around the common normal,

$$\alpha_i = \text{atan2}((\hat{\mathbf{z}}_i \times \hat{\mathbf{z}}_{i+1}) \cdot \hat{\mathbf{x}}_i, \hat{\mathbf{z}}_i \cdot \hat{\mathbf{z}}_{i+1}). \quad (7.31)$$

7.1.3 Experiment and Results

The algorithm was first tested and validated in V-REP (Coppelia Robotics GmbH, Zürich, Switzerland), a robotics simulator. With an extensive library of robotic models available, it permitted testing the algorithm with different models containing varying kinematic structures, and joint types / dispositions in space. The proposed method was tested with the following virtual robot models: i) ABB IRB 140, 6-DoF multipurpose industrial robot; ii) Universal Robot UR3, 6-DoF collaborative robot; iii) KINOVA MICO, 6-DoF collaborative robot with a curved wrist; and the iv) Omron Cobra 650, 4-DoF SCARA robot.

The dimensions of each virtual model were verified against the available online documentation(ABB, 2019; Universal Robots, 2019; Robotics, 2009; Omron, 2019). Tests were conducted in a dynamic simulation environment, meaning that each robot link has a physically modeled body and the joints connecting the links were driven by a dynamically simulated motor based on torque/force input and a PID-controller. These physical elements were handled by the Vortex physical engine (CM Labs, Montreal, QC, Canada). The physics engine inherently introduces noise to the read joint and end-effector positions, adding to the simulation realism.

The proposed method was then tested in two real robotic systems, an LBR iiwa R800 (KUKA, Augsburg, Germany) and a Sawyer (Rethink Robotics - HAHN Group, Bergisch Gladbach, Germany). Both are lightweight anthropomorphic serial manipulators designed for sensitive and cooperative tasks, and tailored for research development.

The **KUKA LBR iiwa R800** robot fits in the SRS (spherical-revolute-spherical) category of redundant manipulators (Faria et al., 2018). In each of its 7 revolute joints the robot includes torque sensors and position encoders. For real-time control and acquisition of the robotic manipulator joint positions, the module Connectivity Sunrise.FRI (Fast Robotic Interface) was used. A client application running in a remote host was developed to communicate with the controller unit (Sunrise Cabinet) in a local network. Each 5 ms the client forwards the target joint positions to be reached in the next control cycle, and receives the current joint positions, Figure 7.4.

The **Sawyer** robot is an anthropomorphic robot with 7 revolute joints that contrary to the LBR iiwa R800 does not have a SRS structure. This is due to several offsets distributed along its kinematic chain - 2 at the shoulder, 1 at the elbow and 1 at the wrist - which make it a particularly interesting case for DH parameter identification. The Intera SDK software interface was used to remotely communicate with the controller using the ROS API (Quigley et al., 2009). The client application receives the robot joint positions each 5ms.

The data acquisition process to determine the model parameters was similar across all robot models. The robots execute a series of position controlled movements defined in joint space, Table 7.1. The sequence

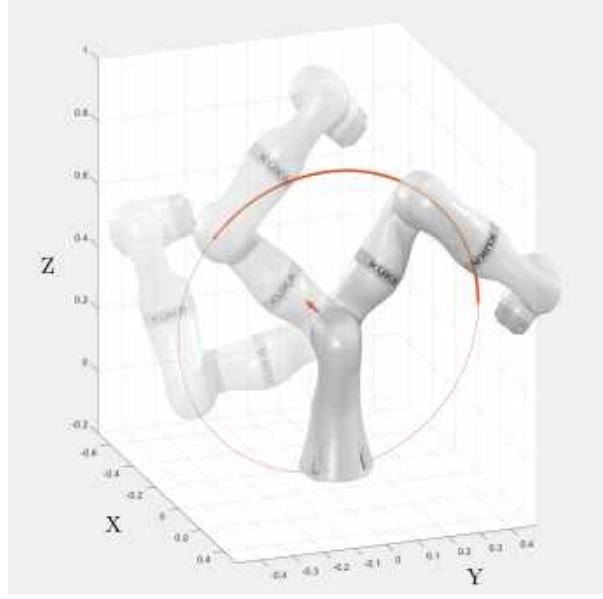


Figure 7.4: Example of the robot motion that generates the tracked point trajectory for the second joint. The thicker lines represent the position of the tracked point acquired relative to the robot base frame. The thinner lines represent the best fitting circle to the trajectories.

of motions proceeds iteratively from the first to the last joint as follows¹:

1. the robot executes a PTP (point-to-point) movement to a set of initial joint positions, θ_{init} ;
2. the robot moves one joint to its starting position, $\theta_{i,start}$;
3. the position of $\mathbf{p}$ starts being acquired;
4. the robot drives the same joint until reaching its end position, $\theta_{i,end}$;
5. the point acquisition stops and the file is saved under the identification of the current joint.

The acquired points datasets files serve as the input for the proposed algorithm implemented in MATLAB², see an example in Figure 7.5.

The proposed algorithm was tested for the robotic manipulators listed above. The measured DH parameters were compared to the nominal values and the results are listed in Table 7.2. The parameters obtained have, for the most part, sub-millimetric and sub-degree deviation to the nominal values for the virtual and the real robotic manipulators. Two odd cases occur as a consequence of the undetermination of the common normals and intersection points in parallel joints of the MICO and the UR3. The value discrepancy is related to the d parameter, the offset from the previous z-axis to the common normal. These differences are related to the non-continuity of the DH parameters in actuators with parallel, or almost parallel consecutive joint axes. These discrepancies are nullified in the next algorithm iteration where the joint axis i and $i + 1$ intersect, causing no impact in the Forward Kinematic calculation. Another issue occurred with the UR3 robot, the algorithm determined the common normal (x-axis) at the 5th joint to be

¹See video at https://youtu.be/DUqfr_Q9n38

²Repository at <https://github.com/neuebot/Kinematic-Calibration>

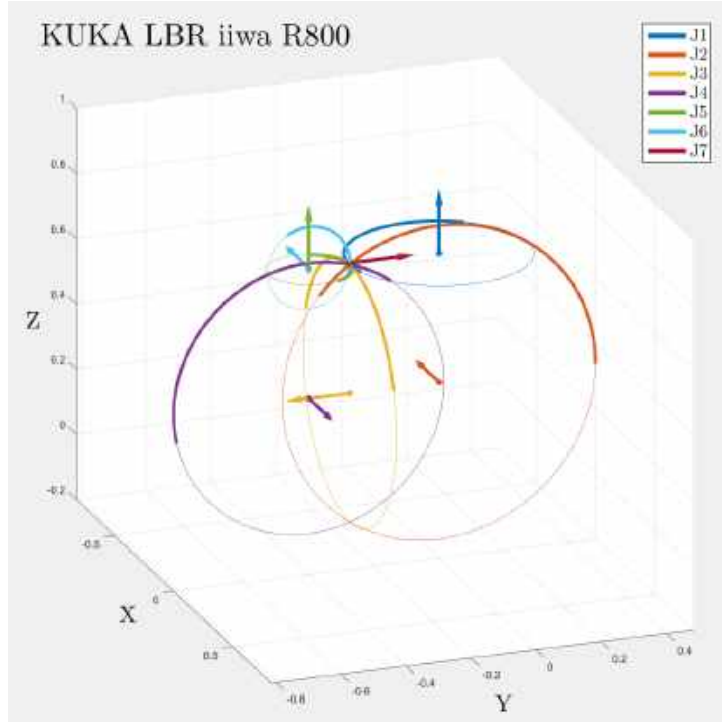


Figure 7.5: Acquired point trajectories. For each revolute joint, the motion axis is represented at the center of rotation.

the inverse of the nominal values. This issue is solved by providing the final frame (${}^{n-1}\mathbf{T}_n$) as suggested in subsection 7.1.2.5.

Experiments were conducted with different ranges of joint trajectories acquired. It is noteworthy that trajectories where each joint traveled about a third of the total mechanical range returned parameters more similar to the nominal values. When the trajectory distance was increased beyond that point, no noticeable differences were reported. On the other hand, decreasing the traveled distance below a third of the total range lead to increasingly dissonant DH parameters from the nominal ones.

7.1.4 Discussion

A novel algorithm for automatic determination of the Standard Denavit-Hartenberg parameters of serial robotic manipulators was proposed. It determines a correct set of DH parameters for several robots with variable kinematic structures and joint types/dispositions. The method was tested in two real robots and four dynamically simulated robots, the later using the Vortex physical engine to reproduce as closely as possible the behavior of the real robots.

The applicability of the algorithm is twofold. First, if the end-effector position is determined using the robot controller, the intrinsic parameters used by the controller can be extracted as they are usually non-accessible. Second, if an external sensor (e.g. optical tracker) is used to measure the end-effector position, one can perform kinematic calibration within the error margin of the sensor.

The algorithm proved to be valid for the detection of DH parameters on all tested systems and yet it is affected by the limitations of the DH notation, i.e. the non-continuity in actuators with parallel, or almost

Table 7.1: Initial, start and end joint coordinates used during the trajectory execution. For readability, the joint positions are displayed in degrees, except for the Cobra’s θ_3 that is displayed in millimeters.

		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
ABB	θ_{init}	0	0	0	0	–	–	–
	$\theta_{i,start}$	-90	-60	-30	-90	-90	-90	–
	$\theta_{i,end}$	90	60	90	90	90	90	–
UR3	θ_{init}	0	0	0	0	–	–	–
	$\theta_{i,start}$	-90	-115	-145	-90	-90	-90	–
	$\theta_{i,end}$	90	0	0	90	90	90	–
MICO	θ_{init}	180	270	270	180	180	180	–
	$\theta_{i,start}$	90	180	90	90	90	90	–
	$\theta_{i,end}$	270	270	270	270	270	270	–
Cobra	θ_{init}	0	0	0	0	–	–	–
	$\theta_{i,start}$	-115	-135	-200	-160	–	–	–
	$\theta_{i,end}$	115	135	0	160	–	–	–
KUKA	θ_{init}	90	-90	0	-90	0	90	0
	$\theta_{i,start}$	-35	-105	-70	-110	-70	-35	-70
	$\theta_{i,end}$	105	35	70	30	70	105	70
Sawyer	θ_{init}	90	-90	0	-90	0	90	0
	$\theta_{i,start}$	-35	-105	-70	-110	-70	-35	-70
	$\theta_{i,end}$	105	35	70	30	70	105	70

parallel consecutive joints.

7.2 Application Accuracy

The prime mover behind the development of the stereotactic method was to create a technique to place surgical instruments accurately and repeatedly at a previously agreed set of coordinates, refer to chapter2. It is therefore only logical to assess any system that attempts to be used in stereotaxy in terms of accuracy.

Neurosurgical robotic systems are evaluated based on their application accuracy, a quantifiable measure that represents the distance between the target location and the actual reached location. The application accuracy depends on the compound errors from each system component involved in target identification or tool placement, as well as errors from external sources. Different authors, e.g. Spetzger et al. (2002); Wang and Song (2011); Gerard and Collins (2015) chose to group the causes of system inaccuracy in distinct categories. In an effort to include all the listed factors, we elaborate the following list,

- system mishandling, include errors that result from inexperience or insufficient training (e.g. misplacement of fiducials), which prevent the system operation at optimal conditions;
- anatomical or procedural errors, these englobe causes unrelated to the system operation that result from natural anatomical shifts from the imaging stage to the intraoperative space (e.g. patient/fiducial movements, brain shift);
- image inaccuracies, which involve the errors related to the identification of the surgical target coordinates in the image reference frame; these are dependent on the imaging resolution, susceptibility to distortions and the capacity to distinguish the target structure from normal parenchyma;
- registration errors, encompass the error of the registration method that determines the image-patient

Table 7.2: Denavit-Hartenberg parameters for the studied robots. The measured Denavit-Hartenberg parameters ($a_{\text{meas},i}$, $\alpha_{\text{meas},i}$, $d_{\text{meas},i}$ and $\theta_{\text{meas},i}$) and the differences to the nominal values ($a_{\text{error},i}$, $\alpha_{\text{error},i}$, $d_{\text{error},i}$ and $\theta_{\text{error},i}$) for each joint are presented. The a and d parameters are represented in millimeters, the α and θ in degrees. Values truncated at $1e - 9$.

	i	$a_{\text{meas},i}$	$a_{\text{error},i}$	$\alpha_{\text{meas},i}$	$\alpha_{\text{error},i}$	$d_{\text{meas},i}$	$d_{\text{error},i}$	$\theta_{\text{meas},i}$	$\theta_{\text{error},i}$
ABB	1	70.0007	6.925e-04	90.0000	0.000e+00	352.1216	1.216e-01	0.0000	0.000e+00
	2	359.9906	-9.351e-03	0.0000	0.000e+00	-0.0525	-5.255e-02	90.0006	5.866e-04
	3	0.0622	6.223e-02	89.9765	-2.351e-02	-0.1640	-1.640e-01	0.0000	0.000e+00
	4	-0.0985	-9.850e-02	-89.9765	2.351e-02	379.9459	-5.412e-02	0.0000	0.000e+00
	5	0.0657	6.573e-02	89.9785	-2.153e-02	-0.0005	-4.775e-04	0.0000	0.000e+00
	6	-0.0112	-1.123e-02	0.0000	0.000e+00	65.0010	9.562e-04	0.0000	0.000e+00
UR3	1	0.0853	8.532e-02	-90.0221	-2.212e-02	152.0021	1.021e-01	0.0000	0.000e+00
	2	243.3269	-3.231e-01	0.0008	7.892e-04	112.3699	1.124e+02	0.1516	1.516e-01
	3	213.4347	1.847e-01	-0.0056	-5.630e-03	-0.0344	-3.438e-02	-0.0200	-1.996e-02
	4	-0.0218	-2.184e-02	-90.0030	-3.018e-03	0.0483	-1.123e+02	0.0000	0.000e+00
	5	0.0167	1.673e-02	-89.9977	-1.800e+02	85.3869	3.695e-02	-179.9813	-1.800e+02
	6	0.0000	0.000e+00	0.0000	0.000e+00	81.7164	-1.836e-01	0.0000	0.000e+00
MICO	1	-0.0306	-3.064e-02	-90.0016	-1.606e-03	275.4593	-4.069e-02	0.0000	0.000e+00
	2	290.0117	1.170e-02	179.9931	-6.909e-03	118.0541	1.181e+02	-89.9911	8.900e-03
	3	-0.0326	-3.256e-02	90.0137	1.369e-02	111.0306	1.180e+02	-90.0000	0.000e+00
	4	0.1422	1.422e-01	-59.9959	4.061e-03	-166.1171	-3.541e-02	0.0000	0.000e+00
	5	-0.1094	-1.094e-01	-60.0130	-1.303e-02	-85.4399	1.234e-01	0.0000	0.000e+00
	6	-0.0002	-2.317e-04	0.0000	0.000e+00	-42.7119	6.978e-02	0.0000	0.000e+00
Cobra	1	399.9868	-1.322e-02	0.0001	9.326e-05	204.9906	-9.373e-03	0.0012	1.200e-03
	2	249.9857	-1.428e-02	0.0274	2.737e-02	-0.0001	-6.623e-05	-0.0045	-4.510e-03
	3	0.0000	0.000e+00	-0.0355	-3.554e-02	0.0000	0.000e+00	0.0000	0.000e+00
	4	0.0253	2.532e-02	0.0000	0.000e+00	0.0091	9.134e-03	0.0000	0.000e+00
KUKA	1	0.0001	1.435e-04	-90.0002	-2.417e-04	339.9918	-8.188e-03	0.0001	1.259e-04
	2	0.0128	1.281e-02	90.0004	3.765e-04	0.0002	1.656e-04	-0.0011	-1.086e-03
	3	0.0002	2.023e-04	89.9993	-6.760e-04	399.9947	-5.270e-03	0.0000	0.000e+00
	4	0.0005	4.710e-04	-89.9997	3.212e-04	-0.0074	-7.380e-03	-0.0003	-3.020e-04
	5	0.0333	3.333e-02	-90.0002	-1.683e-04	399.9951	-4.854e-03	0.0000	0.000e+00
	6	-0.0107	-1.066e-02	89.9942	-5.790e-03	-0.0144	-1.444e-02	0.0008	7.840e-04
	7	0.0000	0.000e+00	0.0000	0.000e+00	126.0081	8.147e-03	0.0000	0.000e+00
Sawyer	1	80.2571	-7.429e-01	-89.8740	1.260e-01	316.2051	-7.949e-01	-0.1954	-1.954e-01
	2	0.0573	5.733e-02	89.9338	-6.624e-02	192.6113	1.113e-01	90.0000	0.000e+00
	3	-0.4088	-4.088e-01	-89.9007	9.927e-02	399.8220	-1.780e-01	0.0000	0.000e+00
	4	0.1744	1.744e-01	89.9926	-7.397e-03	-168.8476	-3.476e-01	0.0229	2.285e-02
	5	-0.7611	-7.611e-01	-90.0842	-8.416e-02	400.1897	1.897e-01	0.0000	0.000e+00
	6	-0.1449	-1.449e-01	89.6653	-3.347e-01	135.8429	-4.571e-01	-0.0396	-3.964e-02
	7	0.0000	0.000e+00	0.0000	0.000e+00	132.6210	-1.129e+00	0.0000	0.000e+00

transformation;

- tracking errors, are the errors associated with the tracking system that relates the markers/features tracked in the intraoperative space;
- tool guidance errors, which comprehend not only the errors associated with the precision of the tool guidance system (robot) but also the errors of the guidance system model dimensions.

That being said, and due to the project phase of the proposed robotic system prototype, it would be unrealistic to address all the errors listed above. Since no clinical trials were performed with our system (as of this moment), no information can be provided about the effects of anatomical or system mishandling caused errors. The same can be said for the influence of the tracking errors, as the system does not currently include a tracking device.

Controlled in vitro tests usually constitute the first step to assess the system's accuracy. By testing the system in a structured environment under controlled conditions, we can isolate the sources of error and objectively evaluate the performance of the proposed robotic system. The application error is usually

assessed with a phantom device containing several targets. The phantom is preferably of similar dimensions to the surgery subject, i.e. the human head, with the targets distributed around its volume. The targets are rigidly fixed to the phantom structure by stalks and ideally represent a point in space. Targets are commonly shaped as spheres (Willems et al., 2001; Morgan et al., 2003; Liu et al., 2006), tips (Chan et al., 2009) or divots (Zeng et al., 2017; Batista et al., 2019). The image-phantom registration method also varies depending on the application: i) frame-based (Willems et al., 2001; Kajita et al., 2015), ii) frameless fiducials (Morgan et al., 2003; Liu et al., 2006; Chan et al., 2009) or iii) frameless surface-based (Zeng et al., 2017; Batista et al., 2019). Finally, the application accuracy can be measured with gauges (Morgan et al., 2003; Chan et al., 2009), imaging techniques (MRI or CT) (Schouten et al., 2010; Kajita et al., 2015; Zeng et al., 2017), CMM inspection (Liu et al., 2006) or vision trackers (Batista et al., 2019).

In this section, we present a cost-effective accuracy test to evaluate our prototype robotic system as-is. The image-phantom registration is performed by a touch calibration method and the accuracy is evaluated in vitro using CT-images. The phantom device is designed to match the approximated dimensions of an adult-sized human head and contains several distributed targets of tip-shape. Once the targets are identified in the image space and the registration process is complete, small-sized plastic beads are placed at the target locations. The application accuracy is measured based on the Euclidean distance between the beads and the actual targets.

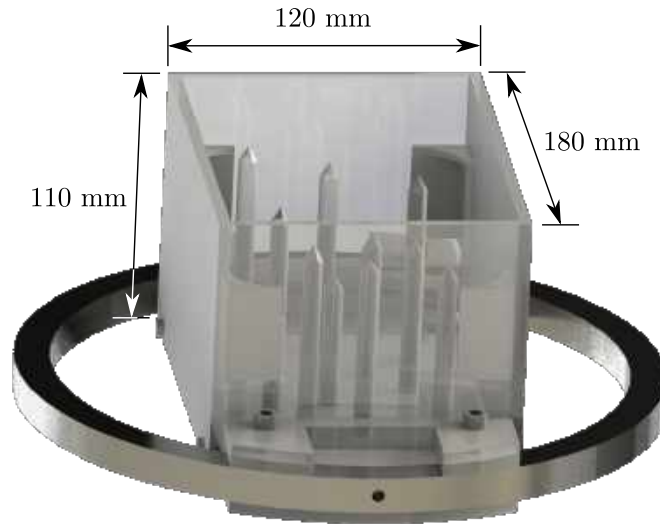
7.2.1 Phantom Design

The phantom was created to provide a basis to reliably assess the system's accuracy, yet also be of simple construction and affordable. Mostly built with acrylic plates and rods, the phantom can either be used in a ZD-frame stereotactic ring, or fixed directly to the operation table. The phantom was designed in a CAD software, see Figure 7.6.

The acrylic plates of 5 mm and 10 mm thickness were machined in an industrial grade laser cutter with 600 dpi resolution (about 0.045 mm resolution) and the acrylic stalks of 8 mm diameter are machined in a lathe with tolerances of ± 0.1 mm, Figure 7.7. Four additional corner pieces are fabricated in PLA to provide better support to the box. The phantom is assembled using acrylic glue and nylon screws so that it can be reused. After assembling, the phantom is sealed to prevent fluid leakage. An agar-agar solution (3%) is prepared and the phantom filled.

7.2.2 Phantom Calibration

The application accuracy is measured based on CT images for two reasons: i) it is the volumetric imaging technique with better spatial resolution and correctness (no errors of imaging distortion), ii) it is used as the standard to determine procedure coordinates. In order to rule out the errors from fabricating and assembling the phantom a pre-imaging procedure was conducted. The objective of the pre-imaging procedure is twofold: first, to determine the real positions of the interlocks and error of the interlock calibration process, and second, to determine the real position and orientation of each target.



(a) Opened phantom attached to the ZD-frame stereotactic ring.



(b) Closed phantom with 3 female interlocks used for registration.



(c) The 11 target stalks of the phantom box.

Figure 7.6: Phantom box concept.

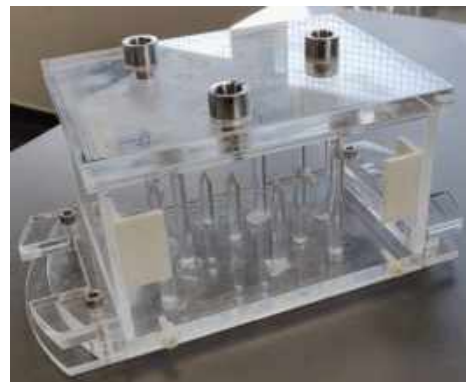
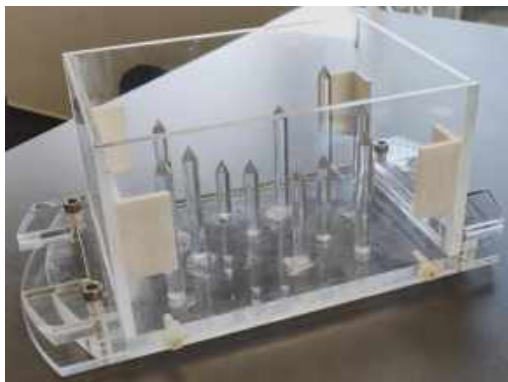
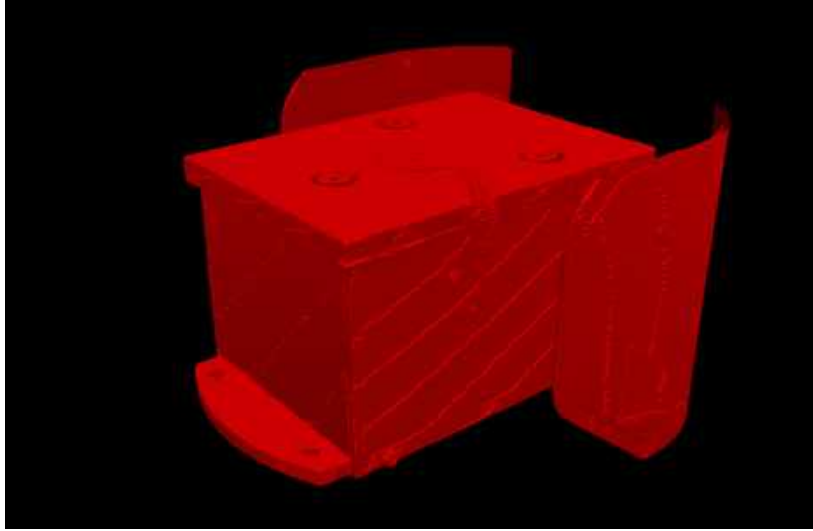
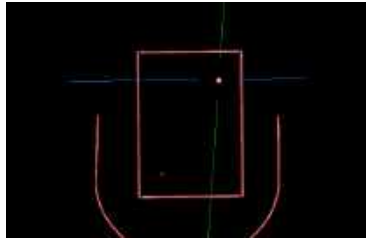


Figure 7.7: Assembled Phantom Box.

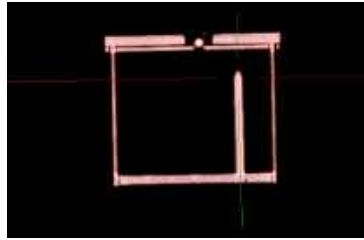
The phantom was scanned in a Siemens SOMATON Definition As (Siemens AG, Munich, Germany) CT machine with a field of view of 170 mm and a matrix of 512×512 in a sequential acquisition with a thickness of 0.6 mm and an angulation of 0° . To obtain the phantom shape in CT-image coordinates, an interval threshold was applied to the image dataset to differentiate the acrylic material, Figure 7.8. The output STL file containing the phantom polygon mesh was post-processed to remove the CT head supports, which have a similar Hounsfield value to the acrylic.



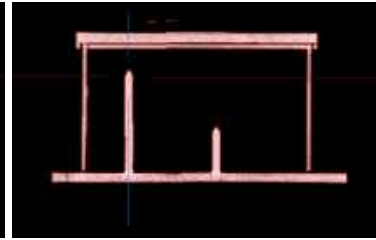
(a) Polygon mesh created after the interval threshold.



(b) Axial view of mesh outline.



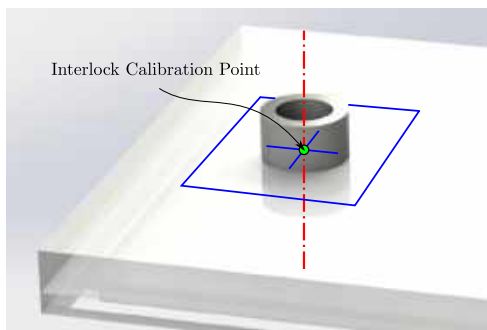
(c) Coronal view of mesh outline.



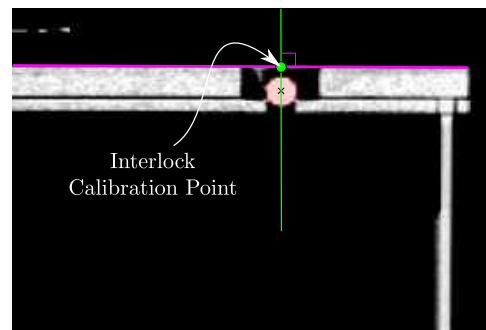
(d) Sagittal view of mesh outline.

Figure 7.8: Pre-imaging procedure of the phantom.

A workaround was required to determine the CT-image coordinates of the interlocks since they are fabricated in rectified carbon steel, which generates significant CT scan artifacts. The registration point of each interlock is determined at the intersection of their axis, and the top surface of the phantom, Figure 7.9a. Instead of the interlocks, CT-ready contrast markers/spheres were placed in the interlock slots, Figure 7.9b. Thus, the interlock calibration points are found by projecting the contrast spheres' center points at the phantom top surface, along the top surface normal, Figure 7.9.



(a) Interlock point in phantom.



(b) Interlock point determined from the contrast sphere position in the CT dataset.

Figure 7.9: Interlock point.

First, the phantom's top surface is determined by a manual 2D segmentation of voxels selected in the

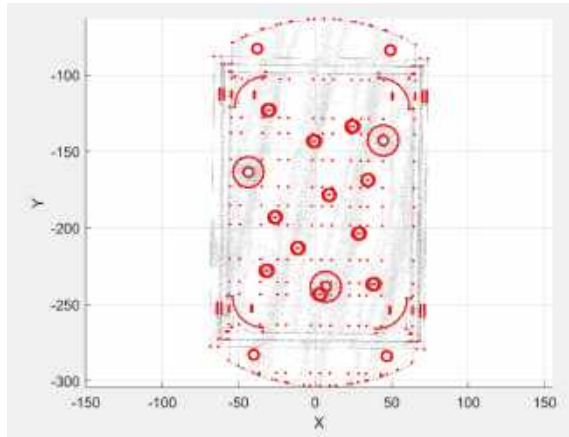
CT dataset. The coordinates of these top surface voxels are transformed into a covariance matrix, which is processed by a singular value decomposition (SVD) algorithm to determine the plane normal corresponding to the least singular value. Second, the contrast spheres are segmented using a 3D interval thresholding algorithm. The point cloud formed by the voxels of each sphere is passed to a least-squares fitting algorithm. The algorithm minimizes the sum of the square root distance between the sphere points and the surface of a sphere with 4 mm radius (nominal dimensions of the CT contrast sphere). Finally, the center position of the least square fitted spheres are projected to the best fitting phantom's top surface plane along the plane normal, to determine the interlock points in image space, Figure 7.9b.

The first objective of the phantom calibration procedure was to determine the interlock calibration error. The phantom mesh extracted from the CT dataset was registered to the mesh of the phantom's model. The CT mesh is specified relative to the image coordinate system, while the model mesh is described in the world coordinate system. To register the image and world space two methods were applied: i) the interlock calibration, based on Horn's algorithm Horn (1987), refer to subsection 4.3.2; and ii) the ICP (iterative closest point). Both methods return a rigid transformation that minimizes the distances between the set of points in image and world coordinates. The difference lies on the set of points used to calculate this transformation; while the interlock calibration method relies on the 3 interlock points from both spaces, the ICP method uses the all the vertices from the image and model meshes, see Figure 7.10.

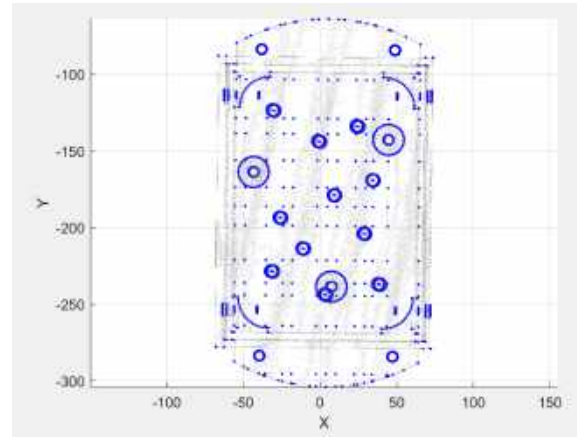
As depicted in Figure 7.10, the transformation calculated from the ICP method permits a closer overlap of the image and world point clouds and is therefore considered as the reference. By multiplying the transformation obtained from the interlock calibration method with the inverse matrix obtained with the ICP method, we obtain the residual error matrix, i.e. the error of the interlock calibration method.

The second objective of the phantom calibration method consists in the determination of the phantom's target coordinates in the world reference frame. One approach to determine the real target coordinates would be to manually select the voxels at the tip of each stalk and then transform these coordinates from the image reference frame to the world reference frame. It is also relevant to note that the CT dataset has a voxel size of $0.56 \times 0.56 \times 0.60$ mm. As a discrete representation of the phantom, the Hounsfield value at acrylic/air boundary voxels corresponds to the ratio of each material occupying that voxel. After the 3D interval segmentation performed previously to differentiate acrylic from air, the voxel will be considered as acrylic material if the Hounsfield value is between the specified threshold. In reality however, only part of the voxel should be considered as acrylic. To deal with this phenomena and also to obtain the target coordinates with sub-pixel precision, we perform a local registration of each stalk model (from design) to the real stalks in image space. After transforming each model stalk to the corresponding real stalk, the target coordinates are determined from the model stalk tip.

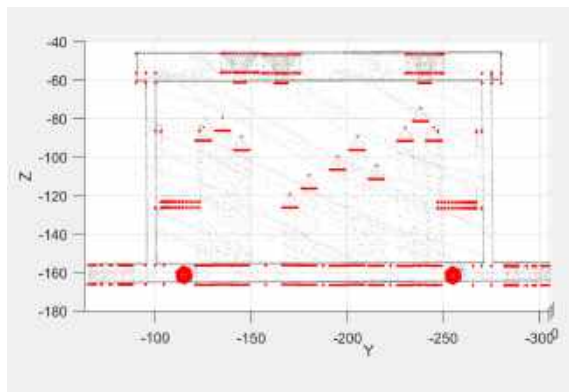
For each target, two points are saved, i) the tip point at the extremity of the stalk and ii) the real target point at 5 mm from the tip of the stalk along the vector coaxial to the stalk (black ring and green dot respectively in Figure 7.11a).



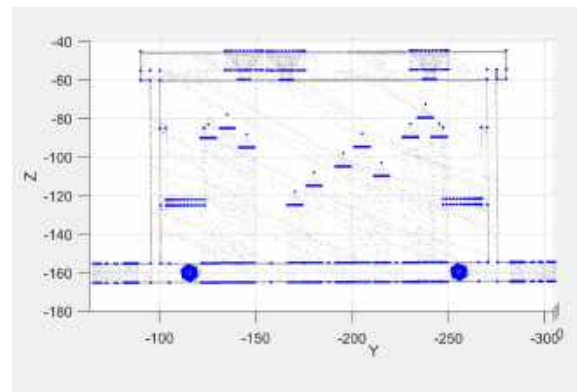
(a) Top view of interlock calibration.



(b) Side view of interlock calibration.



(c) Top view of ICP matching.



(d) Top view of ICP matching.

Figure 7.10: Phantom meshes after registration with the interlock calibration (red) and the ICP (blue). The CT phantom point cloud is represented by gray-colored dots and the model point cloud is either represented by red-colored dots for the interlock calibration, and blue-colored dots for the ICP calibration.

7.2.3 Marker Placement

After the calibration of the phantom and determination of the target coordinates, we proceeded to place small markers at the planned location. The markers are plastic beads of 2.3 mm diameter and 1.5 mm thickness (Figure 7.13). One of the cannulas of the microdrive system (refer to subsection 5.6.2) was adapted to hold a marker at its tip.

Once the phantom container is filled with and agar-agar solution (3%), the phantom-robot registration is performed using the interlock calibration. The end-effector that holds the microdriver is then attached to the robot and the stereotactic surgical procedure is performed as described in subsection 5.8. The robot does an initial approximation to the surgical trajectory and then moves the microdrive along this trajectory to the target depth, Figure 7.13. As noted in the previous subsection 7.2.2, the real target is located 5 mm above the stalk tip. By setting a target with no physical body, we prevent the possibility of the marker colliding with the target and shifting its position.

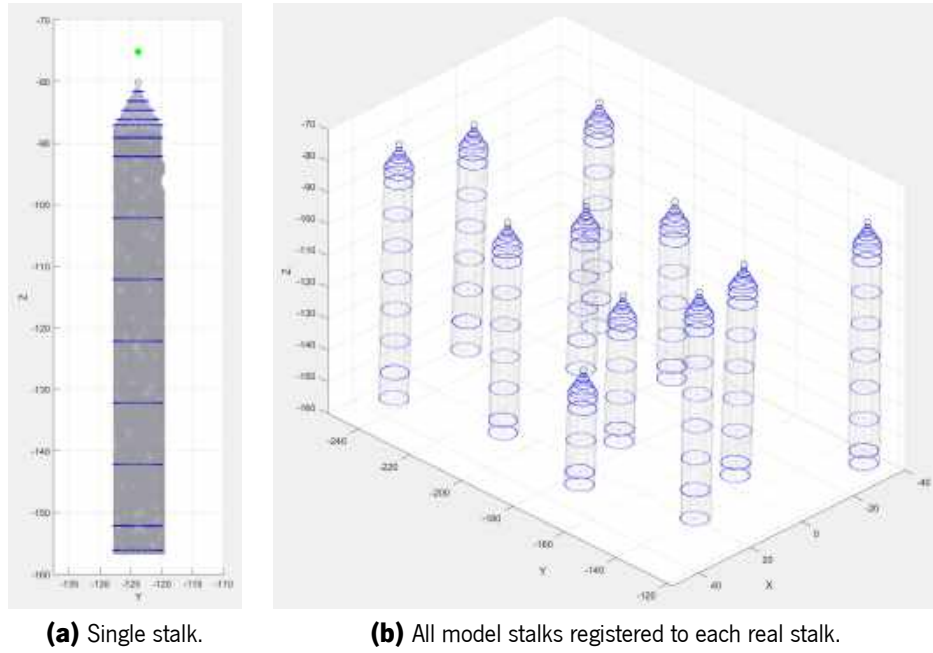


Figure 7.11: Model stalks locally registered to the stalks in the CT dataset. Model stalks' outline is represented by the blue dots while the real stalks from the CT image are represented by a gray shape/dots.

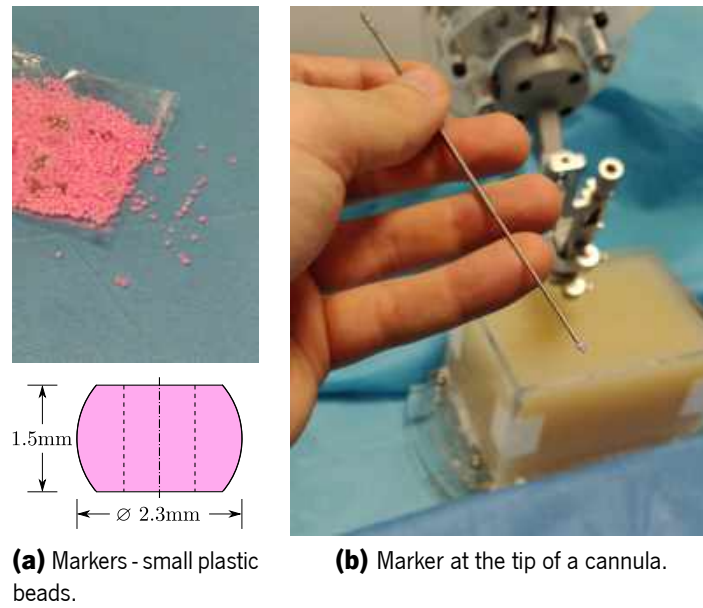


Figure 7.12: Marker placement equipment.

7.2.4 Precision Assessment

Once every marker is placed at the planned position the phantom is then placed in a CT scan again to assess the system's accuracy. For consistency, the same scanner and settings as described in subsection 7.2.2 are used. Interval segmentation was again applied to differentiate the acrylic. The CT contrast spheres and small bead markers were individually segmented using 3D region growing, Figure 7.14.

The meshes of the phantom and of the markers are separately imported. Similar to the process

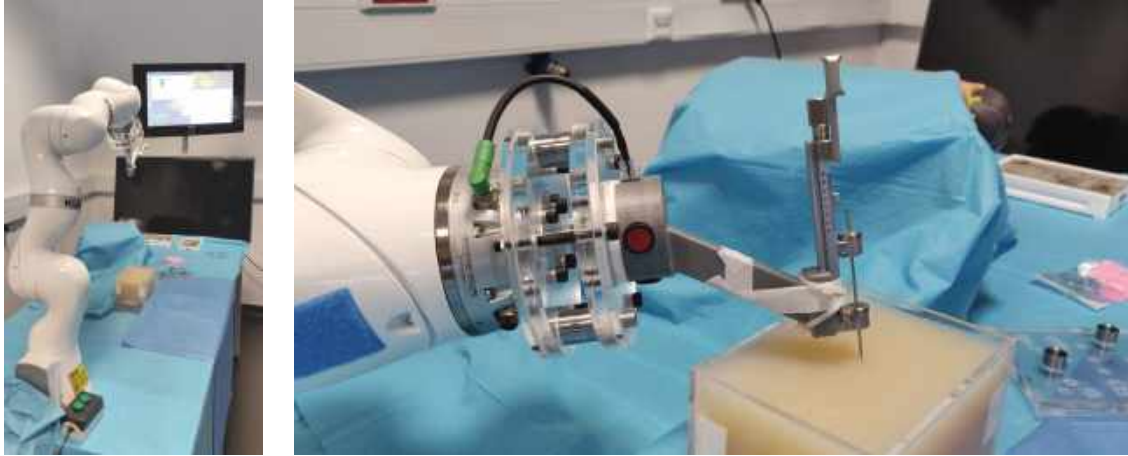
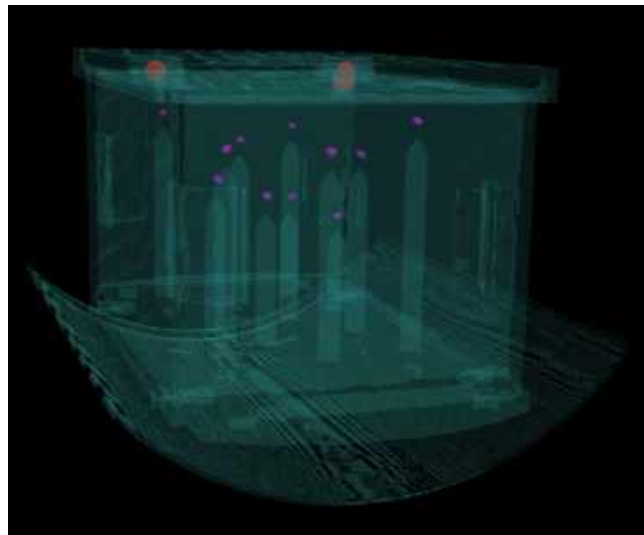
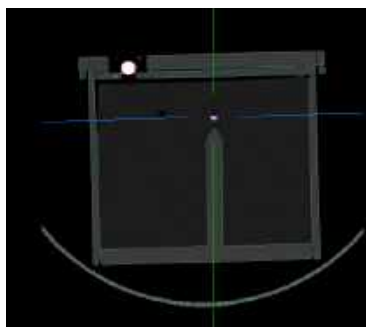


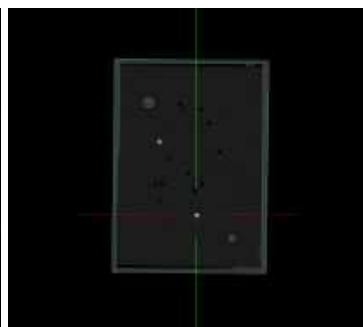
Figure 7.13: Marker placement setup and example.



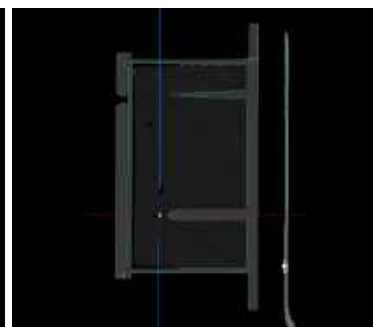
(a) Semi-opaque 3D representation of phantom. CT-contrast spheres are red-colored and the markers are depicted in magenta.



(b) Axial view.



(c) Coronal view.



(d) Sagittal view.

Figure 7.14: Post marker placement CT image scan of the phantom.

previously described for the phantom calibration, the stalk tip, as well as the target 5 mm above the tip, are determined from the model of each stalk. Each model is locally registered to its shape in the CT scan using an ICP algorithm. The nominal position of each stalk is provided as the initial transformation for the ICP algorithm.

The marker's center position is determined by fitting a 2.3 mm diameter sphere to each marker mesh. A least squares fitting algorithm is applied. The final position of every stalk and marker is depicted in Figure 7.15.

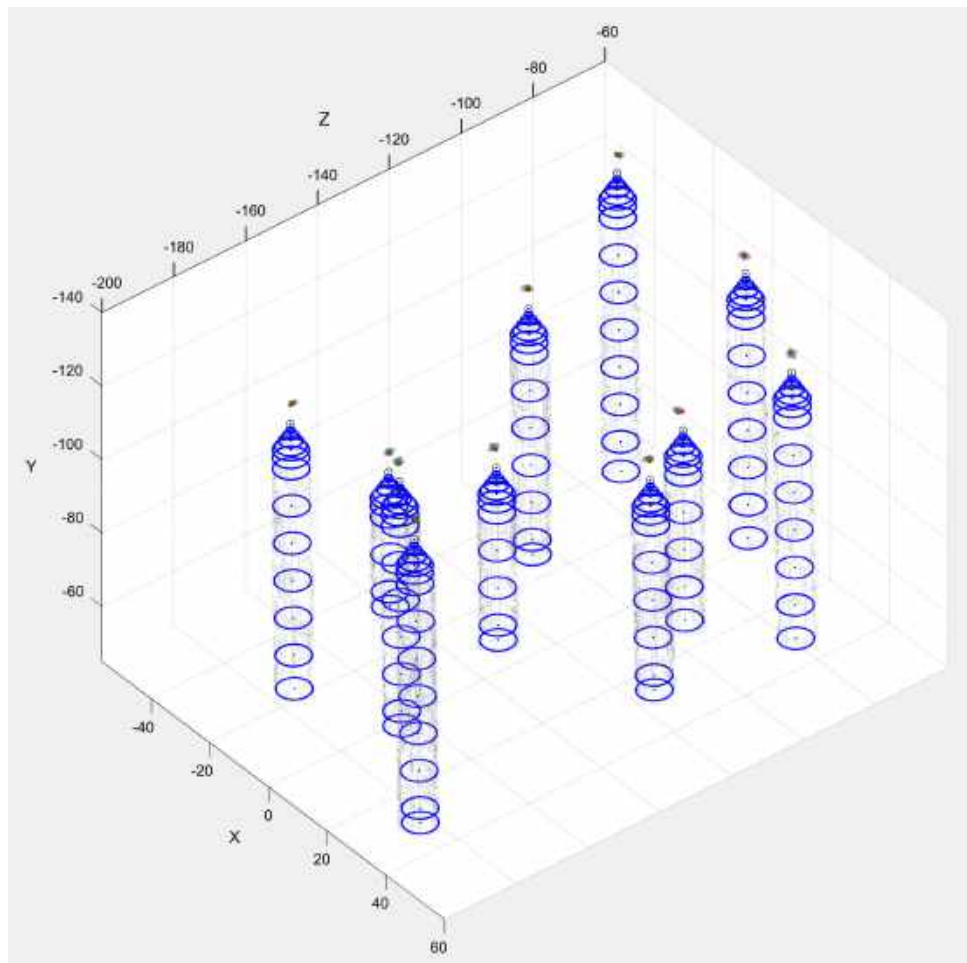


Figure 7.15: System accuracy results. The model stalks (blue outline) are registered to the image stalks (gray dots) and the target, 5 mm above each stalk tip, is represented by the red color while the center position of each marker is identified by a green dot and a black outline.

Out of the 11 targets, the Euclidean distances between the marker's center position and the stalk's target measured an average error of 0.56724 mm with a standard deviation of 0.26547 mm and a maximum error of 1.0944 mm.

Chapter 8

System Diversification

One of the advantages of developing a complete high-level controller rests in the possibility to branch out the system to other, similar applications. In this chapter, we elaborate on two potential applications for our system.

First, the possibility to integrate open-skull surgeries using a stereo-camera system to continuously track the movement of the surgical field and adjust the robot pose accordingly (Faria et al., 2014). In section 8.1, we elaborate on the developed system for measurement and tracking of brain deformation during surgery, a solution that could integrate the proposed architecture to extend its application field.

Second, a new operation mode is developed in the context of the proposed robot control architecture. Two additional components are created to interface the robot controller with an haptic feedback 7-DoF joystick to maneuver the robotic arm. Although the applications of tele-operation reach far and wide, the proposed add-on is being tested in kidney stone removal procedures (Rodrigues et al., 2013; Lima et al., 2017).

8.1 Quantification and Tracking of Brain Deformation

Brain shift represents a major source of error in image-guided neurosurgery (Hill et al., 1998; Hata et al., 1999; Roberts et al., 1998). The skull-fixation system that assures the rigid transformation between preoperative images and the patient coordinated space does not account for the brain deformation relative to the skull (Grunert et al., 2003). In frameless neurosurgery, the registration of the intraoperative space to the imaging scans relies on external landmarks (skin surface or fiducial markers) (Eggers et al., 2006).

There are two sources for brain tissue deformation reported in open skull neurosurgery. The first concerns brain shift and results from a combined effect of: tissue resection, pathology, pharmacologic responses, gravity, edema, and loss of cerebrospinal fluid (Cohen-Adad et al., 2006). Hill et al. (1998) measured the brain surface deformation in 21 patients under a craniotomy and reported a mean cortical surface displacement of 5.6 mm and a maximum displacement over 10 mm.

Another source of brain deformation noticeable in open skull surgery is due to systematic blood pressure variations, breathing and vasomotor oscillations, which result in a pulsating brain motion (Linninger

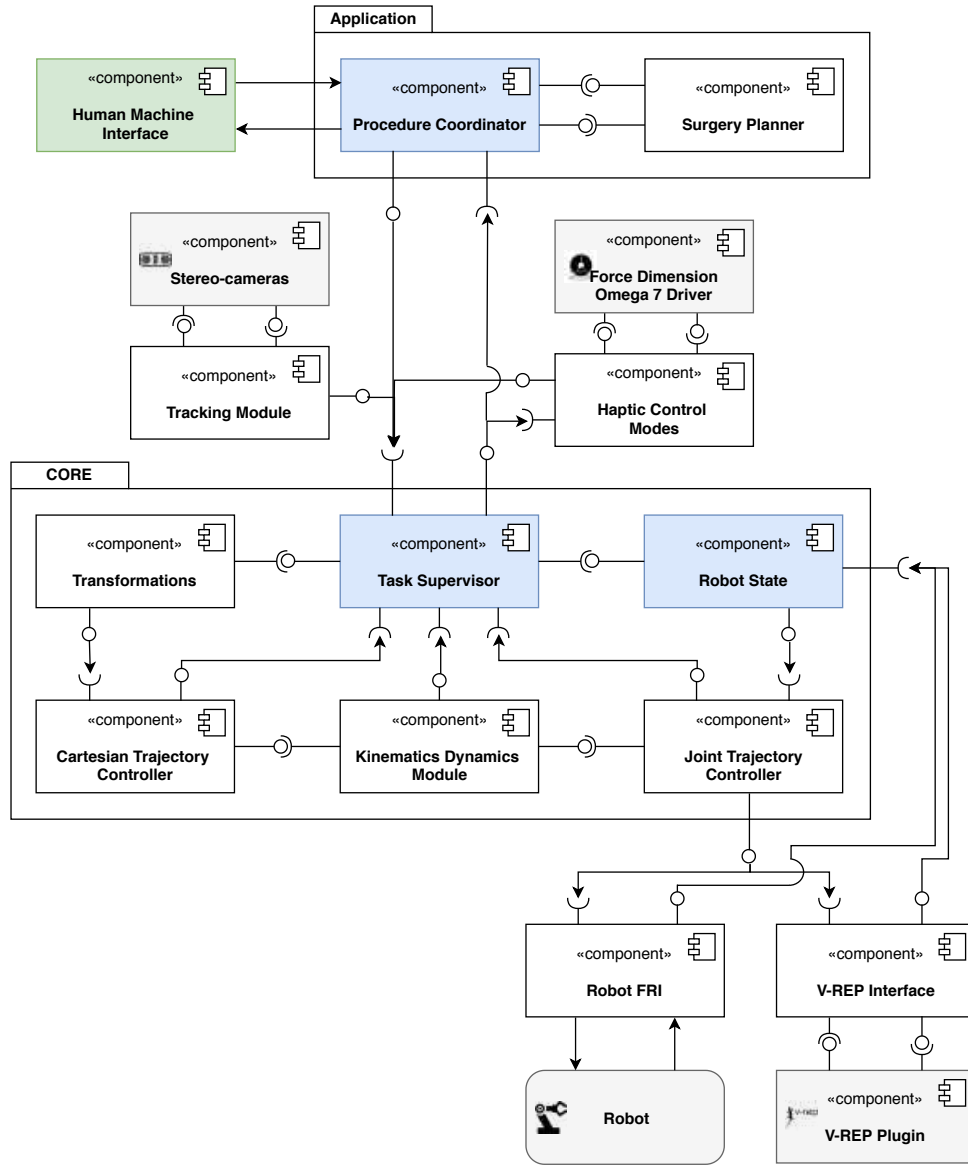


Figure 8.1: Schematic representation of an Extended Modular Architecture.

et al., 2005; Wagshul et al., 2011). Britt and Rossi (1982) quantified the brain pulsation motion of a mammal brain and divided its components into arterial component with a displacement 110-266 μm and a motion frequency between 2-3 Hz, and a respiratory component with a displacement of 300-950 μm and a frequency between 0.1-0.2 Hz. Scarce information was found related to the characteristics of pulsation/breathing motion of the human brain. Wagshul et al. (2011), presents a frequency analysis of brain pulsatility based on intracranial pressure and photoplethysmography, whose main components range between 1-4 Hz. Poncelet et al. (1992), reported a pulsating displacement amplitude to be at most 0.5 mm.

Brain shift is a progressive condition with a very low displacement velocity, thus not requiring a high sampling rate for its quantification. Several approaches have been proposed to measure brain shift: Magnetic Resonance (Greitz et al., 1992; Hill et al., 1998; Black et al., 1999; Nimsky et al., 2000; Hadani et al., 2001), Computed Tomography (Lunsford et al., 1996), Ultrasonography (Koivukangas et al., 1993; Jödicke

et al., 1998; Comeau et al., 2000; Tronnier et al., 2001; Unsgaard et al., 2002), biomechanical/predictive models (Skrinjar et al., 2002; Sun et al., 2003; Cohen-Adad et al., 2006; Jannin et al., 2006; Wittek et al., 2007), or methods that use a priori knowledge of a brain biomechanical model to constrain the registration of 3D intraoperative imaging (Ferrant et al., 2001). However, the computation times, cost of technology and necessary infrastructures as well as the cumbersomeness of user-interface rendered most of these solutions inapt (Paul et al., 2009). In neurosurgeries involving craniotomy, cortical surface can be used to infer brain deformation. Two main research areas stand out: laser range scanning (Sinha et al., 2005; Miga et al., 2003; Cao et al., 2008) and stereoscopic systems (Sun et al., 2005; Paul et al., 2005). However, the low sample rate of laser range scanner systems enables its use to measure brain shift but not brain pulsation deformation.

The process of tracking a surface with a stereoscopic system is a two step process: 1) extraction of 3D positions of salient visual features on the surface and 2) tracking of these features in video frames at different times. Sun et al. (2005) presented a system able to measure brain deformation and pulsation with a stereoscopic microscope (capturing 15 frames per second) achieving a point-localisation average error of 1.02 mm when compared to a optical stylus probe of a Polaris System and capturing the cortical surface shape of a phantom model with an average error inferior to 1.2 mm when compared to a CT scan. In 2011, Ji et al. (2011) reported a stereo vision system to measure cortical surface strain based on differentiation of brain motion using optical flow, with high resolution (within 0.05 to 1 mm) although no explanation is given about how sensitivity is calculated. Later in 2013 Ji et al. (2013) presented a noninvasive method, based on the approach reported in Ji et al. (2011), to produce a dense map of cortical surface deformation due to blood pressure and respiratory cycles in patients undergoing brain tumour resections via craniotomy. The breakthrough from the previous work is based on the use of an average cortical surface deformation to compute the relative cortical deformation instead of considering an arbitrary measurement as the reference state. Ji et al. (2013), also presents a FFT analysis of brain deformation from a clinical case. These papers also report surface cortical displacement up to 1 and 2 mm but none conducted system accuracy or validation tests.

We present a validation method to evaluate the static and dynamic accuracy of a stereo camera. A simple phantom is simultaneously tracked by the stereo camera and by a commercial high-accuracy optical tracking system used as gold-standard reference. Results provide further information about the reliability of stereoscopic systems - not only - in the context of brain shift/deformation (as usually reported in surface tracking papers) but more important, for measurements of brain pulsatility.

To measure the brain motion we use a stereo camera to extract, and robustly and continuously track features on the cortical surface. Three brain motion acquisitions were recorded intraoperative.

8.1.1 Stereo vision system

The stereo vision system includes: a camera module, communication channels, a personal computer and the tracking software. The stereo camera module consists of two mvBlueFox 223C CCD cameras (MATRIX VISION GmbH, Oppenweiler, Germany, $38.8 \times 38.8 \times 58.5$ mm body dimensions) each equipped with

SM25WI 2/3" lenses (Rainbow CCTV, Irvine, Canada, 25 mm 1:1.4 C-Mount, manual iris/focus). The cameras are able to record 1360×1024 resolution, at a maximum frame rate of 20 fps, although the USB 2.0 connection limits the data transfer of full resolution images to 10 fps. The computer with the tracking software runs Microsoft Windows 7 64-bit (Intel i7-3770S CPU, 8 GB of RAM), see Figure 8.2.

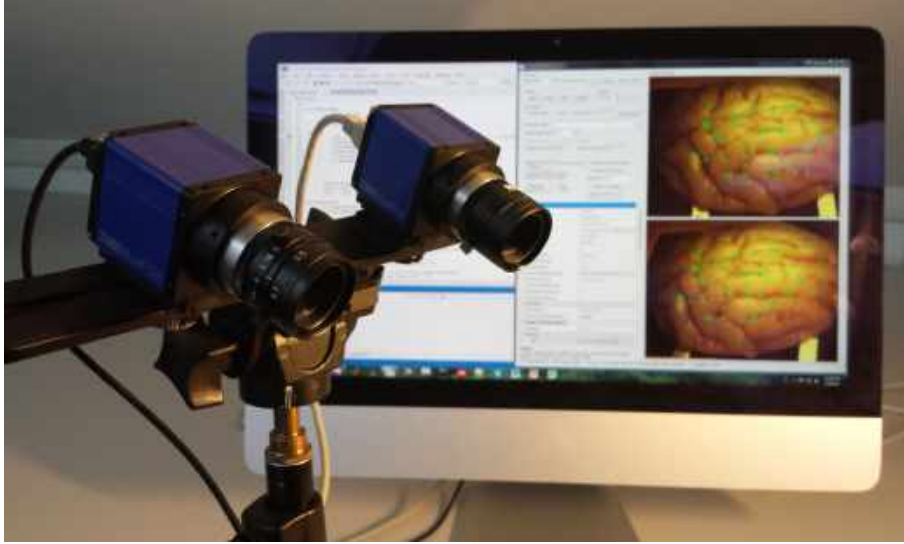


Figure 8.2: Stereo vision system setup. Stereo camera fixed on a bracket, connected to a computer via USB 2.0.

The tracking software consists of an image acquisition and a tracking module. The tracking module can be split into several components: i) feature detection and stereo matching, ii) stereo video tracking, iii) 3D reconstruction.

Our software includes the following features:

- Simultaneous operation of multiple cameras (two, in this work).
- A user-controlled synchronised image capture cycle, which is critical for correct tracking.
- Saving the acquired images in video files.
- Encoding a timestamp into the acquired images to facilitate off-line synchronisation.
- Integration with feature tracking and reconstruction algorithms.

8.1.1.1 Camera Calibration

A calibration of the stereo camera is required prior to its use. The underlying camera model is taken from the OpenCV library and includes a pin-hole projection matrix, radial and tangential lens distortion and a rigid transform relating the two cameras in a stereo system. To ease the calibration process, we use a coded checkerboard pattern from the CALTag toolkit (Atcheson et al., 2010), in which the coordinates of every corner are automatically identified from embedded binary codes, and is thus robust to rotation, occlusion or partial views. Snapshots of the pattern are taken at several different poses (typically 15 to 20) and processed automatically to compute all the calibration parameters.

8.1.1.2 Tracking Software

The first step of the tracking algorithm is feature detection. To accelerate the process, we start by asking the user to specify approximately corresponding rectangular regions of interest (ROIs) on the left and right images. We then execute multiple filters on the acquired left image:

1. Convert the image to a single channel, e.g. from colour to grey.
2. Apply an intensity threshold above which pixels are suspected as specular reflections and not used for features to prevent tracking false visual items.
3. Select “interesting” objects which stand out relative to their neighbourhood. For example, cortical blood vessels are typically darker than the surrounding brain tissue, and therefore we seek for pixels darker than their neighbourhood.
4. To locate specific features, we use the Harris corner detector (Harris and Stephens, 1988) on the left camera image. It finds features with high pixel intensity derivatives in two orthogonal directions.

To avoid clustering all features in a small region, we subdivide the original ROI and attempt to locate several features (e.g. 4) in each rectangular subregion, which increases the area of coverage within the ROI. The total number of tracked features is determined by a specified ceiling amount (typically 100) and an acceptance threshold on the sensitivity of the Harris corner detector filter. The sensitivity parameter can be lowered to increase the number of detected corners present in the image, up to specified ceiling. Compared to the optical flow approach in Ji et al. (2013), which reconstructs a dense displacement map of the surface, we use a relatively sparse set of features. The small number of features allows real time tracking. Notice that NCC is robust to variations in the intensity gain of the patch, especially when compared to the constant intensity of a feature presumed by the optical flow method in Ji et al. (2013). One case where this factors in is when a shadow, not an occlusion, is cast over the tracked region.

Every feature is identified by a small patch (e.g. 15×15) of pixels in its neighbourhood. This patch serves as a template for searching the feature in another image. The matching computes the normalised cross correlation (NCC) (Lewis, 1995) with the template patch over a search region which is described below. The location of maximal NCC score is chosen as a first approximation. Since the physical motion of the tracked feature is not limited to integer pixel units, we refine this approximation by fitting a quadratic surface to a 3×3 neighbourhood of the first guess and taking its apex (location of zero gradient) as the position of the match.

Feature matching is used in two stages of the algorithm: between the first left-camera image (image index $i = 0$ in the sequence) and the first right-camera image (we call these “stereo matches”), and between subsequent images (indexes i and $i + 1$) from the same camera - “video matches” (see Figure 8.3). For each stage, we define a different search region. For stereo matching search, we use the presumed correspondence between left and right ROI rectangles plus a user-specified volume depth estimation to allow for some tolerance. We reconstruct by triangulation (section 8.1.1.3) the four corners and then add and subtract the depth parameter from the Z coordinate to receive eight corners of a box. These are

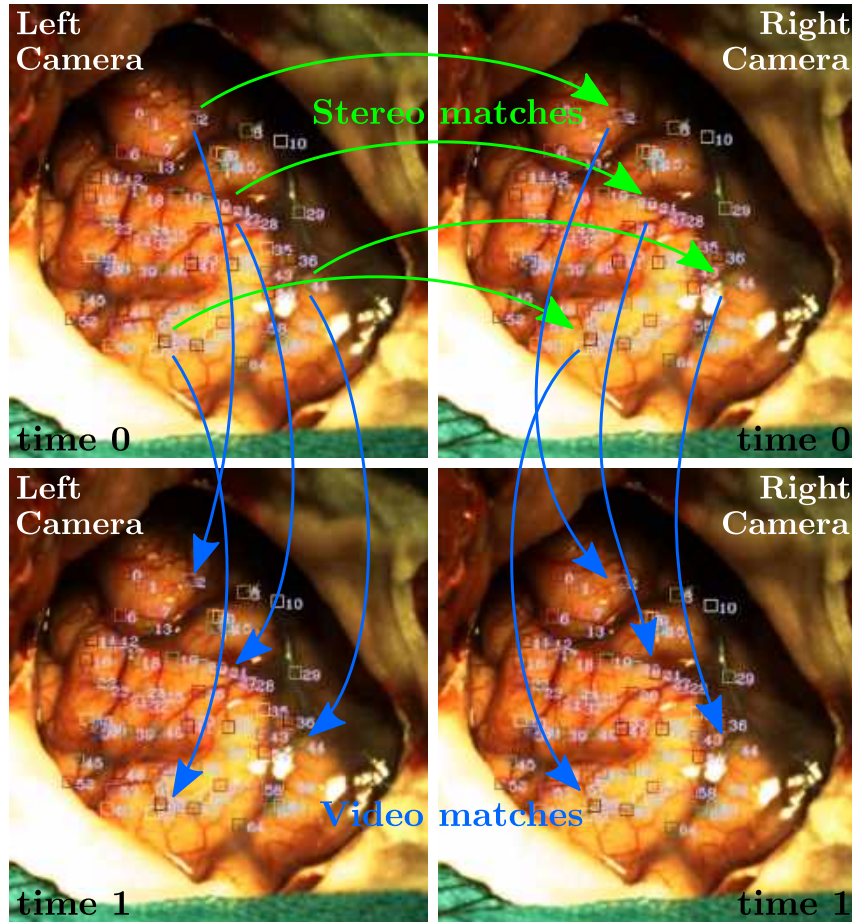


Figure 8.3: Stereo and video feature matching. Each row shows a video frame, and columns stand for the left and right cameras. Stereo matches are depicted by green arrows from the left to the right camera at time 0, and blue arrows show video matches for both cameras from time 0 to time 1. Each feature neighbourhood is depicted as a small numbered square.

reprojected to the image planes and the maximal offsets between these corresponding projections define the search range. The video matching search region is a (small) neighbourhood of the current feature position, based on the assumption that the motion of the feature between frames is limited.

False matches, or outliers, must be removed from the data to achieve an accurate reconstruction of the visible surface. We identify outliers with the following criteria. First, image similarity measures of NCC and normalised mutual information (NMI) (Studholme et al., 1999) are computed between the template and the match candidate, and a minimal score acceptance criterion is used. Empirically, we found that thresholding the NCC alone is insufficient to detect some occlusion cases, and the NMI criterion adds strength to it. Then, for stereo matching, two geometric constraints are tested: (1) the distance of the right-image feature from the epipolar line that is the projection of the left-image feature on the right image plane; and (2) a parabolic surface is fitted to the collection of 3D reconstructed features (see below) as a rough approximation of the brain surface shape, and points too far from this surface are considered as outliers.

8.1.1.3 3D Reconstruction

The pin-hole projection model is based on the following equation

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \mathbf{A}(\mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}), \quad (8.1)$$

where λ is an arbitrary scale; u and v are the coordinates of a pixel; $\mathbf{A}$ is a 3×3 projection matrix (with nonzero elements in the diagonal and the last column); $\mathbf{R}$ is a 3×3 rotation matrix; and $\mathbf{t}$ is a translation. $\mathbf{A}$ is constant per camera, and $\mathbf{R}$ and $\mathbf{t}$ express the pose of the camera relative to the imaged object which is acquired in a calibration process and can involve the stereo relation between the cameras.

The back-projection of a pixel (line L) is henceforth computed as

$$L = -\mathbf{R}^{-1} \cdot \mathbf{t} + \lambda \mathbf{R}^{-1} \mathbf{A}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (8.2)$$

where the first term is the position of the camera's principal point relative to the imaged object, and the second is the direction of the ray that pierces the pixel. In short notation, we will write $L = \mathbf{p} + \lambda \mathbf{g}$ and assume that $\mathbf{g}$ is a unit-norm vector.

Given two pairs of pixel coordinates, (u_1, v_1) and (u_2, v_2) , which presumably are projections of the same point $\mathbf{x}$ from an object, the 3D position of $\mathbf{x}$ is reconstructed by intersecting the two rays. Since the rays usually do not intersect (because of localisation ambiguity and numerical roundoff), we seek for a point that is exactly mid-way along the shortest line segment that connects the two rays. This line segment is also perpendicular to both rays. The perpendicular projection of $\mathbf{x}$ on a line $L = \mathbf{p} + \lambda \mathbf{g}$ is

$$(\mathbf{x} - \mathbf{p}) - ((\mathbf{x} - \mathbf{p})^T \mathbf{g}) \mathbf{g} = (\mathbf{I} - \mathbf{g} \mathbf{g}^T)(\mathbf{x} - \mathbf{p}).$$

And the midway point satisfies the equation

$$(\mathbf{I} - \mathbf{g}_1 \mathbf{g}_1^T)(\mathbf{x} - \mathbf{p}_1) + (\mathbf{I} - \mathbf{g}_2 \mathbf{g}_2^T)(\mathbf{x} - \mathbf{p}_2) = \mathbf{0} \quad (8.3)$$

which is easily solved.

To address the radial and tangential lens distortion of the OpenCV camera model, we use the solution of the linear equation as an initial guess in Levenberg-Marquardt optimisation to minimise the squared difference between the projection of $\mathbf{x}$ on the respective image and the located pixel coordinates of the feature.

8.1.2 Materials

To validate the stereoscopic system (Figure 8.4), a custom designed plate was synchronously tracked by the stereo camera and an industrial optical tracking system considered as the gold standard reference for this project: Optotrak CERTUS (NDI, Ontario, Canada). The choice of the optical tracker was held based on the reported 3D tracking accuracy 0.1 mm (system datasheet).

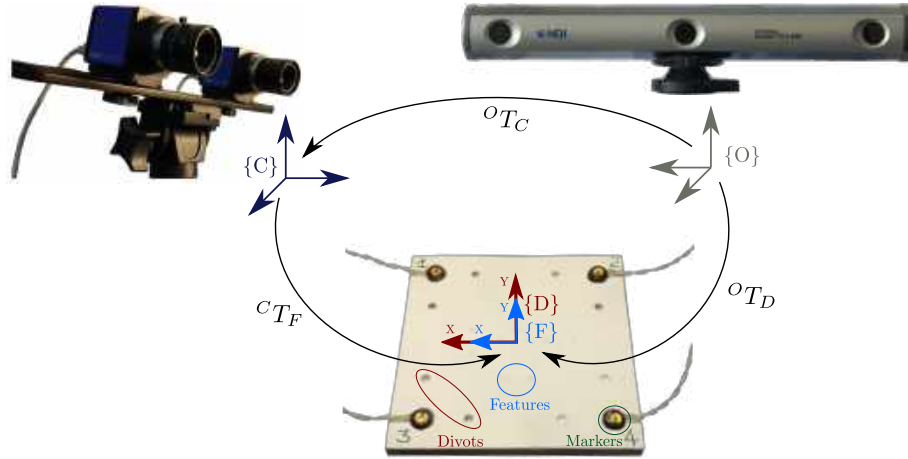


Figure 8.4: Validation setup. The three reference frames: stereo camera {C}, optical tracking system {O} and custom designed plate which includes the features' reference frame {F} and the divots' reference frame {D}, which are overlapped to each other. ${}^C\mathbf{T}_F$ represents the transformation from {F} to {C}, ${}^O\mathbf{T}_D$ from {D} to {O} and ${}^O\mathbf{T}_C$ from {C} to {O}.

8.1.2.1 Synchronisation Logic

To validate the stereo camera against the reference system, both systems must track the same or a related object synchronously. The synchronisation between both systems was achieved through a logic hardware component connecting each system's IO modules and modulating the output signal from CERTUS to trigger the stereo camera (Figure 8.5).

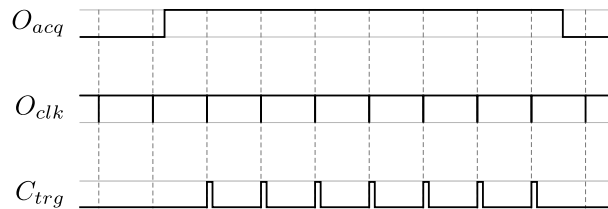


Figure 8.5: External signals from CERTUS: 1) O_{act} is high while the markers' positions are being recorded and 2) O_{clk} goes low for $1\mu s$ with each acquisition of the marker's positions. The input signal of the stereo camera C_{trg} consists of a high-level pulse synchronous to the CERTUS clock (O_{clk}) when it is recording (O_{act} is high).

The short clock acquisition pulse from CERTUS occasionally caused the cameras not to trigger. A monostable multivibrator (HCF4047BE) was included in the logic component to increase the duty cycle of C_{trg} and avoid not-detection problems.

8.1.2.2 Custom Plate

The custom designed plate (Figure 8.6) was built from a 80×80 mm plexiglass and machined with a PowerWASP EVO 3D printer (WASProject, Massalombarda, Italy), which has a 0.1 mm motion resolution.

The plate includes three groups of elements:

- Features, small drilled conical holes of 0.5 mm diameter and depth. These details have high pixel intensity derivatives along two orthogonal axes, which make them salient visual features easily recognised and tracked by the stereo camera's corner-detector algorithm.
- Markers (or active-LEDs), are the elements fixed on the corners of the plate (Figure 8.4), tracked by CERTUS. A rigid body or dynamic reference frame (DRF) $\{M\}$ was created with the markers using the 6D Architect software (NDI, Ontario, Canada) and is recorded, instead of the position of each single marker.
- Divots, are 3 mm etched hemispherical holes with the centre co-planar to the plate's DRF xy -plane, and are used to calibrate the plate's DRF $\{P\}$ (Figure 8.6) to the markers' DRF.

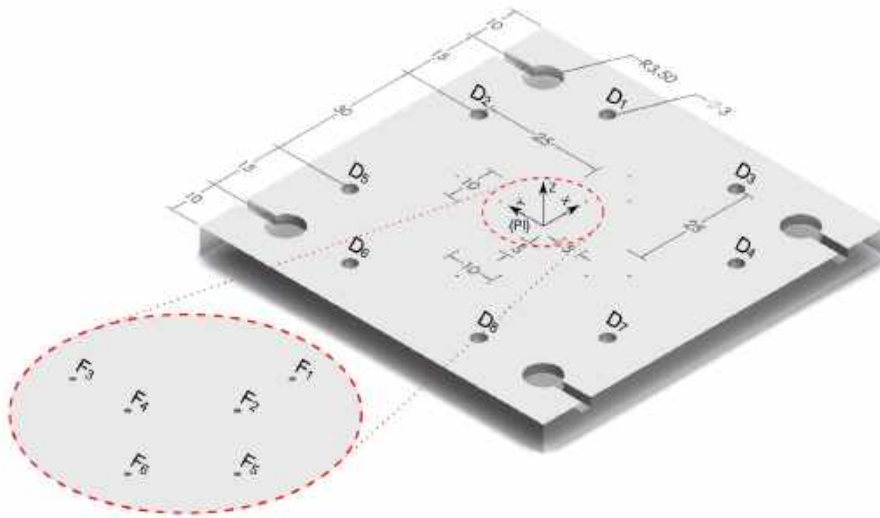


Figure 8.6: Custom designed plate symmetrical along the x and y -axis of its dynamic reference frame, denominated $\{P\}$. Plate dimensions depicted with a “zoom in” of the region with the six tracked features. The divots' $\{D\}$ and features' DRF $\{F\}$ are assigned to be coincident to the plate's DRF $\{P\}$.

The relation between features tracked by the stereo camera and markers tracked by CERTUS was inferred through the plate layout. To avoid the marker localisation error (distance between its true position and the layout position on the plate's DRF $\{P\}$) that depends on the fixation of each marker to the plate, divot sockets were used to compute the transformation from $\{P\}$ to the markers' DRF, since divot positions can be directly measured and they are in the xy -plane of $\{P\}$ like the features (c.f. section 8.1.3.1).

The nominal homogeneous coordinates ($\mathbf{p}_i = [x_i \ y_i \ z_i \ 1]$) of divots in $\{P\}$ are (in mm):

$$\mathbf{p}_{ND} = \begin{bmatrix} 30 & 15 & 30 & 15 & -15 & -30 & -15 & -30 \\ 15 & 30 & -15 & -30 & 30 & 15 & -30 & -15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \quad (8.4)$$

A group of 6 features were selected to be tracked. The nominal homogeneous coordinates of the chosen features in the plate's DRF {P} are (in mm):

$$\mathbf{p}_{NF} = \begin{bmatrix} 10 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 5 & -5 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \quad (8.5)$$

8.1.3 Methods

To assess the accuracy of the stereo vision system based on the accuracy of the reference system, the transformation between both tracking systems (${}^O\mathbf{T}_C$) and the transformation between the tracked elements (${}^D\mathbf{T}_F$) must be known.

8.1.3.1 Plate Calibration

Initially, a calibrated 4 Marker Digitizing Tool (Optotrak CERTUS accessory, NDI) recorded the position of each divot ($\mathbf{p}_D$) in the CERTUS's static reference frame (SRF) {O}, following the order depicted in Figure 8.6, from 1 to 8. The position and orientation of the markers' DRF in {O} was also recorded (${}^O\mathbf{T}_M$). The divots' positions in the markers' DRF ($\mathbf{p}_{D,M}$) were computed:

$$\mathbf{p}_{D,M} = ({}^O\mathbf{T}_M)^{-1} \cdot \mathbf{p}_D \quad (8.6)$$

Taking advantage of the plate shape and to minimise the divots' localisation error, $\mathbf{p}_{D,M}$ were projected to a plane fitted to them by least squares, and thus the projections of the divots on that plane ($\mathbf{p}_{D,P}$) were computed.

Knowing $\mathbf{p}_{D,P}$ and the markers' DRF (${}^O\mathbf{T}_M$) in the CERTUS SRF {O}, we calculated the divots' positions in {O}, $\mathbf{p}_{D,O}$, for each acquisition.

$$\mathbf{p}_{D,O} = {}^O\mathbf{T}_M \cdot \mathbf{p}_{D,P} \quad (8.7)$$

To simplify the notation, a reference frame was assigned to $P_{D,O}$ to be coincident to plate DRF {P} (see Figure 8.4 and 8.6). The divots' DRF {D} x-axis is denoted as $\mathbf{d}_{xx}$, y-axis as $\mathbf{d}_{yy}$ and z-axis as $\mathbf{d}_{zz}$, and are unit vectors ($\hat{\mathbf{d}}$):

$$\mathbf{d}_{xx} = \left(\sum_{i=1,2,3,4}^i \mathbf{p}_{D,O}(i) - \widehat{\sum_{j=5,6,7,8}^j \mathbf{p}_{D,O}(j)} \right) \quad (8.8)$$

$$\mathbf{d}'_{yy} = \left(\sum_{i=1,2,5,6}^i \mathbf{p}_{D,O}(i) - \widehat{\sum_{j=3,4,7,8}^j \mathbf{p}_{D,O}(j)} \right) \quad (8.9)$$

$$\mathbf{d}_{zz} = \mathbf{d}_{xx} \times \mathbf{d}'_{yy} \quad (8.10)$$

$$\mathbf{d}_{yy} = \mathbf{d}_{zz} \times \mathbf{d}_{xx} \quad (8.11)$$

The origin of the divots' DRF ($\mathbf{d}_0$) is computed:

$$\mathbf{d}_0 = \frac{1}{8} \sum_{i=0}^8 \mathbf{p}_{D,O}(i) \quad (8.12)$$

From a stereo camera acquisition, we got the position ($\mathbf{p}_F$) of the 6 selected features in the stereo camera SRF (see Figure 8.6). Likewise to the divots, a reference frame was assigned to $\mathbf{p}_F$ coincident to the plate DRF {P} (see Figure 8.4 and 8.6). Since {D} and {F} are assigned coincident to the {P}, the transformation between the tracked elements (${}^D\mathbf{T}_F$) is the identity matrix,

$${}^D\mathbf{T}_F = I_4 \quad (8.13)$$

Being the DRF {F} x-axis denoted as $\mathbf{f}_{xx}$, y-axis as $\mathbf{f}_{yy}$ and z-axis as $\mathbf{f}_{zz}$, and being unit vectors ($\hat{\mathbf{f}}$):

$$\mathbf{f}_{xx} = \left(\frac{1}{2} (\mathbf{p}_F(1) + \widehat{\mathbf{p}_F(2)}) - \mathbf{p}_F(6) \right) \quad (8.14)$$

$$\mathbf{f}'_{yy} = \left(\frac{1}{2} (\mathbf{p}_F(3) + \widehat{\mathbf{p}_F(4)}) - \mathbf{p}_F(5) \right) \quad (8.15)$$

$$\mathbf{f}_{zz} = \mathbf{f}_{xx} \times \mathbf{f}'_{yy} \quad (8.16)$$

$$\mathbf{f}_{yy} = \mathbf{f}_{zz} \times \mathbf{f}_{xx} \quad (8.17)$$

The origin of the features' DRF ($\mathbf{f}_0$) is computed:

$$\mathbf{f}_0 = \frac{1}{4} \sum_{i=2,4,5,6}^i \mathbf{p}_F(i) \quad (8.18)$$

With the divots' {D} and features' DRF {F}, at each synchronous acquisition from the stereo camera and CERTUS, we know the transformation from {D} to the CERTUS SRF {O} - ${}^O\mathbf{T}_D$ - and the transformation from {F} to the stereo camera SRF {C} - ${}^C\mathbf{T}_F$ (see Figure 8.4).

8.1.3.2 Registration

To close the loop and compare both tracking systems, the transformation from the stereo camera SRF to the CERTUS SRF (${}^O\mathbf{T}_C$) was computed. The stereo camera and CERTUS remained immobile during the whole test.

The plate was placed at 10 different positions and orientations, exploring the stereo camera depth of field and field of view - keeping the features visible by the stereo camera and the markers visible by the CERTUS. It means a plate motion range volume of $0.05 \times 0.05 \times 0.1 \text{ m}^3$ and a rotational motion within 60° around the x and y-axis, and 360° around the z-axis of the stereo camera's SRF {C}. The plate motion range will henceforth be referred to as, plate workspace.

For each plate position, a single snapshot acquisition was recorded by both systems. In the end, we had 10 DRFs of the divots in the CERTUS's SRF (${}^O\mathbf{T}_{D,i}$) and the 10 corresponding DRFs of the features in the stereo camera's SRF (${}^C\mathbf{T}_{F,i}$). As seen in Figure 8.4, and given (8.13),

$${}^O\mathbf{T}_{D,i} = {}^O\mathbf{T}_C {}^C\mathbf{T}_{F,i} \quad i = 1, \dots, 10 \quad (8.19)$$

An adaptation to the algorithm presented in De Momi et al. (2008); Comparetti et al. (2012b) was applied to compute the ${}^O\mathbf{T}_C$ transformation, which is expressed as 6 independent parameters: 3 for translation (x , y and z) and 3 for rotation. The iterative algorithm is used to minimise the cost function F with respect to the 6 translation and rotation unknowns, with Levenberg-Marquardt optimisation algorithm:

$$F = \| {}^O\mathbf{T}_{D,i}^{-1} \cdot {}^O\mathbf{T}_C {}^C\mathbf{T}_{F,i} \| \quad i = 1, \dots, 10 \quad (8.20)$$

In each iteration step, the translational and rotational parameters are computed against the translation and rotation errors, which are scaled to the same order of magnitude.

8.1.3.3 Acquisitions

A static and a dynamic analysis were performed. For both static and dynamic analysis, 4 sets of 10 acquisitions were recorded. In each static/dynamic set of acquisitions, the plate was moved in different depths relative to the stereo camera (see Figure 8.7), within its workspace.

In a different study, we performed acquisitions on 12 drug resistant epileptic patients and measured the craniotomy areas, and obtained the distribution quartiles: $Q_{25} = 2107.62 \text{ mm}^2$, $Q_{50} = 2332.75 \text{ mm}^2$, $Q_{75} = 2552.12 \text{ mm}^2$. Generally, the craniotomy was ellipsoid in shape, but approximating to a circumference it leads to a craniotomy diameter between 50 and 60 mm. The SM25WI 2/3" lenses have an intrinsic angle field of view of $20.1^\circ \times 15.1^\circ$. At a distance of 40cm one camera has a field of view of $141.78 \times 106.03 \text{ mm}$ and at 50 cm the camera field of view is $177.23 \times 132.54 \text{ mm}$. Given that both cameras field of view is as overlapped as possible, the distance between the stereocamera and the surgical field was chosen in order for the field of view to cover a typical craniotomy.

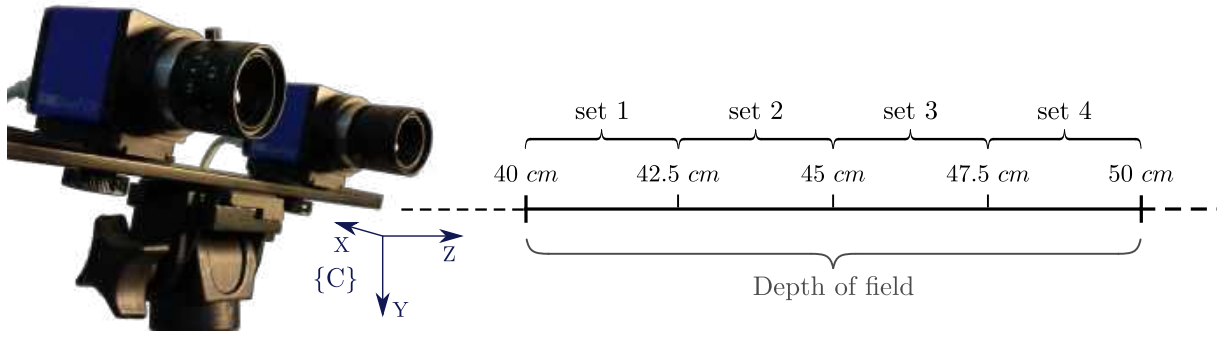


Figure 8.7: Distance from the stereo camera at which the plate was tracked in each test set.

Static Acquisitions Likewise to the registration process (c.f. section 8.1.3.2), the plate was placed at 10 different positions and orientations within the plate workspace. For each acquisition, one snapshot of the plate is taken from the stereo camera (of features' DRF) and from CERTUS (of divots' DRF), respectively ${}^C\mathbf{T}_F$ and ${}^O\mathbf{T}_D$.

Dynamic Acquisitions For each acquisition the plate is manually moved within its workspace and is tracked by both stereo camera and CERTUS for 5 seconds at 10 fps. Due to the velocity requirements to track the cortical surface, we limited the linear and angular speed of the plate during the test. The linear velocity norm (referred as linear speed) and the angle from the “axis-angle representation” of the plate motion throughout all the acquisitions, have mean values of 5 mm/s and $1.471^\circ/\text{s}$ and a maximum value of 25 mm/s and $9.699^\circ/\text{s}$, respectively.

Table 8.1: Test Performed.

	Static Analysis	Dynamic Analysis
Number of sets	4	4
Acquisitions per set	10	10
Frames per acquisition	1	50

8.1.3.4 Data analysis

For each static and dynamic acquisition and according to the registration from the stereo camera SRF to the CERTUS SRF, an error matrix $\mathbf{E}$ was computed:

$$\mathbf{E} = \| {}^O\mathbf{T}_{D,a}^{-1} \cdot {}^O\mathbf{T}_C {}^C\mathbf{T}_{F,a} \| \quad a = \text{each acquisition} \quad (8.21)$$

In both Static and Dynamic analysis the translation error $\mathbf{E}_T$ is computed as the norm of the translation component of the error matrix $\mathbf{E}$:

$$\mathbf{E}_T = \sqrt{x^2 + y^2 + z^2} \quad (8.22)$$

To represent the rotation error ($\mathbf{E}_R$), the angle from angle-axis representation is used.

Static Analysis A non-parametric test (Kruskal-Wallis $p < 0.05$ and Bonferroni correction) was performed to test whether the stereo camera translation and rotation errors - with an immobile target - are significantly different between each test set (different regions of the depth of field).

Dynamic Analysis In the dynamic analysis, the tracking error was analysed as a function of the linear and angular speed of the plate (v and ω). To the translation and rotation errors in each frame, it is associated the linear and angular speed of the plate, based on the CERTUS acquisitions.

Each “error-speed” measurement from each frame was grouped according to the linear and angular speed of the plate in 3 populations. The speed limits that define each population were assigned in order to keep the same number of samples in each population. Thus, the error-speed measurements for linear speed are grouped in: a) low linear speed ($v < 3$), b) medium linear speed ($3 \leq v \leq 6$) and high linear speed ($v > 6$) mm/s. While the error-speed measurements for angular speed are grouped in: a) low angular speed ($\omega < 0.5$), b) medium angular speed ($0.5 \leq v \leq 1.5$) and high angular speed ($v > 1.5$)°/s.

A non-parametric test (Kruskal-Wallis $p < 0.05$ and Bonferroni correction) was performed to test for the null hypothesis that the error is significantly different for each population of low, medium and high linear and angular speed, in all depth of field.

8.1.4 Results

The plane fitting error is the Euclidean distance from the detected position to the plane projected position of the feature or divot. The divot plane fitting was only performed once (c.f. section 8.1.3.1) and the errors present a mean, standard deviation and maximum for the 8 divots of: 0.042, 0.030 and 0.089 mm. The feature plane fitting procedure was performed in each frame of each acquisition, and presented a mean, standard deviation and maximum error for the 6 features of: 0.053, 0.039 and 0.193 mm. Given the shape of the features in the plate, there is an uncertainty of detection of ± 0.25 mm.

The registration matrix ${}^O\mathbf{T}_C$ minimises the function (8.20). The residual errors of the iterative registration method are computed from the error matrices of all the calibration acquisitions. The descriptive statistic analysis of these errors across all the acquisitions are presented in Table 8.2.

Table 8.2: Registration residual errors.

		$\bar{x}$	σ	$\max(x)$
Translation (mm)	X	0.131	0.096	0.331
	Y	0.288	0.183	0.594
	Z	0.162	0.105	0.349
Rotation (deg)	α	0.497	0.293	1.078
	β	0.552	0.324	0.948
	γ	0.573	0.301	0.913

As shown in Figure 8.8, the stereo camera detection error of an immobile plate across all frames is inferior to 0.6 mm and 1.6° (Figure 8.8). The translation error between sets 2 and 4 are significantly

different (* means “populations significantly different”).

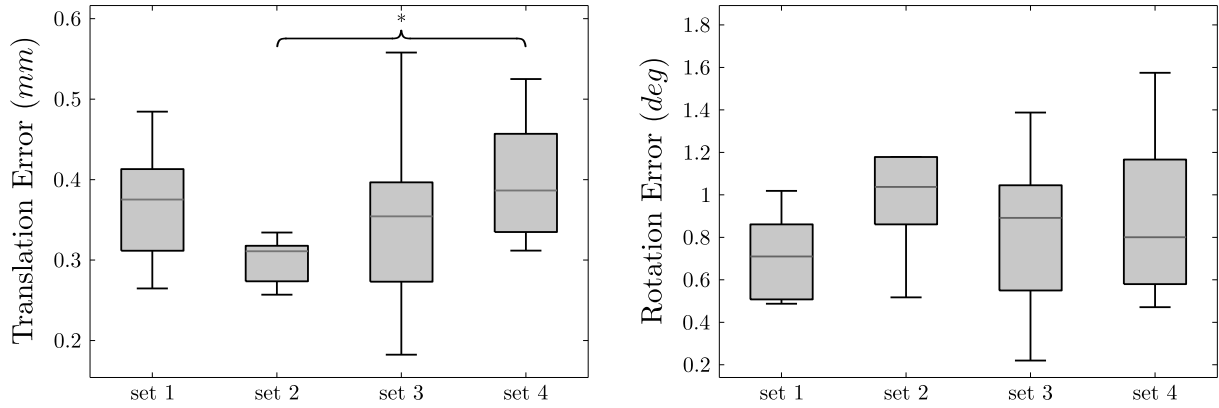


Figure 8.8: Static Analysis - Detection error (translation and rotation component) of an immobile object for each test set.

Besides the difference in sets 2 and 4, for the significance level of the non-parametric test performed, the null hypothesis was not rejected for either translation or rotation error, which means that there is no statistically significant impact on the error by detecting an object nearer or farther in the depth of field.

As depicted in Figure 8.9, the tracking error of the stereo camera for all the acquisitions is inferior to 2 mm and 2.5° . The translation error median value is higher as the plate linear speed increases. According to the non-parametric statistical test performed, there is a significant difference in the translation error when the plate is moving at a different linear speed.

The statistical test also rejects the null hypothesis that the rotation error for different angular speeds fits the same distribution. Although, there is no statistically significant difference in the distribution of the rotation error between low and medium angular speeds, higher angular speed produces significantly greater rotation errors.

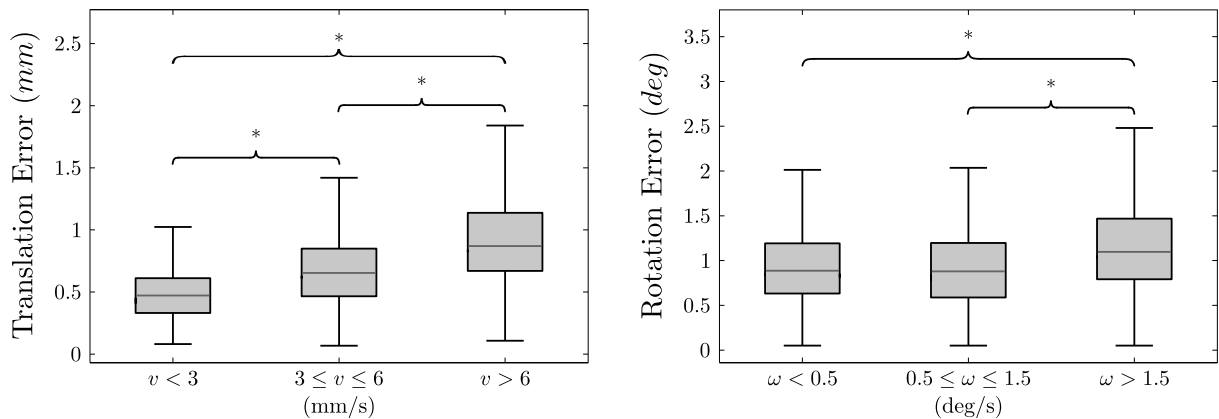


Figure 8.9: Dynamic Analysis - Tracking error of a moving object across all test sets. Low, medium and high linear and angular speeds depicted in left and right figures. Statistically significant difference of median translation error among all angular speeds. Statistically significant difference of median rotation error among all angular speeds, except between low and medium angular speeds.

8.1.5 Brain Data Acquisition

For clinical data, 3 stereo image acquisitions were recorded from 53 year old male patient, with cryptogenic epilepsy, who underwent left antero-mesial temporal lobectomy. After the craniotomy, the stereo camera was positioned on a tripod at 40 cm from the skull opening and pitched 45° downward to avoid touching the sterile drapes. Cortical surface motion was recorded in 3 videos of 25, 33 and 58 seconds at 10 fps.

In each acquisition, features are selected from the brain surface salient visual points and their positions in the stereo camera's SRF are tracked. In the first acquisition 91 features were tracked, in the second video 99, and in the last video 57. To account for brain motion, the combined motion of the cloud of features was considered rather than the motion of singular features. Occluded features are dropped from the analysis.

Due to the limited hard drive access time, some frames were not recorded. Using the timestamp of each frame (c.f. section 8.1.1) the missing frames were identified and a cubic interpolation method was applied to reconstruct signal of the position of each feature along time. The displacement of the cloud of features centre of mass in each acquisition frame was analysed in the time and frequency domain, see Figure 8.10.

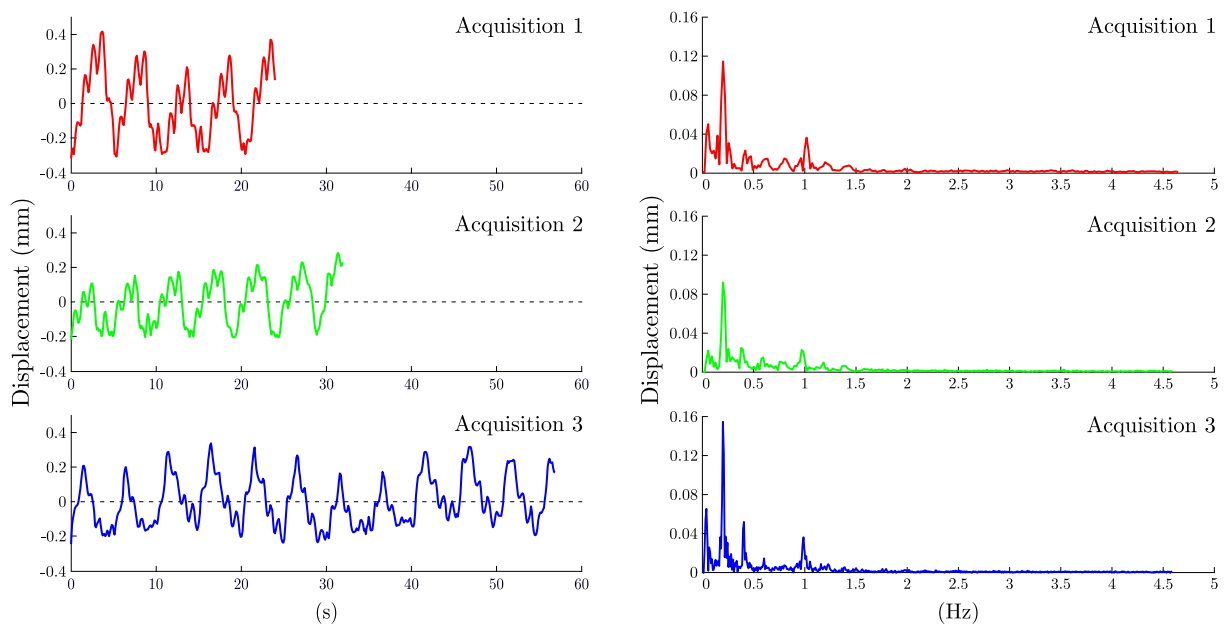


Figure 8.10: Time and Frequency analysis of brain motion in the 3 acquisitions, 25, 33 and 58 seconds. Left plots depict the displacement of the cloud of features' centre of mass throughout the acquisition. Continuous component of the signal was subtracted. Right plots represent the Fast fourier transform (FFT) analysis of the displacement of the cloud of features' center of mass.

The combined displacement of the cloud of features on the brain surface is inferior 0.8 mm in all three acquisitions. In the last acquisition frequency analysis (Figure 8.10 acquisition 3) it is clearly visible four peaks between 0 and 1 Hz. The first peak between 0.03 and 0.05 Hz can be attributed to pressure or temperature control (sympathovagal balance). The second peak has the biggest displacement amplitude with 0.2 Hz and is caused by breathing. After 0.2 Hz there are breathing harmonics with lower amplitude, which are found at integer multiples of the breathing frequency. A pronounced peak at 1 Hz is coherent

with the heart rate, and is caused by the blood pressure variations. Comparing to the results presented by Ji et al. (2013), both show respiration motion component (around 0.2 Hz) as the major cause for brain motion followed by the blood pressure variations component (around 1 Hz).

We also evaluated each features' linear velocity norm between acquisitions. The maximum registered speed was 1.119 mm/s with a mean and standard deviation of 0.184 and 0.137 mm/s, respectively. These features velocities fit the low linear speed population of the dynamic validation analysis.

8.1.6 Discussion

This section presents an approach to track cortical surface shift and pulsatile deformation in real time, based on stereoscopy. The stereo tracking algorithm detects and efficiently tracks up to 100 features selected from the cortical surface, and is robust against false stereo matched points, occlusion and outliers. Having a 10 fps acquisition rate, the stereo camera is able to measure not only the brain surface sag motion along time but also the brain pulsatile motion caused mainly by blood pressure variations and breathing. The ability to track the pulsatile motion is one of the breakthroughs in open skull-surgery brain tracking when compared to image based systems or laser-range scanned applications.

From the stereo vision systems for brain surface tracking able to measure the pulsatile motion presented in literature (Sun et al., 2005; Ji et al., 2011), only Sun et al. (2005), provided some details towards the system accuracy and validation. Their system has a point-localisation average error of 0.87 mm and an average error of capturing the cortical surface shape of 1.2 mm.

A method for objective validation of a stereo vision system towards open-skull neurosurgery was also presented. The phantom-based study performs a static and dynamic evaluation of the detection and tracking capabilities of the stereo vision system, respectively, against a gold-standard optical tracking system of reference, the Optotrak CERTUS. The validation setup was thorough explained since the synchronisation of both vision systems to the design of the phantom and relation between tracked elements.

According to the non-parametric statistic test performed, the translation and rotation error distribution was not significantly affected by the position of the plate within the stereo camera's depth of field. The dynamical analysis shows a statistically significant increase of the translation error as the linear speed of the plate increases. Only when comparing low or medium to high angular speed, the rotation errors show a significant increase.

Within the stereo camera's depth of field, which means tracking an object at a distance of 40 to 50 cm, the system proved to detect the plate position and orientation with an error inferior to 0.5 mm and 1.6° , which is inferior to the reported point-localisation accuracy of Sun et al. (2005). Moreover, considering the maximum feature speed of 1.119 mm/s, which is sensible compared to the maximum brain motion values reported by Greitz et al. (1992) of 1.5 mm/s, both fit low linear speed population of the dynamic validation analysis, which presents a median linear error of 0.5 mm and the maximum 1.03 mm.

The stereo vision system was tested in the operating room. Cameras were placed at a safe distance from the patient, and 3 acquisition videos were acquired during 25, 33 and 58 seconds at 10 fps. The cortical pulsatile motion amplitude quantified after the centre of mass of the features was inferior to 0.8 mm.

Breathing is the largest contribute to cortical brain motion amplitude with frequency of 0.2 Hz.

Due to the short duration of the video recordings it was not possible to quantify the brain shift. An integration of the stereo vision system with the surgical equipment would enable the quantification of brain shift and pulsatility, which could potentially be applied into surgical field stabilisation and robotic motion compensation.

Tracking brain shift deformation, before the surgeons perform brain resection, could benefit the intra-operative functional mapping phase, in which the surgeon stimulates brain cortex. Although not reliable as 3D imaging techniques like CT and MRI to measure the deformation of deep brain structures (Ferrant et al., 2001), cortical surface information can be used by biomechanical models to update the brain model displayed on the navigation system. As further development, we plan to incorporate a real-time volumetric deformation model that extends the estimation of the deformation from the cortical surface to brain deep structures (Rasin et al., 2014). It will allow an easier correlation between the functional and the anatomical mapping. Nonetheless, once the tissue is removed, the features can not be tracked anymore.

Another limitation is due to the possible interference of the cameras to the surgeon movements on the operating field, this could be solved using a microscope, with 3D cameras option. Since this can be displaced during the surgical procedure, it should be continuously tracked with an external localisation system.

8.2 Haptic Feedback Tele-operation

A tele-operation working mode was developed on top of the proposed control architecture. It permits the user to maneuver the robot's pose in real-time and with haptic feedback. The user may impose virtual constraints to limit the robot's movement, integrate motion filtering or apply scaling factors for precise movement increments.

The tele-operation module was developed with the Omega 7 (Force Dimension, Nyon, Switzerland), a device that provides an active force feedback, rotation sensing and active grasping. With 7 controllable DoF, of which 4 (X-, Y-, Z-, and grasp-axis) are actuated and responsible for providing the haptic feedback.

As a module to integrate the proposed control architecture, it is implemented as components that interface with the Task Supervisor component, see Figure 8.1. Two components are described, the Force Dimension Omega 7 Driver - a component that synchronously communicates with the Omega 7 device and provides the necessary information and services to the architecture, and the Haptic Control Modes - a component that implements the task specific actuation modes and respective motion scaling or virtual constraints.

However simple, the current tele-operation approach takes advantage of the developed kinematic inversion algorithm to follow the fed in pose while avoiding singularities or joint limits, and remaining in the defined configuration space (refer to subsection 6.2). It trumps over the most common tele-operation schemes of intrinsically redundant manipulators that lock the arm-angle to avoid dealing with the redundant degree of freedom, or tele-operation schemes that cannot properly handle joint limits or singularities when

following a task-space pose¹.

8.2.1 Force Dimension Omega 7 Driver

The Omega 7 Driver component serves as a wrapper to the device driver (Figure 8.13). The device functionality is partitioned in two libraries, which reflect the two principal operation modes: the DHD - for haptic functions, and the DRD - for robotic functions.

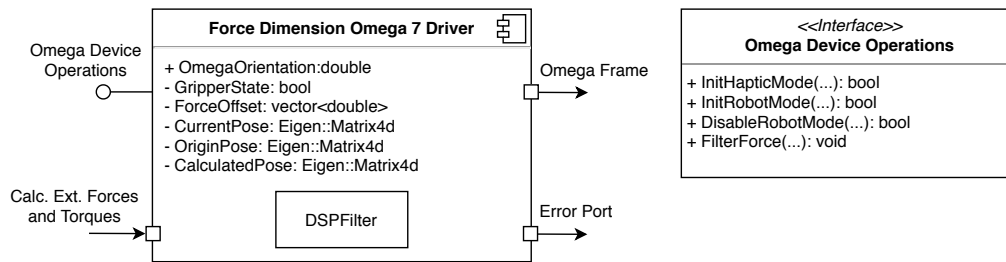


Figure 8.11: Force Dimension Omega Driver component diagram.

The DHD mode API provides access to the device position and orientation, force and torque, gripper position and force, to a gravity compensation mode, to configure the button interface, to set the initial device orientation, etc. It can summarily be described as the sensing mode to perceive the operator input (position/force) and to reproduce the force feedback related to the haptic feature.

The DRD is the actuation mode. It features functions to actively control the device motion or trajectories, provides access to the motion ratios, motion filtering, lock or hold the gripper position, etc.

The same application can call on functions from the DHD and the DRD library, which is the case of the Omega 7 Driver component. By default, the component is initialized and operates under the DHD mode. Whenever the application requires the device to reposition the gripper or have it hold its position, the DRD mode is activated. The device driver is embedded as a library of the described component, which is also responsible to guarantee the communication frequency, set at 100 Hz by the component's ExecutionEngine. The Omega 7 Driver component has an Interface that provides methods to interact with the device control modes, and two ports: i) one input port to receive the external force to be reproduced by the haptic device, and ii) one output port to forward the relative target pose - to be followed by the robot's end-effector.

During the component's configuration step, the initial device orientation is set OmegaOrientation. For simplification, it is assumed that both the Omega 7 device as well as the robot's base reference frames have an aligned z-axis, and the angle between the x-axis is defined by the OmegaOrientation variable, see Figure 8.12.

The component's interface contains one input port - the Calc. Ext. Forces and Torques, where it receives the haptic feedback forces to be executed by the device. Its interface also includes two output ports - the Omega Frame and the Error port. The Error port functions similarly to other components, forwards error messages to the TaskSupervisor component. The OmegaFrame forwards the position and

¹Videos showing the described functionality can be found at <https://youtu.be/ZdrsqeODmnc> and at <https://youtu.be/93gKdHqbgoy>

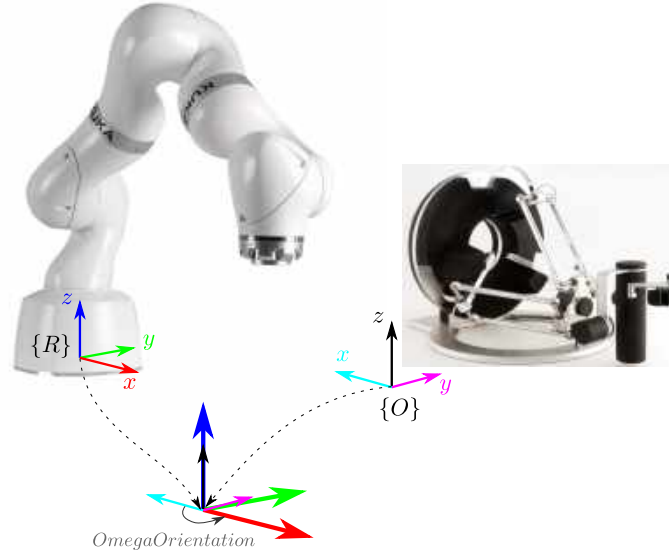


Figure 8.12: Representation of the OmegaOrientation variable.

orientation of the gripper (refer to CalculatedPose in subsection 8.2.1.1) packed into a KDL::Frame to the HapticControlModes component.

8.2.1.1 Gripper Operation

The Omega 7 gripper was configured to as a push-to-move input. Instead of continuously forwarding the gripper poses to the robot controller, the push-to-move mechanism provides the operator with the full control over when and how the manipulator moves. Moreover, the operator can take full advantage of the device workspace to reposition the gripper before engaging on a new manipulator motion.

This routine is implemented using three transformation matrices: i) CurrentPose, the pose of the omega 7 gripper relative to its base (${}^O\mathbf{T}_G$), ii) OriginPose, the gripper pose stored when the gripper close event occurred (${}^O\mathbf{T}_{G_0}$), and iii) CalculatedPose, the current gripper pose relative to the gripper close stored pose (${}^{G_0}\mathbf{T}_G$).

At the beginning of each update cycle, the component executes four calls to the omega 7 device driver API to obtain the device status², gripper frame position and orientation, and gripper opening position. The GripperState is set by the gripper opening position, below a threshold position the gripper is considered closed. Hysteresis was added to the input to reduce the debounce effect.

The gripper frame position and orientation are always returned relative to the Omega 7 reference frame (${}^O\mathbf{T}_G$). Whenever the GripperState switches to “closed” the current gripper position and orientation are stored as the transformation, ${}^O\mathbf{T}_{G_0}$. Then, the target relative frame to be followed by the robot’s end-effector is determined as the gripper position and orientation relative the position and orientation at the moment of the gripper close event, ${}^{G_0}\mathbf{T}_G$,

$${}^{G_0}\mathbf{T}_G = ({}^O\mathbf{T}_{G_0})^{-1} {}^O\mathbf{T}_G. \quad (8.23)$$

²For more information consult the Force Dimension SDK documentation.

8.2.1.2 Motion and Haptic Filter

To reduce the effect of physiological tremor induced by oscillatory reflex loops (Lippold, 1970; Lakie et al., 1986) during tele-operation, we implemented a digital filter to process the relative position and orientation inputs from the Omega 7 device. The filter is a 5th order Butterworth low-pass with a cutoff frequency of 5 Hz to attenuate -30 dB at around 10 Hz - the frequency of physiological tremor.

On a similar note, the force reproduced by the Omega 7 as a result of the external forces applied to the robot end-effector is also filtered. Regardless of the precision of the torque sensors distributed along the manipulator's structure, the dynamic model errors, friction, as well as other small unaccounted forces will generate small errors in the external torques calculated at the robot controller. The external torques which would ideally remain null if no external forces were applied to the robot structure, often present a residual error. To cope with this situation, whenever the tele-operation control modes switch (see the following subsection 8.2.2), a vector buffer is filled with the current external forces and torques during a 3 second interval (at 100 Hz - 300 samples). The mean of external forces and torques is calculated and labeled as the offset value to be subtracted from the following measurements.

8.2.2 Haptic Control Modes

The Haptic Control Modes component interfaces the Force Dimension Omega 7 Driver component with the Task Supervisor / high-level control architecture (Figure 8.13). It is responsible for contextualizing the information sent and received from the Omega device in respect to the robotic task.

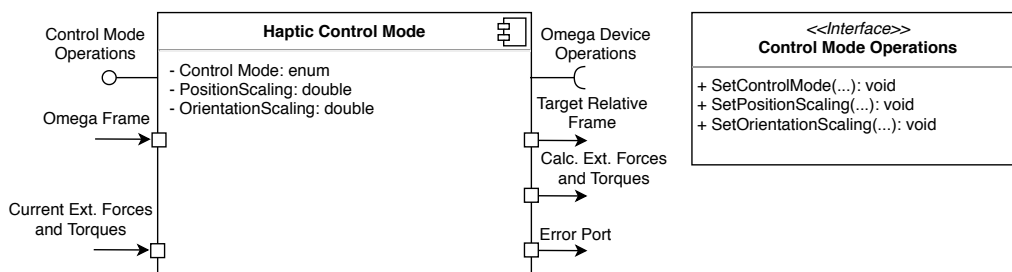


Figure 8.13: Haptic Control Modes component diagram.

The component requires the Omega Device Operations interface to adjust the haptic operation mode (DHD or DRD) and notify about control mode switches, such that the external torque baseline can be filtered out. It provides the Control Mode Operations interface to set the control mode and tune the position and orientation scaling.

To illustrate the functionality of the tele-operation system, three control modes were formulated and implemented:

1. free motion, motion permitted in all task space coordinates;
2. point lock, orientation motion permitted with the position restricted to the TCP;
3. pierce, motion permitted along and around the z-axis of the end-effector.

Each mode can code different position and orientation scaling or force sensitivity, which can also be adjusted in runtime. Linear position and orientation scaling is applied to the received relative frames. The scaling of position displacement vectors is straightforward. The orientation scaling is based on the SLERP algorithm (Shoemake, 1985). Both variables can be defined in the interval $]0, 1]$, where 1 represents no scaling.

Depending on the selected mode motion restrictions, the Omega Frame inputs received are filtered such that only the relevant motion vector components are forwarded to the Task Supervisor as the Target Relative Frame. Conversely, the perceived Current Ext. Forces and Torques can be processed to represent tactile feedback along specific axis, which are then sent to the Omega driver as the Calc. Ext. Forces and Torques.

The free motion mode is designed to permit the operator to execute ampler motions and quickly position the robot near an adequate pose - previous to initiating the procedure. The Omega device is set to the haptic operation mode (DHD) and by default, the motion scaling is set to 1:1 or similar correspondence. Forces perceived at the end-effector are sent as an output through the Calc. Ext. Forces and Torques port to be reproduce by the tele-operation device.

The point lock mode permits adjusting the end-effector orientation around a fixed point that is defined in respect to the end-effector. One logical application of this mode would be to define the orientation of a probe or electrode around an access port/point before advancing to the target. The orientation scaling factor is reduced to permit finer angulation adjustments. The device is set to the robot operation mode (DRD) in order to force the gripper position to remain stationary while the orientation is adjusted. Since no torque feedback can be reproduced by the Omega 7 device, only the forces felt at the end-effector are sent as feedback in this mode.

Finally, the pierce control mode permits moving the robotic manipulator along the end-effector main/actuation axis and restricting its movement in other axes. The Omega device is set to the haptic operation mode (DHD) to reproduce the force felt by the end-effector along its actuation axis. Similar to the point lock control mode, the end-effector movement is scaled down to permit a finer and more controlled motion.

Chapter 9

Conclusions and Future Work

In this thesis, we studied the practice and technologies involved with Stereotactic and Functional Neurosurgery, a subset of traditional Neurosurgery. It was identified the main imaging and tool guidance techniques and apparatus that were historically and are currently applied in neurosurgery operating rooms. We listed what we considered the most relevant stereotactic procedures for diagnosis and treatment of neurological disorders, described the workflow and core principles for each procedure variation from the pre- to the post-operative stage.

Special emphasis was put into the revision of robotic technology for neurosurgical purposes, which stands as an alternative to traditional stereotactic apparatus - the frame. Several studies point out the advantages of robots as consistent, repeatable, accurate and overall safer agents to assist in neurosurgeries. After consulting with neurosurgeons, we concluded that the capacity to replace the repetitive process of setting up the tool guidance apparatus, simple and intuitive interfaces, cost-effectiveness and portability stand out as the most sought after features for a robotic system. A comprehensive list of the key robotic systems for brain neurosurgery is compiled with detailed feature descriptions and comparison. Dexterous, compact, serial manipulators are features consistently found among the most successful systems. For medical institutions, the high-acquisition costs with no immediate investment return stand as the principal deterrent to the engagement with this technology. For research centers involved with the development of new technologies, the highest barrier consists of the certification and regulatory approvals, whose complexity reflects the critical nature of the safety requisites (Faria et al., 2015a).

Two alternative robotic system solutions were devised during this project, a modular and a monolithic control system. The monolithic control system was designed from the adaptation of an industrial manipulator to perform as a direct replacement of the stereotactic frame. The developed system shares the repeatable, precise and flexible aspects of a serial robotic manipulator to withdraw the need for the repetitive setup and adjustment processes of the traditional stereotactic frame. Although the system is not easily certifiable for clinical practice, it offers an easily integrable, expandable and inexpensive solution as proof-of-concept and research instrument (Faria et al., 2016).

A high-level control architecture was proposed as a modular system based on Component-Based Software Engineering principles. While monolithic solutions permit a faster technological output with a dedicated solution based on a specific set of hardware and software, modular systems compartmentalize

the problem into a set of blocks that can be recycled, upgraded or substituted without the need for a system overall revamp. A study of the state-of-the-art on modular software platforms for robotic applications was conducted and we decided to implement our solution with the OROCOS framework based on the hard-real time support, deterministic behavior and a well-designed component and communication model. The control architecture was tested with a KUKA LBR iiwa 7 R800 an anthropomorphic 7 DoF robotic manipulator, certified for human-robot collaboration and which shares a similar structure and interface with the LBR Med, the first standard production robot model certified under the medical electrical equipment and medical device software IEC norms.

To facilitate the control system implementation, we designed and developed a virtual scenario and a virtual robot control strategy to mimic the real system in a robotics simulation software - V-REP. Two alternative and yet similar components bridge all the control system with either the real platform or the virtual platform API seamlessly. A minimal communication model between the hardware/virtual hardware and the control architecture was devised, and most of the control was delegated to the modular architecture. Individual components with standardized interfaces were implemented to operate as task space and joint space controllers, kinematic solvers, application data management, as well as hardware wrappers. We developed a novel analytic inverse kinematics method for redundant manipulators capable of avoiding joint limits, singularities and uniquely solving the nullspace. It stands as the cornerstone of the architecture, permitting a deterministic and predictable robot motion, an intuitive nullspace resolution strategy that guarantees feasible trajectories and complete control over the manipulator's posture from the task space (Faria et al., 2018).

One of the crucial aspects of any surgical robot is their accuracy. Knowing that the modular architecture, responsible for operating the robotic manipulator, operates externally to the robot controller unit, it requires an accurate kinematic model of the serial arm to reproduce precise movements. Most robots are factory calibrated, but seldom are the intrinsic kinematic models available to the robot operator. A novel algorithm was developed to determine the standard DH parameters of serial robotic manipulators with variable kinematic structures and joint types/dispositions. The method relies on controlled robot motions and Dual Vector Algebra to determine the relative frames between each robot joint and consecutively the DH parameters (Faria et al., 2019).

On top of the kinematic calibration, the system's application accuracy was tested in vitro. A low-cost, reusable acrylic phantom was designed and built to contain several point-shaped targets distributed in a volume similar to that of a human head and was filled with an agar-agar solution mimicking the brain tissue. The phantom was scanned in a CT machine and the targets and probe entry points selected. After registering the robotic system to the phantom employing a touch-based calibration procedure, a stereotactic procedure was conducted to place small plastic beads at the location of the phantom targets. Out of the 11 targets, the system achieved an average application accuracy of 0.56724 mm with a maximum error of 1.0944 mm, a result comparable with the in vitro test results of similar systems.

Two additional modules for the proposed modular architecture are also presented. One of the modules implements a method to quantify and track cortical surface displacement in open-skull neurosurgeries. The system is based on a stereo-camera apparatus and is robust against false stereo matched points, occlusion,

and outliers. Operating at 10 Hz, the system of stereo-cameras is capable of measuring the brain surface sag motion as well as the pulsatile motions caused by blood pressure variations and breathing. The system was validated against a high-precision commercial motion tracker presenting a maximum position and orientation error of 0.5 mm and 1.6° in a static regimen, and a maximum error of 1 mm and 2° in a dynamic regimen. It proved to be more precise than comparable systems from literature (Faria et al., 2014).

The second extension module endows the system with a teleoperation functionality. One of the powerful perks of modular architecture is the ability to easily add features to the system without disrupting its structure. A hardware wrapper was created to bridge a haptic feedback 7 DoF joystick with the control architecture and an additional component added to process the received position and orientation input - motion scaling and filtering. Three teleoperation modes were implemented with distinct haptic feedback sensing for laparoscopic surgeries. Despite the module initial development stage, it denotes the potential to be extended and applied to different applications.

The proposed architecture sets up the foundation to facilitate the development of robotic systems oriented to surgical applications. Focused on the motion control aspects, the surgical robotized system is not completed without state-of-the-art registration and imaging planning systems. These sub-systems can be seamlessly integrated into the control architecture and are considered future work. The next logical step to certificate the high-level control architecture would be to test the application accuracy in animal and corpse models.

Bibliography

ABB (2019). ABB IRB 140 Datasheet.

Abelson, J., Curtis, G., Sagher, O., Albucher, R., Harrigan, M., Taylor, S., Martis, B., and Giordani, B. (2005). Deep brain stimulation for refractory obsessive-compulsive disorder. *Biological Psychiatry*, 57(5):510–516.

Active Project (2012). ACTIVE.

Amin, D. and Lunsford, L. (2004). Volumetric resection using the SurgiScope: a quantitative accuracy analysis of robot-assisted resection. *Stereotactic and Functional Neurosurgery*, 82(5-6):250–3.

An, H. H., Clement, W. I., and Reed, B. (2014). Analytical inverse kinematic solution with self-motion constraint for the 7-DOF restore robot arm. In *International Conference on Advanced Intelligent Mechatronics*, pages 1325–1330, Besançon. IEEE.

Anderson, M., Postupna, N., and Ruffo, M. (2003). Effects of high-frequency stimulation in the internal globus pallidus on the activity of thalamic neurons in the awake monkey. *Journal of Neurophysiology*, 89(2):1150–60.

Andrews, D., Scott, C., Sperduto, P., Flanders, A., Gaspar, L., Schell, M., Werner-Wasik, M., Demas, W., Ryu, J., Bahary, J., Souhami, L., Rotman, M., Mehta, M., and Curran, W. (2004). Whole brain radiation therapy with or without stereotactic radiosurgery boost for patients with one to three brain metastases: phase III results of the RTOG 9508 randomised trial. *The Lancet*, 363(9422):1665–1672.

Asfour, T. and Dillmann, R. (2003). Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 1407–1412, Las Vegas. IEEE.

Atcheson, B., Heide, F., and Heidrich, W. (2010). Caltag: High precision fiducial markers for camera calibration. In *Vision Modeling and Visualization - VMV*, pages 41–48. Eurographics Association.

Auer, L., Starkie, S., Auer, D., and Davies, B. (2002). Simulating minimally invasive neurosurgical interventions using an active manipulator. *International Congress Series*, 1247:199–210.

- Batista, P. D., Machado, I. P., Roios, P., Lavrador, J., Cattoni, M. B., Martins, J., and Carvalho, H. (2019). Position and orientation errors in a neuronavigation procedure: a step-wise protocol using a cranial phantom. *World Neurosurgery*, pages 1–9.
- Bekelis, K., Radwan, T., Desai, A., and Roberts, D. (2012). Frameless robotically targeted stereotactic brain biopsy: feasibility, diagnostic yield, and safety. *Journal of Neurosurgery*, 116(5):1002–6.
- Benabid, A. (2003). Deep brain stimulation for Parkinson's disease. *Current Opinion in Neurobiology*, 13(6):696–706.
- Benabid, A., Chabardes, S., Seigneuret, E., Hoffmann, D., and LeBas, J. (2009). The Tailarach Stereotactic System. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 32, pages 495–509. Springer Verlag Berlin Heidelberg, 2nd edition.
- Benabid, A., Cinquin, P., Lavalley, S., Le Bas, J., Demongeot, J., and de Rougemont, J. (1987a). Computer-Driven Robot for Stereotactic Surgery Connected to CT Scan and Magnetic Resonance Imaging. In *Meeting of the American Society for Stereotactic and Functional Neurosurgery*, pages 153–154, Montréal. Karget AG.
- Benabid, A., Hoffmann, D., Seigneuret, E., and Chabardes, S. (2006). Robotics in neurosurgery: which tools for what? *Acta Neurochirurgica. Supplement*, 98:43–50.
- Benabid, A., Pollak, P., Gao, D., Hoffmann, D., Limousin, P., Gay, E., Payen, I., and Benazzouz, A. (1996). Chronic electrical stimulation of the ventralis intermedius nucleus of the thalamus as a treatment of movement disorders. *Journal of neurosurgery*, 84(2):203–14.
- Benabid, A., Pollak, P., Gervason, C., Hoffmann, D., Gao, D., Hommel, M., Perret, J., and de Rougemont, J. (1991). Long-term suppression of tremor by chronic stimulation of the ventral intermediate thalamic nucleus. *Lancet (London, England)*, 337(8738):403–6.
- Benabid, A., Pollak, P., Louveau, A., Henry, S., and de Rougemont, J. (1987b). Combined (thalamotomy and stimulation) stereotactic surgery of the VIM thalamic nucleus for bilateral Parkinson disease. *Applied neurophysiology*, 50(1-6):344–6.
- Bjartmarz, H. and Rehnrcrona, S. (2007). Comparison of accuracy and precision between frame-based and frameless stereotactic navigation for deep brain stimulation electrode implantation. *Stereotactic and Functional Neurosurgery*, 85(5):235–42.
- Black, P., Alexander, E., Martin, C., Moriarty, T., Nabavi, A., Wong, T., Schwartz, R., and Jolesz, F. (1999). Craniotomy for tumor treatment in an intraoperative magnetic resonance imaging unit. *Neurosurgery*, 45(3):423–31; discussion 431–3.
- Bonfè, M., Preda, N., Secchi, C., Ferraguti, F., Muradore, R., Repele, L., Lorenzi, G., and Fiorini, P. (2014). Distributed Control Architecture for Automated Surgical Task Execution with Coordinated Robot Arms. *IFAC Proceedings Volumes*, 47(3):10213–10218.

- Briot, S., Baradat, C., Guégan, S., and Arakelian, V. (2007). Contribution to the Mechanical Behavior Improvement of the Robotic Navigation Device Surgiscope. In 31st Mechanisms and Robotics Conference, Parts A and B, pages 653–661. ASME.
- Britt, R. and Rossi, G. (1982). Quantitative analysis of methods for reducing physiological brain pulsations. *Journal of Neuroscience Methods*, 6(3):219–29.
- Bronstein, J. M., Tagliati, M., Alterman, R. L., Lozano, A. M., Volkmann, J., Stefani, A., Horak, F. B., Okun, M. S., Foote, K. D., Krack, P., Pahwa, R., Henderson, J. M., Hariz, M., Bakay, R. A., Rezai, A., Marks, W. J., Moro, E., Vitek, J. L., Weaver, F. M., Gross, R. E., and DeLong, M. R. (2011). Deep brain stimulation for Parkinson disease: an expert consensus and review of key issues. *Archives of Neurology*, 68(2):165.
- Brouwer, M., Tunkel, A., McKhann, G., and van de Beek, D. (2014). Brain Abscess. *New England Journal of Medicine*, 371(5):447–456.
- Bruyninckx, H., Soetens, P., and Koninckx, B. (2003). The real-time motion control core of the Orocos project. In *IEEE International Conference on Robotics and Automation*, pages 2766–2771, Taipei. IEEE.
- Burdick, J. W. (1989). On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds. In *International Conference on Robotics and Automation*, pages 264–270, Scottsdale. IEEE Comput. Soc. Press.
- Buttrick, S. and Komotar, R. (2016). Introduction for Laser Interstitial Thermal Therapy (LITT) in Neurosurgery Supplement. *Neurosurgery*, 79:S1–S2.
- Bzostek, A., Kumar, R., Hata, N., Schorr, O., Kikinis, R., and Taylor, R. (2000). Distributed Modular Computer-Integrated Surgical Robotic Systems: Implementation Using Modular Software and Networked Systems. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 969–978, Pittsburgh.
- Camarillo, D., Krummel, T., and Salisbury, J. (2004). Robotic technology in surgery: past, present, and future. *American Journal of Surgery*, 188(4A Suppl):2S–15S.
- Cao, A., Thompson, R., Dumpuri, P., Dawant, B., Galloway, R., Ding, S., and Miga, M. (2008). Laser range scanning for image-guided neurosurgery: Investigation of image-to-physical space registrations. *Medical Physics*, 35(4):1593–1605.
- Capital, W. (2014). Mazor Robotics: Outstanding Company Few Investors Know Exists.
- Cardinale, F., Cossu, M., Castana, L., Casaceli, G., Schiariti, M., Miserocchi, A., Fuschillo, D., Moscato, A., Caborni, C., Arnulfo, G., and Lo Russo, G. (2013). Stereoelectroencephalography: surgical methodology, safety, and stereotactic application accuracy in 500 procedures. *Neurosurgery*, 72(3):353–66; discussion 366.

- Cardinale, F. and Mai, R. (2011). Robotic Implantation of Intracerebral Electrodes in Epilepsy Surgery. *Congress of Neurosurgical Surgeons Spring*, 12(2):24–26.
- Cardinale, F., Miserocchi, A., Moscato, A., Cossu, M., Castana, L., Schiariti, M., Gozzo, F., Pero, G., Quilici, L., Citterio, A., Minella, M., Torresin, A., and Lo Russo, G. (2012). Talairach methodology in the multimodal imaging and robotics era. In Scarabin, J.-M., editor, *Stereotaxy and Epilepsy Surgery*, chapter 10, pages 245–273. John Libbey Eurotext, 1st edition.
- Carpentier, A., Chauvet, D., Reina, V., Beccaria, K., Leclercq, D., McNichols, R., Gowda, A., Cornu, P., and Delattre, J. (2012). MR-guided laser-induced thermal therapy (LITT) for recurrent glioblastomas. *Lasers in Surgery and Medicine*, 44(5):361–368.
- Chan, F., Kassim, I., Lo, C., Ho, C., Low, D., Ang, B., and Ng, I. (2009). Image-guided robotic neurosurgery—an in vitro and in vivo point accuracy evaluation experimental study. *Surgical Neurology*, 71(6):640–647.
- Chan, T. F. and Dubey, R. V. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 11(2):286–292.
- Chang, P. (1987). A closed-form solution for inverse kinematics of robot manipulators with redundancy. *IEEE Journal on Robotics and Automation*, 3(5):393–403.
- Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410.
- Cohen-Adad, J., Paul, P., Morandi, X., and Jannin, P. (2006). Knowledge modeling in image-guided neurosurgery: application in understanding intraoperative brain shift. In Cleary, K. R. and Galloway, Jr., R. L., editors, *Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display*, San Diego.
- Colbaugh, R., Seraji, H., and Glass, K. L. (1989). Obstacle avoidance for redundant robots using configuration control. *Journal of Robotic Systems*, 6(6):721–744.
- Comeau, R., Sadikot, a. F., Fenster, A., and Peters, T. (2000). Intraoperative ultrasound for guidance and tissue shift correction in image-guided neurosurgery. *Medical Physics*, 27(4):787–800.
- Comparetti, M., De Momi, E., De Lorenzo, D., Beyl, T., Raczkowski, J., and Ferrigno, G. (2012a). Safe surgical robotic system and workflow design in the ACTIVE project for awake neurosurgery. In *IEEE International Conference on Intelligent Robots and Systems*, Vilamoura.
- Comparetti, M., De Momi, E., Vaccarella, A., Riechmann, M., and Ferrigno, G. (2011a). Optically tracked multi-robot system for keyhole neurosurgery. In *IEEE International Conference on Robotics and Automation*, pages 661–666. IEEE.

- Comparetti, M., Vaccarella, A., De Lorenzo, D., Ferrigno, G., and De Momi, E. (2011b). Multi-robotic approach for keyhole neurosurgery: the ROBOCAST project. In SCATh Joint Workshop on New Technologies for Computer/Robot Assisted Surgery, pages 1–4, Graz.
- Comparetti, M., Vaccarella, A., Dyagilev, I., Shoham, M., Ferrigno, G., and De Momi, E. (2012b). Accurate multi-robot targeting for keyhole neurosurgery based on external sensor monitoring. *Proceedings of the Institution of Mechanical Engineers. Part H, Journal of engineering in medicine*, 226(5):347–59.
- Cosman, E., Nashold, B., and Ovelman-Levitt, J. (1984). Theoretical Aspects of Radiofrequency Lesions in the Dorsal Root Entry Zone. *Neurosurgery*, 15(6):945–950.
- Craig, J. (2005). *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, 3rd edition.
- Curry, D., Gowda, A., McNichols, R., and Wilfong, A. (2012). MR-guided stereotactic laser ablation of epileptogenic foci in children. *Epilepsy & Behavior*, 24(4):408–414.
- Dahm, P. and Joubin, F. (1997). Closed form solution for the inverse kinematics of a redundant robot arm.
- Dandy, W. (1918). VENTRICULOGRAPHY FOLLOWING THE INJECTION OF AIR INTO THE CEREBRAL VENTRICLES. *Annals of Surgery*, 68(1):5–11.
- Daniele Comparetti, M., Beretta, E., Kunze, M., De Momi, E., Raczowsky, J., and Ferrigno, G. (2014). Event-based device-behavior switching in surgical human-robot interaction. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 1877–1882. IEEE.
- Davies, B. (2000). A review of robotics in surgery. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 214(1):129–140.
- Davies, B., Starkie, S., Harris, S. J., Agterhuis, E., Paul, V., and Auer, L. M. (2000). Neurobot: a special-purpose robot for neurosurgery. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4103–4108. IEEE.
- De Lorenzo, D., De Momi, E., Dyagilev, I., Manganelli, R., Formaglio, A., Prattichizzo, D., Shoham, M., and Ferrigno, G. (2011). Force feedback in a piezoelectric linear actuator for neurosurgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 7:268–275.
- De Momi, E., Caborni, C., Cardinale, F., Casaceli, G., Castana, L., Cossu, M., Mai, R., Gozzo, F., Francione, S., Tassi, L., Lo Russo, G., Antiga, L., and Ferrigno, G. (2014). Multi-trajectories automatic planner for StereoElectroEncephaloGraphy (SEEG).
- De Momi, E., Caborni, C., Cardinale, F., Castana, L., Casaceli, G., Cossu, M., Antiga, L., and Ferrigno, G. (2013). Automatic trajectory planner for StereoElectroEncephaloGraphy procedures: a retrospective study. *IEEE Transactions on Biomedical Engineering*, 60(4):986–93.

- De Momi, E., Cerveri, P., and Ferrigno, G. (2009). Robot and sensors integration for Computer Assisted Surgery and Therapy. In International Conference on Advanced Robotics, Munich, Germany. NearLab, Bioengineering Department, Politecnico di Milano.
- De Momi, E., Cerveri, P., Gambaretto, E., Marchente, M., Effretti, O., Barbariga, S., Gini, G., and Ferrigno, G. (2008). Robotic alignment of femoral cutting mask during total knee arthroplasty. *International Journal of Computer Assisted Radiology and Surgery*, 3(5):413–419.
- De Momi, E. and Ferrigno, G. (2010). Robotic and artificial intelligence for keyhole neurosurgery: the ROBOCAST project, a multi-modal autonomous path planner. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 224(5):715–727.
- Deacon, G., Harwood, A., Holdback, J., Maiwand, D., Pearce, M., Reid, I., Street, M., and Taylor, J. (2010). The Pathfinder image-guided surgical robot. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 224(5):691–713.
- DeMers, D. and Kreutz-Delgado, K. (1993a). Global Regularization of Inverse Kinematics for Redundant Manipulators. *Advances in Neural Information Processing Systems*, 5:255–262.
- DeMers, D. and Kreutz-Delgado, K. (1993b). Learning Global Topological Properties of Robot Kinematic Mappings for Neural Network-based Configuration Control. In *Neural Networks in Robotics*, pages 3–17. Springer US, Boston, MA.
- Denavit, J. and Hartenberg, R. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME E, Journal of Applied Mechanics*, 22:215–221.
- Devito, D., Kaplan, L., Dietl, R., Pfeiffer, M., Horne, D., Silberstein, B., Hardenbrook, M., Kiriyanthan, G., Barzilay, Y., Bruskin, A., Sackerer, D., Alexandrovsky, V., Stüer, C., Burger, R., Maeurer, J., Donald, G., Gordon, D., Schoenmayr, R., Friedlander, A., Knoller, N., Schmieder, K., Pechlivanis, I., Kim, I., Meyer, B., and Shoham, M. (2010). Clinical Acceptance and Accuracy Assessment of Spinal Implants Guided with SpineAssist Surgical Robot: Retrospective Study. *Spine*, 35(24):2109–15.
- Dlaka, D., Švaco, M., Chudy, D., Jerbić, B., Šekoranja, B., Šuligoj, F., Vidaković, J., Almahariq, F., and Romić, D. (2018). Brain biopsy performed with the RONNA G3 system: a case study on using a novel robotic navigation device for stereotactic neurosurgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 14(1):e1884.
- Dorfer, C., Stefanits, H., Pataraia, E., Wolfsberger, S., Feucht, M., Baumgartner, C., and Czech, T. (2014). Frameless Stereotactic Drilling for Placement of Depth Electrodes in Refractory Epilepsy. *Neurosurgery*, 10(4):582–591.
- Dostrovsky, J. and Lozano, A. (2002). Mechanisms of deep brain stimulation. *Movement Disorders*, 17 Suppl 3:S63–8.

- Doulgeris, J., Gonzalez-Blohm, S., Filis, A., Shea, T., Aghayev, K., and Vrionis, F. (2015). Robotics in Neurosurgery: Evolution, Current Challenges, and Compromises. *Cancer control: Journal of the Moffitt Cancer Center*, 22(3):352–9.
- Eggers, G., Mühling, J., and Marmulla, R. (2006). Image-to-patient registration techniques in head surgery. *International Journal of Oral and Maxillofacial Surgery*, 35(12):1081–95.
- Elder, J., Amar, A., and Apuzzo, M. (2009). Stereotactic and Image-Guided Biopsy. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 42, pages 645–663. Springer Verlag Berlin Heidelberg, first edition.
- Eljamel, M. (2007). Validation of the PathFinder neurosurgical robot using a phantom. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 3(4):372–7.
- Faria, C., Bicho, E., Rito, M., Louro, L., Monteiro, S., and Erlhagen, W. (2013). Robotic Assisted Deep Brain Stimulation Neurosurgery: First Steps on System Development. In *IASTED International Conference on Biomedical Engineering*, number BioMed, pages 331–338, Innsbruck, Austria. ACTAPRESS.
- Faria, C., Erlhagen, W., Rito, M., De Momi, E., Ferrigno, G., and Bicho, E. (2015a). Review of Robotic Technology for Stereotactic Neurosurgery. *IEEE Reviews in Biomedical Engineering*, 8:125–137.
- Faria, C., Ferreira, F., Erlhagen, W., Monteiro, S., and Bicho, E. (2018). Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. *Mechanism and Machine Theory*, 121:317–334.
- Faria, C., Sadowsky, O., Bicho, E., Ferrigno, G., Joskowicz, L., Shoham, M., Vivanti, R., and De Momi, E. (2014). Validation of a stereo camera system to quantify brain deformation due to breathing and pulsatility. *Medical Physics*, 41(11):113502 1–14.
- Faria, C., Vale, C., Machado, T., Erlhagen, W., Rito, M., Monteiro, S., and Bicho, E. (2015b). Experiential Learning of Robotics Fundamentals Based on a Case Study of Robot-Assisted Stereotactic Neurosurgery. *IEEE Transactions on Education*, 59(2):119 – 128.
- Faria, C., Vale, C., Rito, M., Erlhagen, W., and Bicho, E. (2016). A Simple Control Approach for Stereotactic Neurosurgery Using a Robotic Manipulator. In Garrido, P., Soares, F., and Moreira, A. P., editors, *12th Portuguese Conference on Automatic Control*, pages 397–408, Braga. Springer International Publishing.
- Faria, C., Vilaça, J., Monteiro, S., Erlhagen, W., and Bicho, E. (2019). Automatic Denavit-Hartenberg Parameter Identification for Serial Manipulators. In *IECON 2019 - Annual Conference of the IEEE Industrial Electronics Society*, pages 1–8, Lisbon. IEEE.
- Fei, B., Ng, W. S., Chauhan, S., and Kwok, C. K. (2001). The safety issues of medical robotics. *Reliability Engineering & System Safety*, 73(2):183–192.

- Ferrant, M., Nabavi, A., Macq, B., Jolesz, F., Kikinis, R., and Warfield, S. (2001). Registration of 3-D intraoperative MR images of the brain using a finite-element biomechanical model. *IEEE Transactions on Medical Imaging*, 20(12):1384–97.
- Fiani, B., Quadri, S., Farooqui, M., Cathel, A., Berman, B., Noel, J., and Siddiqi, J. (2018). Impact of robot-assisted spine surgery on health care quality and neurosurgical economics: A systemic review. *Neurosurgical review*.
- Field, M., Witham, T., Flickinger, J., Kondziolka, D., and Lunsford, L. (2001). Comprehensive assessment of hemorrhage risks and outcomes after stereotactic brain biopsy. *Journal of Neurosurgery*, 94(4):545–551.
- Finlay, P. and Morgan, P. (2003). PathFinder image guided robot for neurosurgery. *Industrial Robot: An International Journal*, 30(1):30–34.
- Fischer, I. (1998). *Dual-Number Methods in Kinematics, Statics and Dynamics*. CRC Press, 1st edition.
- Fitzpatrick, J. (2009). Fiducial registration error and target registration error are uncorrelated. In Miga, M. I. and Wong, K. H., editors, *SPIE*, pages 1–12.
- Galloway, R. (1998). Frameless stereotactic systems. In Gildenberg, P. and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 21, pages 177–182. McGraw-Hill Companies, 1st edition.
- Gerard, I. J. and Collins, D. L. (2015). An analysis of tracking error in image-guided neurosurgery. *International Journal of Computer Assisted Radiology and Surgery*, 10(10):1579–1588.
- Gildenberg, P. (1998). The History of Stereotactic and Functional Neurosurgery. In Gildenberg, P., Tasker, R., and Franklin, P., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 1, pages 5–19. McGraw-Hill Companies, 1st edition.
- Golash, A., Eldridge, P., Varma, T., Byrne, P., Badano, F., Nahum, B., and Pittet, P. (2000). 3-D error measurement for checking the application accuracy of a stereotactic robotic system with an infrared space digitisation technique: A phantom study and clinical use. In *Acta Neurochirurgica*, pages 1169–1210, Wien.
- Gomes, P. (2011). Surgical robotics: Reviewing the past, analysing the present, imagining the future. *Robotics and Computer-Integrated Manufacturing*, 27(2):261–266.
- Gonzalez-Martinez, J., Vadera, S., Mullin, J., Enatsu, R., Alexopoulos, A., Patwardhan, R., Bingaman, W., and Najm, I. (2014). Robot-assisted stereotactic laser ablation in medically intractable epilepsy: operative technique. *Neurosurgery*, 10 Suppl 2(2):167–72; discussion 172–3.
- Gorgulho, A., Ishida, W., and De Salles, A. (2009). General Imaging Modalities: Basic Principles. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 17, pages 252–267. Springer Verlag Berlin Heidelberg, 2nd edition.

- Greenberg, B., Rauch, S., and Haber, S. (2010). Invasive Circuitry-Based Neurotherapeutics: Stereotactic Ablation and Deep Brain Stimulation for OCD. *Neuropsychopharmacology*, 35(1):317–336.
- Greitz, D., Wirestam, R., Franck, A., Nordell, B., Thomsen, C., and Ståhlberg, F. (1992). Pulsatile brain movement and associated hydrodynamics studied by magnetic resonance phase imaging. *Neuroradiology*, 34(5):370–380.
- Grunert, P., Darabi, K., Espinosa, J., and Filippi, R. (2003). Computer-aided navigation in neurosurgery. *Neurosurgical Review*, 26(2):73–99.
- Guenot, M., Isnard, J., Ryvlin, P., Fischer, C., Ostrowsky, K., Mauguire, F., and Sindou, M. (2001). Neurophysiological Monitoring for Epilepsy Surgery: The Talairach SEEG Method. *Stereotactic and Functional Neurosurgery*, 77(1-4):29–32.
- Hadani, M., Spiegelman, R., Feldman, Z., Berkenstadt, H., and Ram, Z. (2001). Novel, compact, intra-operative magnetic resonance imaging-guided system for conventional neurosurgical operating rooms. *Neurosurgery*, 48(4):799–807; discussion 807–9.
- Hage, H., Bidaud, P., and Jardin, N. (2011). Practical consideration on the identification of the kinematic parameters of the Stäubli TX90 robot. In *World Congress in Mechanism and Machine Science*, pages 19–25.
- Haidegger, T., Benyó, Z., and Kazanzides, P. (2009). Patient motion tracking in the presence of measurement errors. In *IEEE Engineering in Medicine and Biology Society*, volume 2009, pages 5563–6, Minneapolis.
- Haidegger, T., Kovacs, L., Fordos, G., Benyo, Z., Kazanzides, P., Katashev, A., Dekhtyar, Y., Spigulis, J., and Magjarevic, R. (2008). Future Trends in Robotic Neurosurgery. *14th Nordic-Baltic Conference on Biomedical Engineering and Medical Physics*, 20:229–233.
- Handini, D. (2004). System integration of NeuroBot: a skull-base surgical robotic system. In *IEEE Conference on Robotics, Automation and Mechatronics*, volume 1, pages 43–48, Singapore. IEEE.
- Hariz, M. and Hariz, G. (2013). *Therapeutic stimulation versus ablation*, volume 116. Elsevier B.V., 1 edition.
- Hariz, M. and Zrinzo, L. (2009). CT/MRI Technology: Basic Principles. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 18, pages 269–278. Springer Verlag Berlin Heidelberg, 2nd edition.
- Harris, A., Hadjipanayis, C., Lunsford, L., Lunsford, A., and Kassam, A. (2005). Microsurgical Removal of Intraventricular Lesions Using Endoscopic Visualization and Stereotactic Guidance. *Operative Neurosurgery*, 56:125–132.
- Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 23.1–23.6. Alvey Vision Club.

- Hartov, A. and Roberts, D. (2009). Accuracy in Stereotactic and Image Guidance. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 28, pages 444–452. Springer Verlag Berlin Heidelberg, 2nd edition.
- Hata, N., Nabavi, A., Warfield, S., Wells, W., Kikinis, R., and Jolesz, F. (1999). A Volumetric Optical Flow Method for Measurement of Brain Deformation from Intraoperative Magnetic Resonance Images. In Taylor, C. and Colchester, A., editors, *Medical Image Computing and Computer-Assisted Intervention*, pages 928–935, Cambridge. Medical Image Computing and Computer-Assisted Intervention.
- Hayat, A., Chittawadigi, R., Udai, A., and Saha, S. (2013). Identification of Denavit-Hartenberg Parameters of an Industrial Robot. In *Proceedings of Conference on Advances In Robotics - AIR '13*, pages 1–6, New York, New York, USA. ACM Press.
- Hayati, S. (1983). Robot arm geometric link parameter estimation. In *The 22nd IEEE Conference on Decision and Control*, pages 1477–1483. IEEE.
- He, R., Zhao, Y., Yang, S., and Yang, S. (2010). Kinematic-Parameter Identification for Serial-Robot Calibration Based on POE Formula. *IEEE Transactions on Robotics*, 26(3):411–423.
- Hill, D., Maurer, C., Maciunas, R., Barwise, J., Fitzpatrick, J., and Wang, M. (1998). Measurement of intraoperative brain surface deformation under a craniotomy. *Neurosurgery*, 43(3):514–26; discussion 527–8.
- Hollerbach, J. M. (1985). Optimum kinematic design for a seven degree of freedom manipulator. Technical report, MIT Artificial Intelligence Laboratory, Cambridge.
- Hongo, K., Goto, T., Kakizawa, Y., Koyama, J., Kawai, T., Kan, K., Tanaka, Y., and Kobayashi, S. (2003). Micromanipulator system (NeuRobot): clinical application in neurosurgery. *International Congress Series*, 1256:509–513.
- Hongo, K., Goto, T., Miyahara, T., Kakizawa, Y., Koyama, J., and Tanaka, Y. (2006). Telecontrolled micro-manipulator system (NeuRobot) for minimally invasive neurosurgery. *Acta Neurochirurgica. Supplement*, 98:63–6.
- Horn, B. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(April):629–642.
- Horsley, V. and Clarke, R. (1908). The Structure and Functions of the Cerebellum Examined by a new Method. *Brain*, 31(1):45–124.
- Houeto, J. (2005). Tourette's syndrome and deep brain stimulation. *Journal of Neurology, Neurosurgery and Psychiatry*, 76(7):992–995.
- Hu, X., Ohnmeiss, D., and Lieberman, I. (2013). Robotic-assisted pedicle screw placement: lessons learned from the first 102 patients. *European Spine Journal*, 22(3):661–6.

- Ivaldi, S., Padois, V., and Nori, F. (2014). Tools for dynamics simulation of robots: a survey based on user feedback. Technical report, Cornell University.
- Jannin, P., Grova, C., and Maurer, C. (2006). Model for defining and reporting reference-based validation protocols in medical image processing. *International Journal of Computer Assisted Radiology and Surgery*, 1(2):63–73.
- Jeong, I., Khandwala, Y., Kim, J., Han, D., Li, S., Wang, Y., Chang, S., and Chung, B. (2017). Association of Robotic-Assisted vs Laparoscopic Radical Nephrectomy With Perioperative Outcomes and Health Care Costs, 2003 to 2015. *JAMA*, 318(16):1561–1568.
- Ji, S., Fan, X., Roberts, D., Hartov, A., and Paulsen, K. (2013). Tracking Cortical Surface Deformation Using Stereovision. In *Society for Experimental Mechanics Series*, pages 169–176. Springer New York.
- Ji, S., Fan, X., Roberts, D., and Paulsen, K. (2011). Cortical surface strain estimation using stereovision. In *Medical Image Computing and Computer-Assisted Intervention*, volume 14, pages 412–9.
- Jödicke, A., Deinsberger, W., Erbe, H., Kriete, A., and Böker, D. (1998). Intraoperative three-dimensional ultrasonography: an approach to register brain shift using multidimensional image processing. *Minimally Invasive Neurosurgery*, 41(1):13–9.
- Joskowicz, L., Shamir, R., Freiman, M., Shoham, M., Zehavi, E., Umansky, F., and Shoshan, Y. (2006). Image-guided system with miniature robot for precise positioning and targeting in keyhole neurosurgery. *Computer Aided Surgery*, 11(4):181–93.
- Joskowicz, L., Shamir, R., Israel, Z., Shoshan, Y., and Shoham, M. (2011). Renaissance Robotic System for Keyhole Cranial Neurosurgery : In-vitro Accuracy Study. In *Proceedings of the Simposio Mexicano en Cirugia Asistida por Computadora y Procesamiento de Imagenes Medicas*, pages 2–4, Iztapalapa. Universidad Autónoma Metropolitana.
- Joskowicz, L., Shoham, M., Shamir, R., Freiman, M., Zehavi, E., and Shoshan, Y. (2005). Miniature robot-based precise targeting system for keyhole neurosurgery: Concept and preliminary results. *International Congress Series*, 1281:618–623.
- Kajita, Y., Nakatsubo, D., Kataoka, H., Nagai, T., Nakura, T., and Wakabayashi, T. (2015). Installation of a Neuromate Robot for Stereotactic Surgery: Efforts to Conform to Japanese Specifications and an Approach for Clinical Use—Technical Notes. *Neurologia medico-chirurgica*, 55(12):907–914.
- Kapoor, A., Deguet, A., and Kazanzides, P. (2006). Software components and frameworks for medical robot control. In *IEEE International Conference on Robotics and Automation*, pages 3813–3818, Orlando. IEEE.
- Kazanzides, P., Deguet, A., and Kapoor, A. (2008). An Architecture for Safe and Efficient Multi-threaded Robot Software. In *Technologies for Practical Robot Applications*, pages 89–93, Woburn.

- Kazanzides, P., DiMaio, S., Deguet, A., Vagvolgyi, B., Balicki, M., Schneider, C., Kumar, R., Jog, A., Itkowitz, B., Hasser, C., and Taylor, R. (2010). The Surgical Assistant Workstation (SAW) in Minimally-Invasive Surgery and Microsurgery. *MIDAS journal*, 2010:2–9.
- Kazanzides, P., Jung, M., Deguet, A., Vagvolgyi, B., Balicki, M., and Taylor, R. (2011). Component-based software for dynamic configuration and control of computer assisted intervention systems: Release 1.0. *MIDAS journal*, 2011(2):2–9.
- Kazanzides, P., Zuhars, J., Mittelstadt, B., Williamson, B., Cain, P., Smith, F., Rose, L., and Musits, B. (1992). Architecture of a surgical robot. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1624–1629. IEEE.
- Kelly, P. (1992). Stereotactic Resection and Its Limitations in Glial Neoplasms. *Stereotactic and Functional Neurosurgery*, 59(1-4):84–91.
- Kelly, P. (1998). Volumetric Stereotactic Tumor Surgery. In Gildenberg, P. and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 58, pages 509–522. McGraw-Hill Companies, first edition.
- Ketchel, J. and Larochelle, P. (2006). Collision detection of cylindrical rigid bodies for motion planning. In *Proceedings 2006 IEEE International Conference on Robotics and Automation.*, pages 1530–1535. IEEE.
- Khalil, W., Besnard, S., Lemoine, P., Khalil, W., Besnard, S., and Lemoine, P. (2009). Comparison study of the geometric parameters calibration methods. *International Journal of Robotics and Automation*, 15(2):56–67.
- Kim, H. and Rosen, J. (2015). Predicting Redundancy of a 7 DOF Upper Limb Exoskeleton Toward Improved Transparency between Human and Robot. *Journal of Intelligent & Robotic Systems*, 80(S1):99–119.
- Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154, Sendai. IEEE.
- Koivukangas, J., Louhisalmi, Y., Alakuijala, J., and Oikarinen, J. (1993). Ultrasound-controlled neuronavigator-guided brain surgery. *Journal of Neurosurgery*, 79(1):36–42.
- Kortenkamp, D., Simmons, R., and Brugali, D. (2008). Robotic Systems Architectures and Programming. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 350–372. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Kratimenos, G., Thomas, D., Shorvon, S., and Fish, D. (1993). Stereotactic insertion of intracerebral electrodes in the investigation of epilepsy. *British Journal of Neurosurgery*, 7(1):45–52.

- Kreutz-Delgado, K., Long, M., and Seraji, H. (1992). Kinematic Analysis of 7-DOF Manipulators. *The International Journal of Robotics Research*, 11(5):469–481.
- Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems*, volume 58 of Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Kuhlemann, I., Schweikard, A., Jauer, P., and Ernst, F. (2016). Robust inverse kinematics by configuration control for redundant manipulators with seven DoF. In *International Conference on Control, Automation and Robotics*, pages 49–55, Hong Kong. IEEE.
- Kuncel, A. M. and Grill, W. M. (2004). Selection of stimulus parameters for deep brain stimulation. *Clinical Neurophysiology*, 115(11):2431–41.
- Kupsch, A., Benecke, R., Müller, J., Trottenberg, T., Schneider, G., Poewe, W., Eisner, W., Wolters, A., Müller, J., Deuschl, G., Pinsker, M., Skogseid, I., Roeste, G., Vollmer-Haase, J., Brentrup, A., Krause, M., Tronnier, V., Schnitzler, A., Voges, J., Nikkhah, G., Vesper, J., Naumann, M., and Volkmann, J. (2006). Pallidal Deep-Brain Stimulation in Primary Generalized or Segmental Dystonia. *New England Journal of Medicine*, 355(19):1978–1990.
- Kwoh, Y., Hou, J., Jonckheere, E., and Hayati, S. (1988). A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery. *IEEE Transactions on Biomedical Engineering*, 35(2):153–60.
- Lakie, M., Walsh, E., and Wright, G. (1986). Passive mechanical properties of the wrist and physiological tremor. *Journal of neurology, neurosurgery, and psychiatry*, 49(6):669–76.
- Lavallee, S., Troccaz, J., Gaborit, L., Cinquin, P., Benabid, A. L., and Hoffmann, D. (1992). Image guided operating robot: a clinical application in stereotactic neurosurgery. In *IEEE International Conference on Robotics and Automation*, pages 618–624, Nice. IEEE Comput. Soc. Press.
- Lee, S. and Bejczy, A. (1991). Redundant arm kinematic control based on parameterization. In *International Conference on Robotics and Automation*, pages 458–465, Sacramento. IEEE Comput. Soc. Press.
- Lefranc, M., Capel, C., Pruvot, A., Fichten, A., Desenclos, C., Toussaint, P., Le Gars, D., and Peltier, J. (2014a). The impact of the reference imaging modality, registration method and intraoperative flat-panel computed tomography on the accuracy of the ROSA® stereotactic robot. *Stereotactic and Functional neurosurgery*, 92(4):242–50.
- Lefranc, M., Capel, C., Pruvot-Ocean, A., Fichten, A., Desenclos, C., Toussaint, P., Le Gars, D., and Peltier, J. (2014b). Frameless robotic stereotactic biopsies: a consecutive series of 100 cases. *Journal of Neurosurgery*, pages 1–11.
- Lefranc, M. and Le Gars, D. (2012). Robotic implantation of deep brain stimulation leads, assisted by intra-operative, flat-panel CT. *Acta Neurochirurgica*, 154(11):2069–74.

- Lewis, J. (1995). Fast template matching. *Vision Interface*, 1(1):1–7.
- Li, Q., Zamorano, L., Pandya, A., Perez, R., Gong, J., and Diaz, F. (2002). The application accuracy of the NeuroMate robot—A quantitative comparison with frameless and frame-based surgical localization systems. *Computer Aided Surgery*, 7(2):90–8.
- Lima, E., Rodrigues, P., Mota, P., Carvalho, N., Dias, E., Correia-Pinto, J., Autorino, R., and Vilaça, J. (2017). Ureterscopy-assisted Percutaneous Kidney Access Made Easy: First Clinical Experience with a Novel Navigation System Using Electromagnetic Guidance (IDEAL Stage 1). *European Urology*, 72(4):610–616.
- Linninger, A., Tsakiris, C., Zhu, D., Xenos, M., Roycewicz, P., Danziger, Z., and Penn, R. (2005). Pulsatile cerebrospinal fluid dynamics in the human brain. *IEEE Transactions on Biomedical Engineering*, 52(4):557–65.
- Lippold, O. (1970). Oscillation in the stretch reflex arc and the origin of the rhythmical, 8-12 C-S component of physiological tremor. *The Journal of physiology*, 206(2):359–82.
- Liu, J., Zhang, Y., and Li, Z. (2006). The Application Accuracy of NeuroMaster: a Robot System for Stereotactic Neurosurgery. In *International Conference on Mechatronics and Embedded Systems and Applications*, number 37, pages 1–5. IEEE.
- Lollis, S. and Roberts, D. (2009). Robotic placement of a CNS ventricular reservoir for administration of chemotherapy. *British Journal of Neurosurgery*, 23(5):516–20.
- Lozano, A., Gildenberg, P., and Tasker, R. (2009). *Textbook of Stereotactic and Functional Neurosurgery*. Springer Verlag Berlin Heidelberg, 2nd edition.
- Luck, C. L. and Lee, S. (1993). Self-motion topology for redundant manipulators with joint limits. In *International Conference on Robotics and Automation*, pages 626–631, Atlanta. IEEE Comput. Soc. Press.
- Lunsford, L., Kondziolka, D., and Bissonette, D. (1996). Intraoperative imaging of the brain. *Stereotactic and Functional Neurosurgery*, 66(1-3):58–64.
- Machado, A., Rezai, A., Kopell, B., Gross, R., Sharan, A., and Benabid, A. (2006). Deep brain stimulation for Parkinson’s disease: surgical technique and perioperative management. *Movement Disorders*, 21 Suppl 1(S14):S247–S258.
- Machado, T., Sousa, M., Monteiro, S., and Bicho, E. (2012). CoopDynSim: a 3D robotics simulator. In *12th International Conference on Autonomous Robot Systems and Competitions.*, pages 45–51, Guimarães, Portugal. University of Minho.
- Maciunas, R., Galloway, R., and Latimer, J. (1994). The application accuracy of stereotactic frames. *Neurosurgery*, 35(4):682–695.

- Magill, S., Lau, D., Raleigh, D. R., Sneed, P. K., Fogh, S. E., and McDermott, M. W. (2017). Surgical Resection and Interstitial Iodine-125 Brachytherapy for High-Grade Meningiomas. *Neurosurgery*, 80(3):409–416.
- Makarenko, A., Brooks, A., and Kaupp, T. (2006). Orca: Components for Robotics. In *International Conference on Intelligent Robots and Systems*, pages 163–168, Beijing.
- Marcus, H., Seneci, C., Payne, C., Nandi, D., Darzi, A., and Yang, G. (2014). Robotics in keyhole transcranial endoscope-assisted microsurgery: a critical review of existing systems and proposed specifications for new robotic platforms. *Neurosurgery*, 10 Suppl 1(1):84–95; discussion 95–6.
- Marks, W. (2010). Introduction: The expanding role of deep brain stimulation. In Marks Jr., W. J., editor, *Deep Brain Stimulation Management*, chapter 1, pages 1–3. Cambridge University Press, 1st edition.
- Mattei, T., Rodriguez, A., Sambhara, D., and Mendel, E. (2014). Current state-of-the-art and future perspectives of robotic technology in neurosurgery. *Neurosurgical Review*.
- Mayberg, H., Lozano, A., Voon, V., McNeely, H., Seminowicz, D., Hamani, C., Schwalb, J., and Kennedy, S. (2005). Deep Brain Stimulation for Treatment-Resistant Depression. *Neuron*, 45(5):651–660.
- McGirt, M., Woodworth, G., Coon, A., Frazier, J., Amundson, E., Garonzik, I., Olivi, A., and Weingart, J. (2005). Independent predictors of morbidity after image-guided stereotactic brain biopsy: a risk assessment of 270 cases. *Journal of Neurosurgery*, 102(5):897–901.
- Medtech S.A (2010). Premarket Notification 510(K) Submission Rosa Surgical Device. Technical report, Montpellier.
- Medtech S.A (2012). Press-Publication, Available at: <http://www.medtechsurgical.com/Press-room/Peer-review>.
- Metta, G., Fitzpatrick, P., and Natale, L. (2006). YARP: Yet Another Robot Platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48.
- Miga, M., Sinha, T., Cash, D., Galloway, R., and Weil, R. (2003). Cortical surface registration for image-guided neurosurgery using laser-range scanning. *IEEE Transactions on Medical Imaging*, 22(8):973–85.
- Mink, J. W., Walkup, J., Frey, K. A., Como, P., Cath, D., Delong, M. R., Erenberg, G., Jankovic, J., Juncos, J., Leckman, J. F., Swerdlow, N., Visser-Vandewalle, V., and Vitek, J. L. (2006). Patient selection and assessment recommendations for deep brain stimulation in Tourette syndrome. *Movement Disorders*, 21(11):1831–8.
- Moniz, E., Carvalho, L., and Lima, A. (1931). Angiopneumographie. *Presse med*, 39:996–999.
- Montgomery, E. B. (2010). *Deep Brain Stimulation Programming: Principles and Practice*. Oxford University Press, Inc., Oxford, 1st edition.

- Moradi, H. and Lee, S. (2005). Joint Limit Analysis and Elbow Movement Minimization for Redundant Manipulators Using Closed Form Method. In International Conference on Intelligent Computing, pages 423–432. Springer, Berlin, Heidelberg.
- Morgan, P., Carter, T., Davis, S., Sepehri, A., Punt, J., Byrne, P., Moody, A., and Finlay, P. (2003). The application accuracy of the Pathfinder neurosurgical robot. *International Congress Series*, 1256:561–567.
- Motkoski, J. and Sutherland, G. (2014). Progress in Neurosurgical Robotics. In Jolesz, F. A., editor, *Intraoperative Imaging and Image-Guided Therapy*, chapter 46, pages 601–612. Springer New York, New York, NY, 1st edition.
- Mozdy, M. (2017). *Inventing the N-Localizer for Stereotactic Neurosurgery: the Story of a Young Researcher in Radiology and Imaging Sciences*.
- Munyon, C., Sweet, J., Luders, H., Lhatoo, S., and Miller, J. (2015). The 3-Dimensional Grid: A Novel Approach to Stereoelectroencephalography. *Operative Neurosurgery*, 11(1):127–134.
- Muradore, R., Fiorini, P., Akgun, G., Barkana, D., Bonfe, M., Boriero, F., Caprara, A., De Rossi, G., Dodi, R., Elle, O., Ferraguti, F., Gasperotti, L., Gassert, R., Mathiassen, K., Handini, D., Lamercy, O., Li, L., Kruusmaa, M., Manurung, A., Meruzzi, G., Nguyen, H., Preda, N., Riolfo, G., Ristolainen, A., Sanna, A., Secchi, C., Torsello, M., and Yantac, A. (2015). Development of a Cognitive Robotic System for Simple Surgical Tasks. *International Journal of Advanced Robotic Systems*, 12(4):37.
- Nachbichler, S. and Kreth, F. (2018). *Intracranial Gliomas Part II - Adjuvant Therapy*, volume 31 of *Progress in Neurological Surgery*. S. Karger AG.
- Nakamura, Y. and Hanafusa, H. (1986). Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163.
- Nakamura, Y., Hanafusa, H., and Yoshikawa, T. (1987). Task-Priority Based Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research*, 6(2):3–15.
- Nathoo, N., Cavişoğlu, M. C., Vogelbaum, M. A., and Barnett, G. H. (2005). In touch with robotics: neurosurgery for the future. *Neurosurgery*, 56(3):421–33; discussion 421–33.
- Nathoo, N., Pesek, T., and Barnett, G. H. (2003). Robotics and neurosurgery. *The Surgical Clinics of North America*, 83(6):1339–50.
- Nelson, B. J. and Khosla, P. K. (1995). Strategies for Increasing the Tracking Region of an Eye-in-Hand System by Singularity and Joint Limit Avoidance. *The International Journal of Robotics Research*, 14(3):255–269.
- Nenchev, D. N., Tsumaki, Y., and Takahashi, M. (2004). Singularity-consistent kinematic redundancy resolution for the S-R-S manipulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), volume 4, pages 3607–3612. IEEE.

- Nimsky, C., Ganslandt, O., Cerny, S., Hastreiter, P., Greiner, G., and Fahlbusch, R. (2000). Quantification of, visualization of, and compensation for brain shift using intraoperative magnetic resonance imaging. *Neurosurgery*, 47(5):1070–9; discussion 1079–80.
- Nimsky, C., Rachinger, J., Iro, H., and Fahlbusch, R. (2004). Adaptation of a hexapod-based robotic system for extended endoscope-assisted transsphenoidal skull base surgery. *Minimally Invasive Neurosurgery*, 47(1):41–6.
- Nowinski, W. (2009). Anatomical and Probabilistic Functional Atlases in Stereotactic and Functional Neurosurgery. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 27, pages 395–441. Springer Verlag Berlin Heidelberg, 2nd edition.
- Omron (2019). *Robotic Automation Industrial Robots Datasheets - Cobra 650*.
- Ormond, R. and Hadjipanayis, C. (2014). The history of neurosurgery and its relation to the development and refinement of the frontotemporal craniotomy. *Neurosurgical Focus*, 36(4):E12.
- Ortler, M., Sohm, F., Eisner, W., Bauer, R., Dobesberger, J., Trinka, E., Widmann, G., and Bale, R. (2011). Frame-Based vs Frameless Placement of Intrahippocampal Depth Electrodes in Patients With Refractory Epilepsy: A Comparative in Vivo (Application) Study. *Neurosurgery*, 68(4):881–887.
- Pandya, S., Motkoski, J., Serrano-Almeida, C., Greer, A., Latour, I., and Sutherland, G. (2009). Advancing neurosurgery with image-guided robotics. *Journal of Neurosurgery*, 111(6):1141–9.
- Paul, P., Fleig, O., and Jannin, P. (2005). Augmented virtuality based on stereoscopic reconstruction in multimodal image-guided neurosurgery: methods and performance evaluation. *IEEE Transactions on Medical Imaging*, 24(11):1500–11.
- Paul, P., Morandi, X., and Jannin, P. (2009). A surface registration method for quantification of intraoperative brain deformations in image-guided neurosurgery. *IEEE Transactions on Information Technology in Biomedicine*, 13(6):976–83.
- Perlmutter, J. and Mink, J. (2006). Deep brain stimulation. *Annual Review of Neuroscience*, 29(1):229–57.
- Pillay, P. (1997). Image-guided stereotactic neurosurgery with the multicoordinate manipulator microscope. *Surgical Neurology*, 47(2):171–176.
- Pincirolì, C., Trianni, V., O’Grady, R., Pini, G., Bruschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L. M., and Dorigo, M. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.
- Pitonakova, L., Giuliani, M., and Winfield, A. (2018). Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators. In *Annual Conference Towards Autonomous Robotic Systems*, number July, pages 357–368, Bristol. Springer.

- Poncelet, B., Wedeen, V., Weisskoff, R., and Cohen, M. (1992). Brain parenchyma motion: measurement with cine echo-planar MR imaging. *Radiology*, 185(3):645–51.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, 3(3.2):5.
- Rajeevlochana, C., Subir, S., and Shivesh, K. (2012). Automatic Extraction of DH Parameters of Serial Manipulators using Line Geometry. In *The 2nd Joint International Conference on Multibody System Dynamics*, pages 1–9, Stuttgart.
- Rasin, I., Pekar, Z., Sadowsky, O., Forte, A., Galvan, S., Dini, D., Shoham, M., and Joskowicz, L. (2014). Real-Time Modeling of Intra-operative Brain Shift Based on Video Tracking. In *Proceedings of The Hamlyn Symposium on Medical Robotics*, London.
- Ringel, F., Stüer, C., Reinke, A., Preuss, A., Behr, M., Auer, F., Stoffel, M., and Meyer, B. (2012). Accuracy of robot-assisted placement of lumbar and sacral pedicle screws: a prospective randomized comparison to conventional freehand screw implantation. *Spine*, 37(8):E496–501.
- Roberts, D., Hartov, A., Kennedy, F., Miga, M., and Paulsen, K. (1998). Intraoperative brain shift and deformation: a quantitative analysis of cortical displacement in 28 cases. *Neurosurgery*, 43(4):749–58; discussion 758–60.
- Robotics, K. (2009). KINOVA MICO Robotic arm - User Guide.
- Rodrigues, P., Vilaça, J., Oliveira, C., Cicione, A., Rassweiler, J., Fonseca, J., Rodrigues, N., Correia-Pinto, J., and Lima, E. (2013). Collecting System Percutaneous Access Using Real-Time Tracking Sensors: First Pig Model In Vivo Experience. *Journal of Urology*, 190(5):1932–1937.
- Roessler, K., Czech, T., Dietrich, W., Ungersboeck, K., Nasel, C., Hainfellner, J., and Koos, W. (1998). Frameless stereotactic-directed tissue sampling during surgery of suspected low-grade gliomas to avoid histological undergrading. *Minimally Invasive Neurosurgery*, 41(4):183–6.
- Roessler, K., Ungersboeck, K., Czech, T., Aichholzer, M., Dietrich, W., Goerzer, H., Matula, C., and Koos, W. (1997). Contour-guided brain tumor surgery using a stereotactic navigating microscope. *Stereotactic and Functional Neurosurgery*, 68(1-4 Pt 1):33–8.
- Rohmer, E., Singh, S., and Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE.
- Sanchez, A., Poignet, P., Dombre, E., Menciassi, A., and Dario, P. (2014). A design framework for surgical robots: Example of the Araknes robot controller. *Robotics and Autonomous Systems*, 62(9):1342–1352.
- Sandoval, J., Su, H., Vieyres, P., Poisson, G., Ferrigno, G., and De Momi, E. (2018). Collaborative framework for robot-assisted minimally invasive surgery using a 7-DoF anthropomorphic robot. *Robotics and Autonomous Systems*, 106:95–106.

- Schouten, M., Ansems, J., Renema, W., Bosboom, D., Scheenen, T., and Fütterer, J. (2010). The accuracy and safety aspects of a novel robotic needle guide manipulator to perform transrectal prostate biopsies. *Medical Physics*, 37(9):4744–4750.
- Schreiber, G., Stemmer, A., and Bischoff, R. (2010). The Fast Research Interface for the KUKA Lightweight Robot. In *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers*, pages 15–21.
- Schulder, M. (2003). *Handbook of Stereotactic and Functional Neurosurgery*. CRC Press, 1st edition.
- Schulder, M. and Oubré, A. (2009). CT/MRI safety in functional neurosurgery. In Lozano, A., Gildenberg, P., and Tasker, R., editors, *Textbook of Stereotactic and Functional Neurosurgery*, chapter 19, pages 279–286. Springer Verlag Berlin Heidelberg, 2nd edition.
- Seijo, F., Alvarez-Vega, M., Lozano, B., Fernández-González, F., Santamarta, E., and Saiz, A. (2009). Common Questions and Answers to Deep Brain Stimulation Surgery. In Rogers, M. H. and Anderson, P. B., editors, *Deep Brain Stimulation: Applications, Complications And Side Effects*, chapter 1, pages 1–29. Nova Science Publishers, New York, first edition.
- Seraji, H. (1989). Configuration control of redundant manipulators: theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–490.
- Serletis, D., Bulacio, J., Bingaman, W., Najm, I., and González-Martínez, J. (2014). The stereotactic approach for mapping epileptic networks: a prospective study of 200 patients. *Journal of Neurosurgery*, 121(5):1239–46.
- Shamir, R., Freiman, M., Joskowicz, L., Shoham, M., Zehavi, E., and Shoshan, Y. (2005). Robot-assisted image-guided targeting for minimally invasive neurosurgery: planning, registration, and in-vitro experiment. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 8(Pt 2):131–8.
- Shimizu, M., Kakuya, H., Yoon, W.-K., Kitagaki, K., and Kosuge, K. (2008). Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution. *IEEE Transactions on Robotics*, 24(5):1131–1142.
- Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH computer graphics*, 19(3):245–254.
- Shoham, M., Lieberman, I., Benzel, E., Togawa, D., Zehavi, E., Zilberstein, B., Roffman, M., Bruskin, A., Fridlander, A., Joskowicz, L., Brink-Danan, S., and Knoller, N. (2007). Robotic assisted spinal surgery-from concept to clinical practice. *Computer Aided Surgery*, 12(2):105–115.
- Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, London, 1st edition.

- Siegfried, J. and Lippitz, B. (1994). Chronic Electrical Stimulation of the VL-VPL Complex and of the Pallidum in the Treatment of Movement Disorders: Personal Experience since 1982. *Stereotactic and Functional Neurosurgery*, 62(1-4):71–75.
- Singh, G. K. and Claassens, J. (2010). An analytical solution for the inverse kinematics of a redundant 7DoF Manipulator with link offsets. In *International Conference on Intelligent Robots and Systems*, pages 2976–2982, Taipei. IEEE.
- Sinha, T., Dawant, B., Duay, V., Cash, D., Weil, R., Thompson, R., Weaver, K., and Miga, M. (2005). A method to track cortical surface deformations using a laser range scanner. *IEEE Transactions on Medical Imaging*, 24(6):767–81.
- Sivakumar, G., Smith, S., and Morgan, P. (2003). The bionic surgeon: Initial Experience with the PathFinder Robot.
- Skrinjar, O., Nabavi, A., and Duncan, J. (2002). Model-driven brain shift compensation. *Medical Image Analysis*, 6(4):361–73.
- Spetzger, U., Hubbe, U., Struffert, T., Reinges, M. H. T., Krings, T., Krombach, G. A., Zentner, J., Gilsbach, J. M., and Stiehl, H. S. (2002). Error Analysis in Cranial Neuronavigation. *Minimally Invasive Neurosurgery*, 45(1):6–10.
- Spiegel, E., Wycis, H., Marks, M., and Lee, A. (1947). Stereotaxic Apparatus for Operations on the Human Brain. *Science*, 106(2754):349–350.
- Spire, W., Jobst, B., Thadani, V., Williamson, P., Darcey, T., and Roberts, D. (2008). Robotic image-guided depth electrode implantation in the evaluation of medically intractable epilepsy. *Neurosurgical focus*, 25(3):1–5.
- Starr, P., Martin, A., Ostrem, J., Talke, P., Levesque, N., and Larson, P. (2010). Subthalamic nucleus deep brain stimulator placement using high-field interventional magnetic resonance imaging and a skull-mounted aiming device: technique and application accuracy. *Journal of neurosurgery*, 112(3):479–90.
- Stone, H. (1987). *Kinematic Modeling, Identification, and Control of Robotic Manipulators*, volume 29 of *The Kluwer International Series in Engineering and Computer Science*. Springer US, Boston, MA, 1st edition.
- Studholme, C., Hill, D., and Hawkes, D. (1999). An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, 32(1):71–86.
- Suh, K. and Hollerbach, J. (1987). Local versus global torque optimization of redundant manipulators. In *International Conference on Robotics and Automation*, volume 4, pages 619–624, Raleigh. IEEE.

- Sun, H., Farid, H., Rick, K., Hartov, A., Roberts, D., and Paulsen, K. (2003). Estimating Cortical Surface Motion Using Stereopsis for Brain Deformation Models. In Ellis, R. and Peters, T., editors, *Medical Image Computing and Computer-Assisted Intervention*, volume d, pages 794–801, Montréal. Springer Berlin Heidelberg.
- Sun, H., Roberts, D., Farid, H., Wu, Z., Hartov, A., and Paulsen, K. (2005). Cortical Surface Tracking Using a Stereoscopic Operating Microscope. *Neurosurgery*, 56(January):86–97.
- Sutherland, G., Latour, I., and Greer, A. (2008). Integrating an image-guided robot with intraoperative MRI: a review of the design and construction of neuroArm. *IEEE Engineering in Medicine and Biology Magazine*, 27(3):59–65.
- Sutherland, G., McBeth, P., and Louw, D. (2003). NeuroArm: an MR compatible robot for microsurgery. *International Congress Series*, 1256:504–508.
- Švaco, M., Šekoranja, B., Šuligoj, F., Vidaković, J., Jerbić, B., and Chudy, D. (2017). A Novel Robotic Neuronavigation System: RONNA G3. *Journal of Mechanical Engineering*, 63(12):725–735.
- Talairach, J., Aboulker, P., Ruggiero, G., and David, M. (1954). Using a stereotactic radio method for in situ radioactive treatment of brain tumors. *Revue neurologique*, 90(5):656–7.
- Talairach, J. and Bancaud, J. (1973). Stereotaxic Approach to Epilepsy. In Krayenbühl, H., Maspes, P., and Sweet, W., editors, *Progress in Neurological Surgery*, pages 297–354. Karger.
- Talairach, J., Bancaud, J., Bonis, A., Szikla, G., and Tournoux, P. (1962). Functional Stereotaxic Exploration of Epilepsy. *Stereotactic and Functional Neurosurgery*, 22(3-5):328–331.
- Talairach, J. and Tournoux, P. (1988). *Co-Planar Stereotaxic Atlas of the Human Brain*. Thieme Publishers, 1st edition.
- Talamini, M., Chapman, S., Horgan, S., and Melvin, W. (2003). A prospective analysis of 211 robotic-assisted surgical procedures. *Surgical Endoscopy*, 17(10):1521–4.
- Taylor, R., Mittelstadt, B., Paul, H., Hanson, W., Kazanzides, P., Zuhars, J., Williamson, B., Musits, B., Glassman, E., and Bargar, W. (1994). An image-directed robotic system for precise orthopaedic surgery. *IEEE Transactions on Robotics and Automation*, 10(3):261–275.
- Taylor, R., Paul, H., Kazanzides, P., Mittelstadt, B., Hanson, W., Zuhars, J., Williamson, B., Musits, B., Glassman, E., and Bargar, W. (1991). Taming the bull: safety in a precise surgical robot. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pages 865–870 vol.1. IEEE.
- Taylor, R. and Stoianovici, D. (2003). Medical robotics in computer-integrated surgery. *IEEE Transactions on Robotics and Automation*, 19(5):765–781.

- Tolani, D., Goswami, A., and Badler, N. I. (2000). Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. *Graphical Models*, 62(5):353–388.
- Tondu, B. (2006). A closed-form inverse kinematic modelling of a 7R anthropomorphic upper limb based on a joint parametrization. In *International Conference on Humanoid Robots*, pages 390–397, Genoa. IEEE.
- Tronnier, V., Bonsanto, M., Staubert, A., Knauth, M., Kunze, S., and Wirtz, C. (2001). Comparison of intraoperative MR imaging and 3D-navigated ultrasonography in the detection and resection control of lesions. *Neurosurgical Focus*, 10(2):E3.
- Universal Robots (2019). UR3 - Technical Details.
- Unsgaard, G., Gronningsaeter, A., Ommedal, S., and Nagelhus Hernes, T. (2002). Brain operations guided by real-time two-dimensional ultrasound: new possibilities as a result of improved image quality. *Neurosurgery*, 51(2):402–11; discussion 411–2.
- Vahrenkamp, N., Asfour, T., Metta, G., Sandini, G., and Dillmann, R. (2012). Manipulability analysis. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 568–573. IEEE.
- Varma, T. and Eldridge, P. (2006). Use of the NeuroMate stereotactic robot in a frameless mode for functional neurosurgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2(2):107–13.
- Varma, T., Eldridge, P., Forster, A., Fox, S., Fletcher, N., Steiger, M., Littlechild, P., Byrne, P., Sinnott, A., Tyler, K., and Flintham, S. (2003). Use of the NeuroMate Stereotactic Robot in a Frameless Mode for Movement Disorder Surgery. *Stereotactic and Functional Neurosurgery*, 80(1-4):132–135.
- Vaughan, R., Gerkey, B., and Howard, A. (2004). On device abstractions for portable, reusable robot code. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 2421–2427, Las Vegas. IEEE.
- Veitschegger, W. and Chi-haur Wu (1987). A method for calibrating and compensating robot kinematic errors. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 39–44. Institute of Electrical and Electronics Engineers.
- Vidailhet, M., Vercueil, L., Houeto, J., Krystkowiak, P., Benabid, A., Cornu, P., Lagrange, C., Tézenas du Montcel, S., Dormont, D., Grand, S., Blond, S., Detante, O., Pillon, B., Ardouin, C., Agid, Y., Destée, A., and Pollak, P. (2005). Bilateral Deep-Brain Stimulation of the Globus Pallidus in Primary Generalized Dystonia. *New England Journal of Medicine*, 352(5):459–467.
- Vitek, J. (2002). Mechanisms of deep brain stimulation: excitation or inhibition. *Movement Disorders*, 17 Suppl 3:S69–72.

- von Langsdorff, D., Paquis, P., and Fontaine, D. (2014). In vivo measurement of the frame-based application accuracy of the Neuromate neurosurgical robot. *Journal of Neurosurgery*, pages 1–4.
- Wagshul, M., Eide, P., and Madsen, J. (2011). The pulsating brain: A review of experimental and clinical studies of intracranial pulsatility. *Fluids and Barriers of the CNS*, 8(5):1–23.
- Wang, M. N. and Song, Z. J. (2011). Classification and Analysis of the Errors in Neuronavigation. *Neurosurgery*, 68(4):1131–1143.
- Willems, P., Noordmans, H., Berkelbach van der Sprenkel, J., Viergever, M., and Tulleken, C. (2001). An MKM-mounted instrument holder for frameless point-stereotactic procedures: a phantom-based accuracy evaluation. *Journal of Neurosurgery*, 95(6):1067–74.
- Willems, P., Noordmans, H., Ramos, L., Taphoorn, M., Berkelbach van der Sprenkel, J., Viergever, M., and Tulleken, C. (2003). Clinical evaluation of stereotactic brain biopsies with an MKM-mounted instrument holder. *Acta Neurochirurgica*, 145(10):889–97; discussion 897.
- Wittek, A., Miller, K., Kikinis, R., and Warfield, S. (2007). Patient-specific model of brain deformation: application to medical image registration. *Journal of Biomechanics*, 40(4):919–29.
- Wu, Y., Klimchik, A., Caro, S., Furet, B., and Pashkevich, A. (2015). Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments. *Robotics and Computer-Integrated Manufacturing*, 35:151–168.
- Xia, T., Baird, C., Jallo, G., Hayes, K., Nakajima, N., Hata, N., and Kazanzides, P. (2008). An integrated system for planning, navigation and robotic assistance for skull base surgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 4(4):321–30.
- Yan, L., Mu, Z., and Xu, W. (2014). Analytical inverse kinematics of a class of redundant manipulator based on dual arm-angle parameterization. In *International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3744–3749, San Diego. IEEE.
- Yang, X., Wu, L., Li, J., and Chen, K. (2014). A minimal kinematic model for serial robot calibration using POE formula. *Robotics and Computer-Integrated Manufacturing*, 30(3):326–334.
- Yoshikawa, T. (1985a). Manipulability and redundancy control of robotic mechanisms. In *International Conference on Robotics and Automation*, volume 2, pages 1004–1009, St. Louis. Institute of Electrical and Electronics Engineers.
- Yoshikawa, T. (1985b). Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research*, 4(2):3–9.
- Yoshikawa, T. (1991). Translational and rotational manipulability of robotic manipulators. In *International Conference on Industrial Electronics, Control and Instrumentation*, pages 1170–1175, Kobe. IEEE.

- Zeng, B., Meng, F., Ding, H., and Wang, G. (2017). A surgical robot with augmented reality visualization for stereoelectroencephalography electrode implantation. *International Journal of Computer Assisted Radiology and Surgery*, 12(8):1355–1368.
- Zhou, Z.-L. and Nguyen, C. C. (1997). Joint configuration conservation and joint limit avoidance of redundant manipulators. In *International Conference on Robotics and Automation*, volume 3, pages 2421–2426, Albuquerque. IEEE.
- Zimmermann, M., Krishnan, R., Raabe, A., and Seifert, V. (2004). Robot-assisted navigated endoscopic ventriculostomy: implementation of a new technology and first clinical results. *Acta Neurochirurgica*, 146(7):697–704.
- Zrinzo, L. (2012). Pitfalls in precision stereotactic surgery. *Surgical Neurology International*, 3(Suppl 1):S53–61.