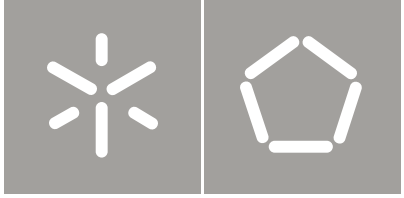
Universidade do Minho
Escola de Engenharia

Carlos André de Oliveira Faria

Robotic Implantation of Intracerebral Electrodes for Deep Brain Stimulation

Outubro de 2012

Robotic Implantation of Intracerebral Electrodes for Deep Brain Stimulation
Carlos André de Oliveira Faria
UMinho | 2012



Universidade do Minho
Escola de Engenharia

Carlos André de Oliveira Faria

Robotic Implantation of Intracerebral Electrodes for Deep Brain Stimulation

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Biomédica

Trabalho efetuado sob a orientação de
Professora Doutora Estela Bicho
Neurocirurgião Doutor Manuel Rito

Outubro de 2012

Page intentionally left blank.

Para os meus pais e irmão,

Acknowledgements

I want to express a very special thanks to my scientific advisor Prof. Estela Bicho, for the opportunity to work in this fascinating project, for the extent knowledge, experience and guidance provided. Her unwavering support and enthusiasm were deeply appreciated in both good and difficult moments and a determinant factor for the final outcome of this work.

I would like to thank my co-advisor Dr. Manuel Rito for proposing the idea of the dissertation, for the motivation, support and availability. His knowledge and experience were fundamental.

I would also like to express my gratitude to all my lab colleagues, Eliana Silva, Emanuel Sousa, Flora Ferreira, Hung Vu, Luís Louro, Miguel Sousa, Rui Silva, Tiago Malheiro and Toni Machado. To them i am sincerely thankful for all they taught me, for their warm welcoming and for their friendship.

I want to thank Coimbra University Hospitals Neurosurgery Service for the availability and welcoming environment, and for the opportunity to visit a neurosurgery operating room and attend a real surgery.

I want to thank Prof. Wolfram Erlhagen for the opportunity to work on the project Pest-C/MAT-UI0013/2011 with the Centre of Mathematics of the University of Minho with the grant support provided by "Fundação para a Ciência e a Tecnologia" with the reference UMINHO/BIC/8/2012.

I would like to finish my acknowledgments by expressing my deepest gratitude to my parents and my brother for the unconditional support in all moments of my life, because without them it would be impossible to have reached this far.

Abstract

The objective of this dissertation is to develop an initial approach of a robotic system to play an assistive role in Deep Brain Stimulation (DBS) stereotactic neurosurgery. The robot is expected to position and manipulate several surgical instrumentation in a passive or semi-active role according to pre-operative directives and to medical team instructions. The current impact of neurological disorders sensitive to DBS, the underlying knowledge of neurostimulation and neuroanatomy, and practical insight about DBS surgery is studied to understand the ultimate goal of our project. We elaborated a state of the art search on neurosurgery robots to get the picture of what was done and what could be improved. Upon determining the optimal robotic system characteristics for DBS surgery, we conducted a search on industrial robotic manipulators to select the best candidates. The geometric and differential kinematic equations are developed for each robotic manipulator. To test the kinematic equations and the control application in a virtual operating room environment, we used the *CoopDynSim* simulator. Being this simulator oriented to mobile robots, we introduced the serial manipulator concept and implemented the selected robots with all specifications. We designed a control application to manoeuvre the robot and devised an initial interface towards positioning/manipulation of instrumentation along surgical trajectories, while emphasizing safety procedures. Although it was impossible to assess the robot's precision in simulation, we studied how and where to place the manipulator to avoid collisions with surrounding equipment without restricting its flexibility.

Keywords: *Deep Brain Stimulation; Robotic Neurosurgery; Non-redundant and Redundant Manipulator Kinematics; 3D Robotics Simulator*

Resumo

O objectivo desta dissertação é o desenvolvimento de uma abordagem inicial a um sistema robótico para desempenhar um papel de assistência em neurocirurgia estereotáxica de Estimulação Cerebral Profunda (DBS). O robô deve posicionar e manipular variados instrumentos cirúrgicos de uma forma passiva ou semi-ativa de acordo com diretivas pré-operativas ou com as instruções da equipa médica. O impacto atual dos distúrbios neurológicos sensíveis a DBS, o conhecimento subjacente de neuro-estimulação e neuro-anatomia, e conhecimento prático sobre a cirurgia de DBS são estudados para concluir sobre o objectivo final do nosso projeto. Nós elaborámos uma pesquisa sobre o estado da arte em robots neurocirúrgicos para perceber o que tem sido feito e o que pode ser melhorado. Após determinar o conjunto óptimo de características de um sistema robótico para cirurgia de DBS, nós procuramos manipuladores robóticos industriais para escolher os melhores candidatos. As cinemáticas geométricas e diferenciais são desenvolvidas para cada manipulador robótico. Para testar as equações cinemáticas e a aplicação de controlo num ambiente virtual de uma sala de operações, nós usamos o simulador *CoopDynSim*. Sendo este manipulador orientado a robôs móveis, nós introduzimos o conceito de manipuladores em série e implementamos os robôs selecionados com todas as especificações. Nós projetamos uma aplicação de controlo para manobrar os robôs e desenvolvemos uma interface inicial no sentido do posicionamento/manipulação de instrumentação ao longo de trajetórias cirúrgicas, enfatizando os procedimentos de segurança. Embora não tenha sido possível avaliar a precisão do robô em simulação, nós estudamos como e onde posicionar o manipulador de forma a evitar colisões com o equipamento circundante sem restringir a sua flexibilidade.

Contents

Nomenclature

BDI Beck Depression Inventory

CNS Central Nervous System

CT Computed Tomography

DALY Disability Adjusted Life Year

DBS Deep Brain Stimulation

DOF Degrees of Freedom

ET Essential Tremor

FDA Food and Drug Administration

GABA Gamma-AminoButyric Acid

GBD Global Burden of Disease

GPe Globus Pallidus Externus

GPi Globus Pallidus Internus

HRI Human Robot Interaction

Hz Hertz

IPG Implanted Pulse Generator

MDRS Mattis Dementia Rating Scale

MMES Mini Mental Status Examination

MRI Magnetic Resonance Imaging

PD Parkinson's Disease

PPN Pedunculopontine Nucleus

RMIS Robotic-assisted Minimally Invasive Surgery

SNc Substantia Nigra (pars compacta)

SNr Substantia Nigra (pars reticulata)

STN Subthalamic Nucleus

UI User Interface

UPDRS III Unified Parkinson's Disease Rating Scale, Part III(Motor Subscale)

VMI Ventral Intermediate Nucleus of the Thalamus

WHO World Health Organization

YLD Years Lived with Disability

YLL Years of Life Lost

List of Figures

List of Tables

Part I

Introduction

Chapter 0

Aim and Outline of the dissertation

The aim of this dissertation is to contribute to the development of a robotic system able to facilitate the implantation of intracortical electrodes for Deep Brain Stimulation (DBS), for symptomatic treatment of neurological disorders. The robotic system should be able to hold and manipulate instrumentation according to the planned trajectories and assist other neurosurgeon needs. More important, the aimed solution should be pragmatic, easy to integrate in a healthcare institution, with low budget acquisition and maintenance costs. We will focus in implementing an initial solution in simulation, to test the control application, the implemented algorithms, the robotic systems selected and the feasibility of the overall project. Due to time and resources limitations real robots implementation will not be covered in the dissertation.

Deep Brain Stimulation is a surgical treatment based in applying controlled electrical pulses directly to specific regions of the basal ganglia, and it has proven to be rather successful in mitigating symptoms of neurological disorders (e.g. Parkinson's disease, epilepsy, dystonia, among others) when other conventional drug therapies or resection neurosurgeries fail [2] [3]. The number of cases assigned to DBS has been steadily increasing, however due to the time demanding and exhausting characteristics of the procedure, they can not be performed at the necessary pace [4]. Medical teams involved in DBS neurosurgery believe that the procedure could be made less physically and cognitively demanding if robots

could take over some tasks, but naturally always under the ultimate control of neurosurgeons. Skull drilling, implantation of multiple electrodes for monitoring and stimulating brain structures are some examples of tasks that could be accomplished by a robotic system. Additionally, the precision, steadiness and tirelessness so characteristic of robotic systems are a major contribute for improving the final outcome of the treatment.

The presented dissertation is organized in three major parts that subdivide in chapters and are presented as follows.

Part I, includes the introductory chapters:

In Chapter 1, we present the impact of the DBS sensitive diseases in current society, some principles about neurostimulation and neuroanatomy. We describe a typical DBS surgery and identify the potential benefits of using a robotic system.

In Chapter 2 is formalized the motivation of the dissertation and provided some general concepts about robotic neurosurgery. It is also elaborated a state of the art review on the robots in neurosurgery that can be potentially adapted to DBS. In the end, we present the current trends and challenges for neurosurgery robotically assisted.

Part II, describes the steps to reach an initial solution:

In Chapter 3 it is presented a search in industrial robotic systems that can be used in our project. It starts by explaining the robotic features and its impact in light of our aimed system. We defined an optimal robot profile and compared the products of the most renowned industrial robots industries to find the best candidate.

Chapter 4 describes the algorithms followed to develop the 6 DOF and 7 DOF geometric and differential kinematics. It is also described some of the selected robots specifications that have direct implication on the developed kinematic equations.

In Chapter 5, we present the work developed in upgrading the *CoopDynSim* robotics simulator, the creation of the operating room environment and the imple-

mentation of the selected robotic serial manipulators. We also present the designed controller application and all the embedded features, taking into account the expected DBS assistive behavior.

Part III, shows some first conclusions about the ongoing work:

Chapter 6, it is presented some results regarding the size and flexibility implications of each robotic system considering the intraoperative environment and the placement of the robot within the operating room. It is also depict the robot's trajectory execution.

Chapter 7 has a brief overview of the developed solution, discusses some of the findings and options taken along the dissertation, and suggests some objectives to be answered in future work.

Chapter 1

Neurological Disorders and Deep Brain Stimulation

This chapter reviews the epidemiology of neurological disorders sensitive to DBS treatment and attends theoretical principles regarding DBS treatment and surgery. It also discusses the utility of a robotic manipulator as an assistive tool within the operating room. This review is based in the following textbooks [5] [6] [7] [2] [8].

1.1 Neurological Disorders Impact in Public Health

Public health has always been a main concern in any complex society and a growing effort has been put towards health promotion and disease prevention. Health promotion is a process that enables people to have a better control over their health and to further enhance it. It aims to raise the global consciousness about health risks, establish healthy policies and provide environments to sustain good health practices. The *World Health Organization* structured a healthy public philosophy as a conjunction of health education, community development and interventions. On the other hand, disease prevention stands for a policy of actively fighting the disease in its various stages:

- Primary prevention, consists in quantifying health parameters to avoid the appearance of a disease;

- Secondary prevention, stands for a correct diagnosis and administration of the standard treatment while taking into account the risks involved;
- Tertiary prevention, includes rehabilitation, palliative care or treatment to restrain complications and miniaturize the effect of the disease.

While recognizing the importance of each strategy, in this dissertation we will focus the DBS treatment which is part of the tertiary prevention.

Neurological disorders are one of the major public health threats, causing motor and cognitive impairment that directly restricts an ordinary life style. Such disabilities imply direct and indirect costs to the society, so it is of great interest to prevent or manage its consequences. Several studies and projects have been commenced to evaluate the impact of neurological diseases, more specifically the epidemiology and burden to the community. Among the other epidemiology studies relative to specific neurological disabilities, it was also regarded a broader essay known as the *Global Burden of Disease* leaded by WHO, the World Bank and the Harvard School of Public Health. The primary objective of this section is to envisage the panorama of neurological diseases and the impact they exert in the society, without acknowledging the causes. Furthermore, from all neurological disorders we will narrow our scope to the ones sensitive to DBS therapy [9] such as Parkinson's Disease, Dystonia, Essential Tremor, Epilepsy, Neuropathic Pain and Psychiatric Disorders.

1.1.1 General Burden

The *Global Burden of Disease* study, presents not only the raw numbers of incidence and prevalence of each disease, but also quantifies the impact it has in peoples lives in terms of DALY, YLD and YLL, (figure 1.1). The neurological disorders consequences go beyond the statistical data, and to assess the real repercussions one needs to consider the impairment caused to each afflicted individual.

Quoting data presented in GBD, neurological disorders contributed to 92 million DALYs in 2005, overcoming Tuberculosis, HIV/AIDS, Malignant neoplasms and Ischaemic heart, Respiratory and Digestive diseases. Epilepsy and Parkinson's

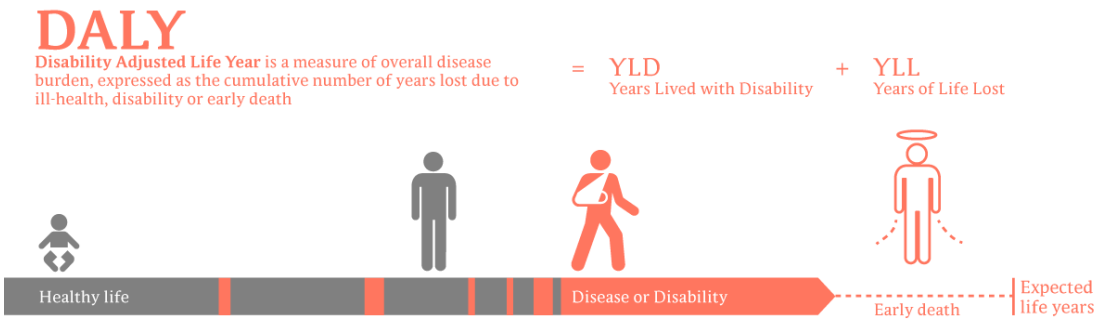


Figure 1.1: Disability adjusted life-year[1].

disease alone constitute almost 9 million DALYs which represents more than 0.6% of the total DALYs for all the diseases covered in the GBD essay. One of the most worrying facts, is the detected growing tendency of both disease DALYs, which is expected to ascend 6% by 2030.

Another interesting information regarding the motto of this dissertation, is to quantity how DBS sensitive neurological disorders affect each stratum of society. GBD presents in their *table 2.5* [5], the impact of each neurological disease in countries of high, upper middle, lower middle and low income categories according to the World Bank. The Epilepsy and Parkinson's disease, the only ones sensitive to DBS treatment among the studied have the highest DALYs per 100 000 population in low income countries. Almost half of the total DALYs caused by these disorders are associated to low and lower middle income categories, or in other words, population with hardly any access to excellence healthcare services, or rather expensive treatments like DBS or involving robotic surgery.

The large impact that these neurological disorders have in today's society further reinforce the idea that efficient solutions like DBS are welcome to brief the problem. Along with the DBS treatment the numbers displayed in the subsections below justify the need of an auxiliary robotic system that would assist the neurosurgical team to achieve better outcomes in less time, with better working

[1]**subsection 1.1.1** Free licensed media from Wikimedia Commons, with the authorship of Planemad `http://commons.wikimedia.org/wiki/File:DALY_disability_affected_life_year_infographic.png`

conditions and therefore allow more DBS procedures to be performed in the same time interval. An affordable robotic equipment would enable more healthcare institutions, even in low income countries, to acquire and exploit all the advantages of an assistive robotic system in DBS surgeries.

We will hereby list the more important diseases responsive to DBS treatment, introducing its definition, enumerating the symptoms associated and providing an overall perspective of its significance in todays society.

1.1.2 Parkinson's Disease

Parkinson's disease is a chronic and progressive neurodegenerative disorder of insidious onset, recognized by the bradykinesia, rest tremor and posture disturbances. It can be later on, associated to other motor and non-motor symptoms like postural instability, falls, freezing gait, speech and swallowing difficulties, among others [5].

It holds the second position among the most common neurodegenerative disorders, next only to Alzheimer's disease, and is expected to cause increasing economical and social burden due to the populations aging. Parkinson's disease causes are largely unknown and furthermore its subtle progression and symptoms are also similar to other motor disorders. Thus a neurologist can at best provide a probable diagnosis, which may later be definitely confirmed *post-mortem*.

Adding to the variability inherent to an epidemiology study from sampling the population, there is yet room for subjective methodologies relative to the diagnosis criteria. To reduce the impact of these inconsistencies, we gathered information from large sample studies. A broad study carried on in 2006, states that the prevalence of PD in industrialized countries is estimated to be 0.3% and reaches 4% in elder population [10]. The same study reports an incidence rate of 8-18 per 100 000 person-years, however this range englobes the population from all ages and the onset of PD is rarely noticed before 50 years old. As a result of being a chronic disease, the prevalence is much higher than its incidence.

The disease's course and outcome varies from patient to patient, and the symptoms which are usually mild and unilateral at the first stages, if left untreated after

several years, can lead to significant motor deterioration with loss of independence. The symptoms limit an ordinary lifestyle, emotionally, socially and economically as it usually implies an early retirement, the stigma of a chronic and incurable condition and an additional burden to families or communities.

1.1.3 Epilepsy

Epilepsy is a chronic neurological disorder prevalent in both genres and contrary to PD, common at all ages. The WHO defined it as *a disorder of the brain characterized by an enduring predisposition to generate epileptic seizures, and by the neurobiological, cognitive, psychological and social consequences of this condition. The definition of epilepsy requires the occurrence of at least one epileptic seizure.* Theoretically any individual with a functioning brain can have a seizure, the factor that triggers it is related to a threshold which can be set by genetic variations or premature birth. Epileptic seizures can manifest in various forms depending on the underling cause and lead to different prognosis, which ultimately condition the treatment approach. The epilepsy can be characterized by the age of the beginning of symptoms. When the seizures begin at childhood they normally remit spontaneously, while when they begin in adolescence they are often lifelong but frequently sensitive to antiepileptic drugs.

Epilepsy affects people of all ages, so the YLDs related to this disease are in average more per individual than PD, and thus the need to measure the different impact it has in society. According to Sander et al. [11], the incidence of Epilepsy ranges 40-70 per 100 000 person year, and 100-190 per 100 000 person year in poorer resource countries. So not only are the people in this countries more prone to develop Epilepsy, they have also less access to quality healthcare. In the study presented by the WHO, it states that the incidence among the children is higher and more variable ranging from 25 to 840 cases per 100 000 population per year. Even knowing from community based studies that 70 to 80% of the people afflicted will achieve remission, the disability and stigma caused to the others makes it one of the most dramatic psychological and social impairments since an early age.

The burden of Epilepsy extends beyond the estimated 50 million peopled affected worldwide, to roughly 200 million family members or friends who are accountable for them and therefore indirectly affected too [5].

1.1.4 Essential Tremor

Essential tremor is known as the most prevalent tremor disorder, and one of the most common neurological disorders. Despite this alarming fact, the etiology and pathophysiology are still not well understood according to neurologists [12]. This movement disorder is a chronic, progressive and degenerative brain disease, distinguishable by a 4-12 Hz kinematic tremor that occurs during volitional motions unlike the tremor latent in PD, which is predominantly basal. Aside from the tremor, ET afflicted patient's often experience ataxic gait and balance problems. Moreover in a psychological level it is usually diagnosed anxiety, depressive symptoms and social phobia [13].

Once again for ET, it is not an easy task to present a definite study that describes the real impact of the disease in terms of prevalence and incidence in population due to the flexibility of diagnose criteria. Quoting Louis, ED. [14], it is estimated that 30 to 50% of supposed cases of ET are misdiagnosed as parkinsonian or other forms of tremor, which means that the epidemiology data may be underestimated. Nonetheless, ET is expected to have a prevalence range from 0.4 to 3.9%, and an incidence rate of 77-326.3 cases per 100 000 person years as stated by recent studies.

1.1.5 Dystonia, Psychiatric and other Neurological disorders

Besides the previously mentioned nearological disorders, also Dystonia [15], Neuropathic pain [16], and Psychiatric disorders [3] like Obsessive Compulsive Disorder [17] and Gilles de la Tourette [18] symptom's can be lessened by means of Deep Brain Stimulation. Despite displaying lower incidence and prevalence rates, they are not in any way easier to bear than the previous ones. Patient's afflicted face

unsurmountable and incapacitating problems that go from the inability to carry on a normal life, to the non-acceptance of disease related symptoms by society, which ultimately results in depression, obsessive-compulsive behaviors, stress and in extreme cases self-injurious behaviors or suicide.

Dystonia is a *syndrome of sustained muscle contractions frequently causing twisting and repetitive movements, or abnormal posture*, which may be classified according to the age of beginning of symptoms, body distribution and cause. It has a prevalence of 2-50 cases per million population year for early onset, that is before 20 years old, and 30 to 7320 per million of population year for the late onset, after 20 years old [15].

Gilles de la Tourette is condition whose symptoms include repeated and quick movements or sounds performed without the control of the patient, that last longer than a year. These motor and vocal disorders are referred as tics. Tourette syndrome is also characterized by the relative young age of its onset that ranges from 2 to 21 years with a mean of 7 years old [18]. It has therefore a tremendous negative impact in any afflicted individual. Despite the expected remission [19] a study by Pappert et al. showed that from a set of cases followed since the onset of the disease, around 90% of the patients still presented tics in adulthood [20].

1.2 Deep Brain Stimulation Principles

Deep brain stimulation is a technique used in functional neurosurgery, which consists in applying controlled electrical pulses directly to deep brain structures through implanted electrodes linked to a neuropacemaker [21] [2]. It has proven successful in mitigating symptoms of neurological disorders (e.g. Parkinson's disease, Epilepsy, Essential Tremor, Dystonia, other Neurological and Psychiatric disorders), where other conventional drug therapies or resection neurosurgeries fail [3] [9]. Despite the previous works on brain stimulation through electric signals, only in the 1990s did Benabid and his colleagues brought together the definitive electrode implantation associated to a neuropacemaker device which introduced the long-term chronic DBS [22]. The growing acceptance and success of this tech-

nique is in large measure due to its non-destructive and reversible characteristics, with unknown side effects to date [23]. Even though the theory of localized electrical stimulation is known for decades, it has only recently been employed in patients as a consequence of recent developments in imaging technology and clinical instrumentation [24].

The major impediment for the adoption of DBS surgical treatment is the required knowledge of both electrophysiology and neural principles with applied electrical signals, which stands as a challenge to already overworked healthcare professionals. As healthcare professionals are more familiarized with pharmacologic concepts, the choice of treatment is often biased to drug therapy. Moreover, the DBS method does not seem to be as attractive as genetics or stem cell studies for treatment of Parkinson or Epilepsy disorders, if we compare the number and impact of papers submitted per year for each solution. Specialists also tend to amplify risks involved in a standard DBS procedure and disregard the drawbacks latent in today's drug based treatments. However the most relevant fact is the fact that DBS symptomatic treatment clinical results far exceed those from stem cell treatment [23].

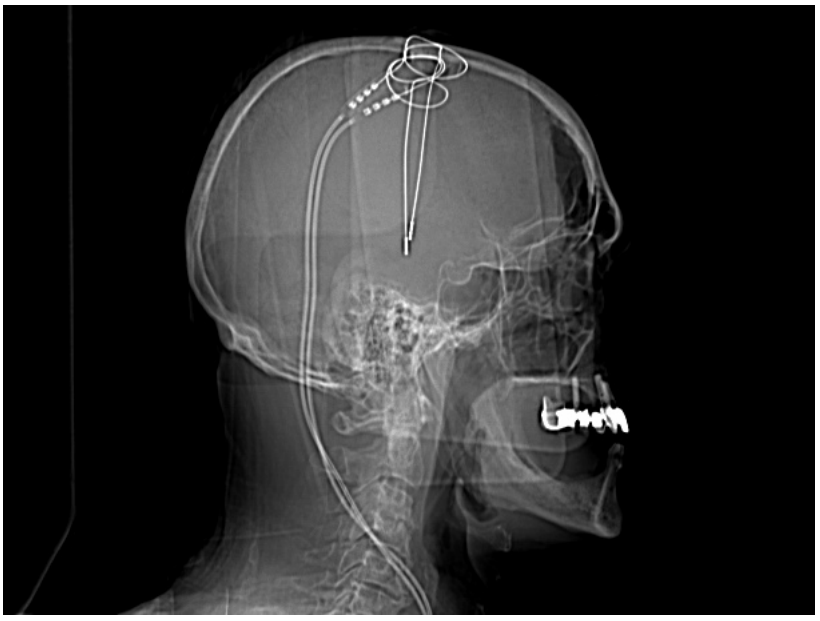


Figure 1.2: Postoperative X-Ray scan after the attended DBS surgery that took place at the Coimbra University Hospitals.

Literature brings some guiding and approaches to obtain the best results from DBS therapy, and addresses topics like number and configuration of electrodes, type of stimulation used in terms of amplitude and shape of the signal, and the type of electrical field generated by electrodes. DBS is focused in symptomatic management, but also provides an unique chance to study brain function and dysfunction theories [23] [9] [2].

One of the main drawbacks of this therapy is the absence of an absolute knowledge about the electrical stimulation mechanisms of action on deep brain structures. The treatment works like a black box in which, the neurosurgeons place stimulation electrodes in planned coordinates to cover a volume around the expected optimal target. Then it is applied controlled electrical inputs and assessed the patient's symptomatic variation in a qualitative scale by neurologists. Even the slightest change in the input parameters, like electrode position or signal properties, has a significant impact in the symptomatic response. Consequently, it is imperative to guarantee the correct positioning of each electrode, carefully select and program the input stimulus to obtain the optimal symptomatic response, which will be in part referred in our work.

The success of DBS therapy is strongly dependent on three factors: *i)* a careful selection of patients, *ii)* the correct placement of the leads into the sensorimotor regions of the target nuclei, and *iii)* the optimal choice of electrical parameters for stimulation. In this subsection we will address the candidate selection step and discuss the major guidelines to choose the electrical parameters. Electrode placement methodologies will be addressed in the *Deep Brain Stimulation Surgery* section (cf. section 1.3).

1.2.1 Patient Selection

Patient selection is the first and one of the most crucial steps of DBS as not all candidates are apt to receive the treatment. The final decision whether to apply the treatment is usually deliberated based on a multidisciplinary group composed of neurologists, neurosurgeons and neuropsychologists [25]. They must always weight

the risks of both the surgery and treatment in light of the potential benefits.

Not all the patients afflicted by DBS sensitive neurological disorders are eligible for this treatment. Therefore, it is particularly important to attend to patient selection also as a mean to quantify the impact that a treatment like DBS can have in society. However and as it will be shown, despite the relative high percentage of people who would benefit from this treatment, a large number is left apart [26].

DBS treatment is currently considered a case of success, fact that lead to a widespread interest and as a result, large and increasing numbers of patients are regularly assigned to DBS surgery. The large number of patients adding to the limited resources to provide this treatment, the limited experienced neurosurgeons and movement disorders neurology teams, dictate the need to define a criteria to restrict the number of patients with access to DBS. The selection method should consider the potential surgery morbidity and mortality, all costs associated to DBS, the time and effort demanded from the patient, caregivers and all the medical team assigned [26].

Neurologists have understood to a certain point the advantages and limitations of DBS and are able to predict different treatment improvements based on the symptoms. DBS tends to be an effective solution for symptoms like tremor, bradykinesia, rigidity, dyskinesia and motor fluctuations. On the other hand, the therapy response is not convincing enough for symptoms such as dysautonomia, cognitive dysfunction, dysphagia, micrographia, hypophonic speech and gait or balance problems [27].

One of the first steps when evaluating the eligibility of a PD patient is to confirm the idiomatic PD diagnosis as it is the more sensitive to DBS than other parkinsonian syndromes. Being this condition confirmed, the neurologist evaluates the patient's feedback to Levodopa[2]. Despite the fact that DBS is often a substitute to conventional pharmacological treatments when they become ineffective, neurologists expect a minimum feedback from drugs as a sign of potential improvement.

[2]**subsection 1.2.1** One of the most common drugs used to treat PD and dopamine-responsive Dystonia.

Neurologists do not recommend DBS to patients above a certain age limit. This decision is backed up by higher risks of intraoperative cardiopulmonary events and to a direct relation between age and incidence of cognitive changes after surgery. Patients who present preoperative cognitive decline tend to get worse postoperatively as concluded from empirical data. However it is hard to specify a rigid threshold for a chronological age from which to deny DBS treatment, as it does not necessarily correlate with the biological age.

The table 1.1 taken from [27] presents some practical and explicit criteria to decide the eligibility of a PD patient for DBS treatment [25].

Table 1.1: Summary of generally accepted criteria of deep brain stimulation candidacy for treatment of Parkinson's disease.

Inclusion Criteria	Exclusion Criteria
Diagnosis of idiopathic PD	Serious surgical comorbidities
Disabling or troubling motor symptoms, including motor fluctuations or dyskinesia, despite optimized pharmacological treatment	Uncontrolled psychiatric illness, including anxiety and mood disorder (BDI[3] > 15)
Robust motor response (other than tremor) to levodopa ($> 30\%$ improvement of UDPRS III score[4])	Dementia (MMSE[5] ≤ 24, MDRS[6] ≤ 130)
Clear understanding of risks and realistic expectations from surgery	Preoperative MRI with extensive white matter changes or severe cerebral atrophy

Also for Epilepsy, the cost and complexity associated to DBS surpasses the anti epileptic drug treatment. Regardless of this fact, brain stimulation is often

[3]**subsection 1.2.1** Multiple choice self-report inventory to assess the severity of depression
[4]**subsection 1.2.1** Scale to assess motor signs in patients with PD
[5]**subsection 1.2.1** Brief questionnaire to screen for cognitive impairment and/or dementia
[6]**subsection 1.2.1** Scale that measures cognitive functioning dementia

used as a suplementar method for patients affected with localized epilepsy and low responsive/intolerable to drugs or that lack the precondition to undergo surgery. One of the disadvantages of DBS is that, patients treated this way have the number of seizures reduced but rarely become seizure-free, unlike in resection surgery. A particular interesting advantage of DBS, is that its toxicity does not overlap the medical therapy one, and until date there are no known side-effects [28].

Dystonia is one of the most difficult to evaluate movement disorders since it is a clinical syndrome instead of a disease, which results from several causes. The patient selection criteria for Dystonia follows the standard preventive and awareness guidelines common to other disorders when entrusted to DBS. Neurologists believe that when the patient is afflicted by focal dystonia, one can be successfully treated with botulinum toxin injections in most cases. However in hemidystonia, segmental or generalized dystonia as a result of the extent of muscles involved, the medical treatment is often inefficient, which leaves stereotactic surgery as the most indicated solution [29].

In 2006, Mink et al. [30] elaborated a viewpoint considering the current outcome expectancy of DBS when treating Tourette syndrome patients, and presented some guidelines to help select the best candidates for the treatment and consequently assure the best results. Tourette syndrome is often associated to neurological or psychiatric comorbid symptoms, that have a direct impact in the disorder expression, in the surgery outcome and in the recovery [30]. Regarding the patient selection problem, the authors defined both inclusion and exclusion criteria, table 1.2.

To conclude the topic of patient selection it is essential for neurologists to elucidate the patient about all the procedure since the preoperative until treatment period, explaining the risks as well as the realistic expectations for the DBS. By submitting to this therapy the patient should understand that it consists in a palliative care and therefore will not have any effect in both curing or delaying the progression of the disease, instead it is only focused in suppressing the symptoms to provide a better quality of life. Patients should be aware of the duration of the treatment and the necessity of tuning the parameters of DBS to reach an

Table 1.2: Summary of generally accepted criteria of deep brain stimulation candidacy for treatment of Tourette syndrome.

Inclusion Criteria	Exclusion Criteria
> 25 years old[7]	Movement disorders resulting from other conditions
Chronic and severe tic disorder, with major functional impairment (YGTSS > 35[8])	Severe medical, neurological or psychiatric disorders that may affect the surgery outcome or the recovery
Failed conventional medical therapy (lacking efficiency or unbearable side-effects)	Sensitive to other non-invasive treatments
Patient fit for surgery, with treated comorbidity conditions	Patient unwilling to undergo surgery, or not totally aware of its implications.

optimal symptomatic response as it is not a straightforward solution. Furthermore, candidates to DBS should not expect a functional improvement greater than the best peak of effect of medication, for most symptoms except tremor.

1.2.2 Basal ganglia anatomy and circuitry

Being the deep cortical structures like the basal ganglia, the aim for electrode placement and stimulation, we want to provide some insight regarding the basic anatomy and circuitry of this area.

Deep inside the brain gray matter is located the basal ganglia which along with the diencephalon and the cerebral cortex constitute the prosencephalon or forebrain. This anatomical region is connected to the cortex and thalamus and consists of four main nuclei: the striatum, the globus pallidus, the subthalamic

[7] **subsection 1.2.1** Stable degree of severity with low chances for remission.

[8] **subsection 1.2.1** Yale Global Tic Severity Scale that goes from 0 to 50, and provides score to measure the frequency and gravity of symptoms .

nucleus and the substantial nigra. In figure 1.3 we present a diagram layout of a brain coronal section with the principal elements that form the basal ganglia subtitled [31]. The basal ganglia system, is according to current standards subdivided into three functional territories comprising sensorimotor, associative and limbic which process motor, cognitive and emotional or motivational information, respectively. For that reason this anatomical region is the main target of DBS electrode implants.

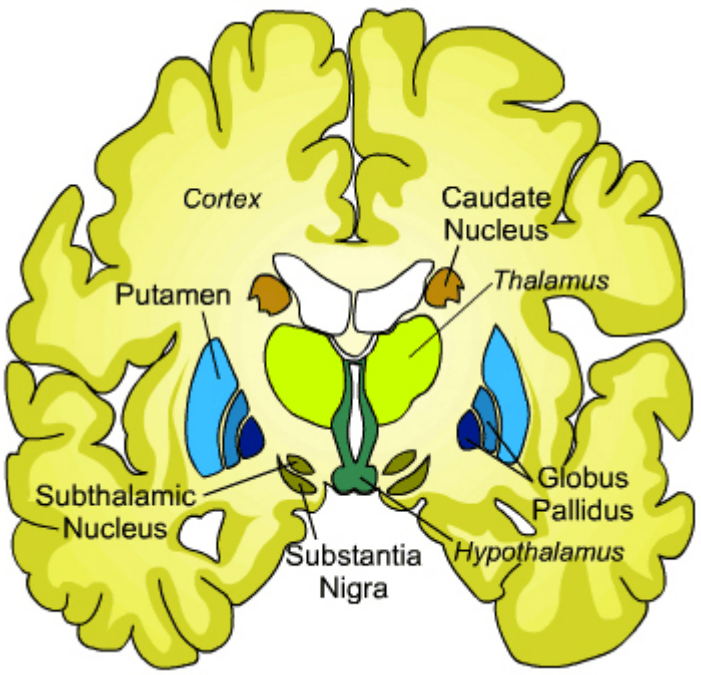


Figure 1.3: Schematic of the basal ganglia anatomy. [31]

The interaction between structures functions as a "center-surround" mechanism to execute desired actions or inhibit unwanted movements. The caudate and putamen are considered the input nuclei of the basal ganglia for sensorimotor information, and together form the striatum which receives excitatory information mostly from the cerebral cortex. The information then flows from the striatum to GPi and SNr, which are thought to be the main output, through anatomical and neurochemical distinct projections. From there it goes to the thalamus, and afterwards to the frontal cortex or dorsolateral prefrontal cortex, depending on whether the desired output is pallidal or nigral respectively. This corticocor-

tical loop passes through the well-known *direct pathway* basal ganglia circuitry. However to correctly modulate its functioning, it is important to refer the *indirect pathway* which involves the GPe and then the STN. After reaching the subthalamic nucleus, the information then proceeds to the GPi and SNr, ending in the same structure as in the *direct pathway*. The striatum, the GP and the STN receive a dopaminergic input from the SNc, which controls and balances the activity of the direct and indirect pathways. Moreover, the STN also receives excitatory inputs from both motor and premotor cortexes and applies a strong excitatory stimuli on its target nuclei. Figure 1.4 summarily explains the relations between nuclei and how the information is handled [8] [32].

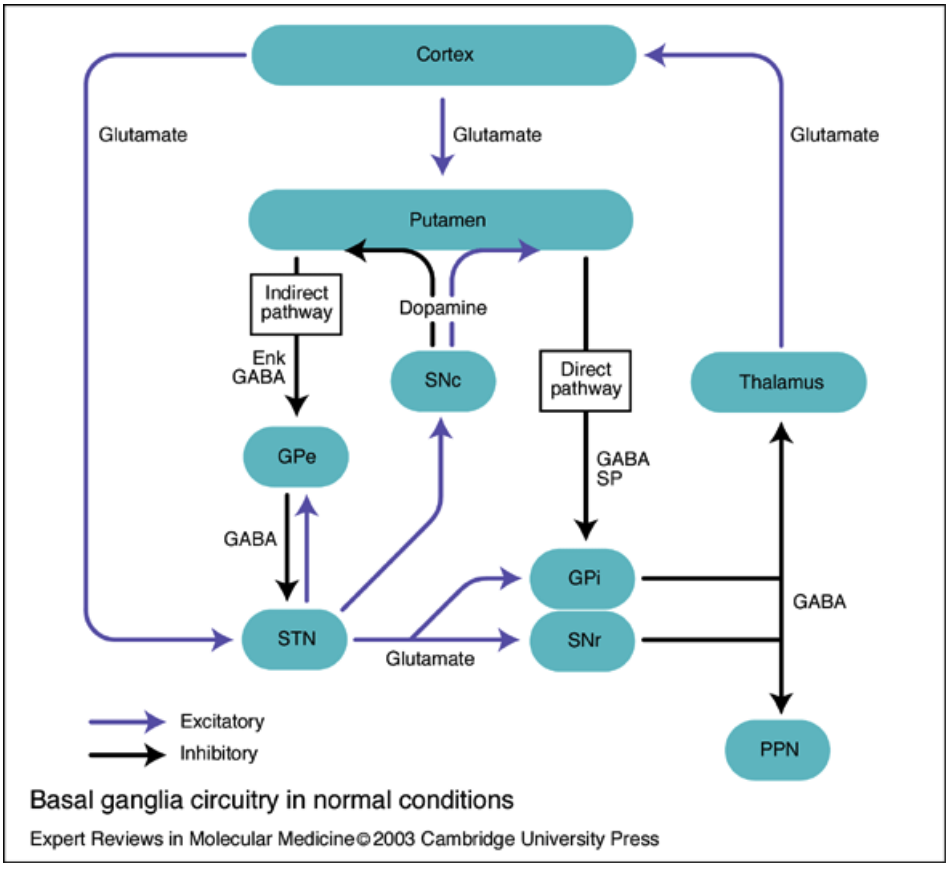


Figure 1.4: Diagram of the basal ganglia pathways [33].

As we can see in figure 1.4, the dopamine has a differential effect on different subpopulations of striatal neurons, playing an excitatory role in the *direct pathway* and inhibitory role in the *indirect pathway*. This control decreases the inhibitory

effect of the system and allows the execution of a desired movement/behaviour. In the Parkinsonian state the lack of dopamine leads to a disinhibition of the GPi and SNr and an increase inhibition of the thalamocortical projections. Such events have as final effect the appearance of a resting tremor consequence of the non-suppression of involuntary movement and bradykinesia or akinesia due to the stronger inhibition felt at the thalamocortical projections, which are common symptoms of PD.

According to a largely cited review by Perlmutter et al. [24], the neurosurgeons/neurologists team select different neurological targets depending on the disorder to be treated. For instance, when treating essential tremor, the VIM stimulation presents an average tremor reduction over 80% in most patients, however stimulating the VIM region to reduce PD symptoms only affects limb tremor having no effect in other symptoms. Still in PD, GPi stimulation can reduce most of the motor disorders as well as brief medication related side-effects but it does not free the patient from drug treatment which is a major drawback. So regarding DBS to treat PD, the most common target is the STN, which can reproduce similar symptom reduction effects while also reducing the patient dependency of dopaminergic medication. However, the final target selection, follows a bit more complex analysis than the one here described.

Since Dystonia manifestations are similar to PD and other tremor disorders and in front of the success of GP and thalamus stereotaxic ablative procedures, it was a matter of time before trying DBS on this very same structures. The result of this trials, defined GP as the primary target to lessen Dystonia symptoms with an improvement ranging from 30 to 50%, having the thalamus as a secondary target. In a relatively recent application of DBS to treat Tourette syndrome, the target selected was the centromedian-parafascicular complex of the thalamus.

1.2.3 Neurostimulation

Being the Neurostimulation the ultimate goal of our project, we decided to briefly describe the fundamental theoretical basis behind the brain stimulation of deep

structures.

The fact that DBS is relatively recent, the lack of complete knowledge about the interaction between electrical pulses and brain activity plus the physiological variability among the patients makes the programming process of DBS variables an extensive and complex equation. To understand the effect of all electrode configurations and stimulation parameters, it would be necessary to conduct a systematic and thorough assessments [23]. The choice of mono over multipolar configurations, the polarity and distribution of microelectrodes, even characteristics of stimulation like amplitude, pulse width and rate of signals need to be adjusted while considering the inherent variability of each patient. An iterative approach would take countless hours to determine the optimal selection of variables.

Despite the variety of combinations of DBS parameters for stimulation that may rise to 12964 possible combinations of pulse width, frequency, and voltage plus 65 electrode configuration combinations, the majority of patients responded well to a known and specific set of configurations [34]. To the other cases however, it is necessary to spend more time studying the patient and calibrating variables to reach an optimal response.

One can brief the problem of DBS programming into two subquestions. The first that indicates how to adjust the properties of the generated electric field, which allows the selective stimulation of different neural elements within the field. This method explores DBS's electrophysiological principles and is based in the influence of each variable such as pulse width, electric current or voltage of the stimulus and electrode configuration. The management of this parameters has a key influence in the improvement of DBS symptomatic control. The second takes into account parameters like size, shape and anatomical positioning of the electric field. It reduces or even eliminates side-effects and/or other possible complications.

Improved DBS efficiency

Dostrivsky et al. [35] believed that high frequency stimulation of GPi, STN and VIM, the most common method used to suppress neurological disorder's symptoms had an inhibitory effect that lead to a decrease of the simulated structure output,

by analogy to the outcome of ablative/resection methods. On the other hand, low frequencies were thought to worsen symptoms and signs of PD, as it would excite the target. However, Dostrivsky explanation left key issues untouched and has recently been refuted due to clear evidence from studies like [36] and [37], which confirm that both low and high frequency DBS stimulation can excite and increase the output of the target structure [38].

Low frequencies have already proven their usefulness when compared to high frequencies, by providing a better symptomatic control over dystonia, speech or gait problems. High frequencies present finer symptomatic control in motor disorders, ET and typical signs of PD. The exact mechanism of how the frequency modulates the neural activity is yet to be uncovered.

In terms of signal modulation, there seems to be clear advantages pointing out to current stimulation. When the signal is voltage modulated, the injected current varies in inverse proportion to the local impedance obeying the Ohm's Law. However, the impedance is difficult to calculate due to the subjective nature of each patient and to the properties of the brain volume around the target. Consequently there exists a risk of the current overcoming the safety limits due to an unforeseen abrupt drop of impedance[9]. If current modulation is chosen, the signal current is controlled and its waveform is preserver [39].

Reduction of Side-effects

As previously referred in subsection 1.2.2, a stimulus can produce different responses depending on the stimulation site. This variations can be ascribed to the singular neuronal circuitry related to each structure, to the variations of neurons shape and properties among each region. The variation of symptomatic response is so marked that even inside the same neural structure, the stimulation can reproduce very distant effects as a result of the non homogenous shape of a nuclei.

There are authors like Montgomery et al. [37], that consider DBS success to be a trade-off between efficient stimulation of desired targets and the ability to avoid

[9] **subsection 1.2.3** Could easily occur as a result of an internal bleeding.

the spreading of electrical current to nearby neural structures. By applying greater intensities of current or stimulating larger areas it is possible to recruit more action potentials in a greater number of neural elements. Aside from the increased efficacy on briefing the symptoms, more unwanted structures are also being stimulated which leads to more side-effects. This fact supports the categorical importance of a precise electrode placement and also helps neurologists/neurosurgeons teams to troubleshoot the current physiological position of the electrode based on the symptoms and side-effects.

If the stimulation side-effects still have a significant impact even upon reaching optimal stimulation coordinates, the medical team may change the combinations of current, voltage, signal shape, area of effect and even the distribution of the electrical field that can be modulated through the use of various contacts of the quadripolar electrode.

Other policies to restrict the activation of undesired structures exploit the principles of how neural elements react to the different characteristics of the stimulus. For instance, axons with a larger diameter have lower thresholds than smaller ones. Additionally, axons have lower thresholds when compared to dendrites, which in their turn are easier to excite than cell bodies. Knowing this beforehand allows the adjustment of the applied stimuli to exclusively affect more sensitive targets like larger diameter axons or axonal terminals.

1.3 Deep Brain Stimulation Surgery

Deep brain stimulation affects only local brain structures and circuitry, instead of the whole body as in medication. However, the amplitude of the stimulation signal decreases with the distance from the electrode, so the correct placement of the lead is crucial for the success of the therapy. After addressing the basic anatomy and circuitry of the Basal Ganglia and unveiling some of the principles behind neurostimulation, we will now talk about the general DBS surgical procedure. As we aim to develop a robotic oriented solution to aid in the intraoperative DBS process it is of key importance to fully comprehend each surgery step. Understanding

how the surgery unfolds and perceiving any pre or intraoperative requirements, environment or procedure constraints is key information to set how the robotic system will operate.

In order to acquire practical insight about the surgery, to grasp when and how can a robotic manipulator be of use, why would it improve both working conditions and the final outcome, we assisted a DBS surgery in the Service of Neurosurgery, Coimbra University Hospitals in Portugal. It will be henceforth described a bilateral DBS surgery conducted in a patient affected by Parkinson's disease. The surgical steps performed in a DBS procedure are independent of the neurological disorder that affects the patient as the only difference is the location of the target to be stimulated[10]. More information about DBS surgical technique can be found in ([9] chapter 1, [40], [24] and [25]).

One can briefly divide this procedure in sequential substeps that include: initial image guided anatomical targeting, skin incision and burr hole drill and attachment of the head reference system, intraoperative microelectrode recording and micro/macroelectrode stimulating and finally macroelectrode quadripolar stimulation upon reaching the optimal coordinates. This quadripolar electrodes are then connected to a neurostimulator device called IPG or implantable pulse generator, which goes under the skin [8].

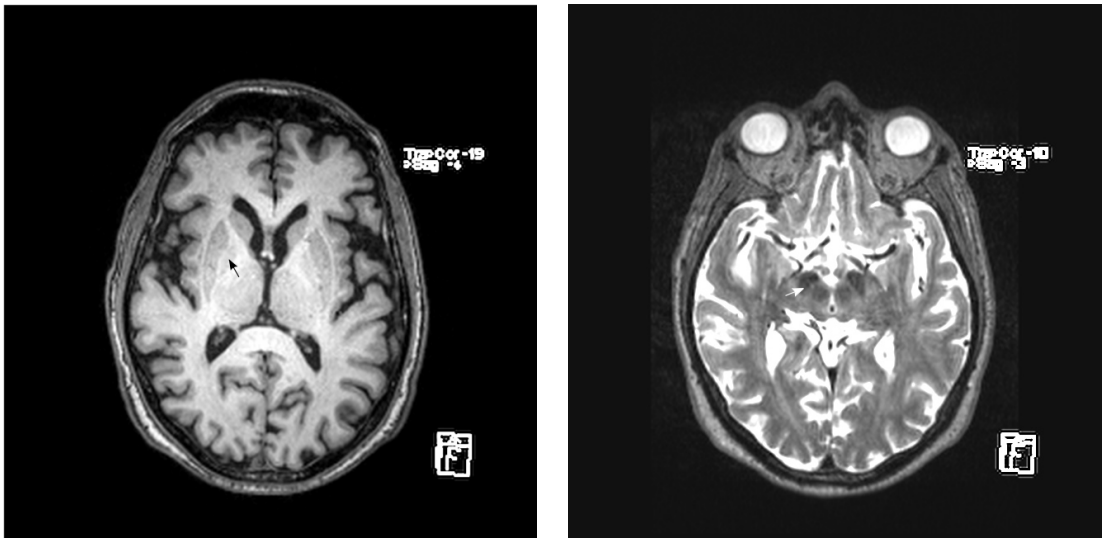
1.3.1 Preoperative Management and Target Selection

After deciding the eligibility of the patient to DBS treatment, the next step is to plan the surgical procedure regarding his symptoms, tolerance to surgery, medical team preferences and available equipment. Furthermore, the medical team must also decide whether to apply unilateral or bilateral stimulation according to the pretended control over the patient's symptoms and medication reduction. Neurosurgeons can either chose to implant bilateral DBS leads and IPGs in the same day; or implant bilateral leads one day and insert both IPGs the day after; or

[10]**section 1.3** The surgical technique described, does not exclude the fact that other surgery services may carry it out differently.

do an unilateral insertion of both devices simultaneously or in separate interventions. Surgeons usually master the use of their institution's medical equipment, and comply with the center operating policies.

About a week previous to the surgical placement of the electrodes, the patient undergoes a MRI scan to determine the targets to stimulate, figure 1.5. Although other exams like CT or ventriculography are also eligible for the same effect, MRI is the most advantageous because CT lacks the contrast and definition when analyzing soft tissues with close densities, and ventriculography may provoke hemorrhage, confusion and headache [41].



(a) T1-weighted preoperative MRI (GPi pointed by black arrow).

(b) T2-weighted preoperative MRI. (STN pointed by white arrow).

Figure 1.5: Preoperative MRI scans previous to the attended surgery at the Coimbra University Hospitals.

In the surgery day, the reference system of the stereotaximeter is attached to the patient's head[11]. To assure the patient safety and prevent infections his head is shaved and prepared with betadine[12]. Then the patient is sedated and the

[11]**subsection 1.3.1** The reference system of the stereotaximeter used in the CHUC neurosurgery service had a ring shape. In order to simplify its notation we will henceforth call it stereotactic ring.

[12]**subsection 1.3.1** Common topical antiseptic used in hospitals to prepare patient's skin and/or used to clean and disinfect surgeon's hands prior to surgery.

pins of the stereotactic ring are inserted with caution to avoid the supraorbital nerve. When placing the ring there must be a special care so that it does not contact with the nose or the occipital bone/neck as it may cause skin erosion. The stereotactic ring should also not block the patient medic line of sight and thus allow the assessment of the ocular movement during the surgery. The pins should firmly secure the ring to keep it from being displaced during each step of the procedure, but not overly tightened, which may cause twists in the structure and affect the system's accuracy.

With the stereotactic ring correctly placed the patient undergoes a CT scan. Computed tomography scans are more suited at this point, as it is compatible with the ferromagnetic material of the ring, and unlike MRI, CT is less affected by distortional artifacts caused by inhomogeneities in the magnetic field. In the end, an imaging planning software merges the high contrast MRI scans where the neurosurgeons segmented the targets, with the spatially precise CT scans and superimposes the information of each exam to achieve a three dimensional representation as similar as possible to the patient's brain. This information is used to aid the neurosurgeon to pick the target and the entry coordinates, so that the final electrode trajectory does not cross any functional or vascular structures.

The final target coordinates to place the leads, can either be achieved using the direct or indirect targeting method. Direct targeting is done by visualizing STN and GPi in imaging scans like T2-weighted MRI or inversion recovery images. Being the STN one of the most common targets for treating PD's symptoms, together with GPi, both regions are generally identified recurring to T2-weighted MRI scans due to the marked contrast between the white matter and both nucleus. Some image processing techniques like windowing can facilitate this task and improve the overall accuracy. Axial and coronal are the most important perspectives when it comes to defining the spatial coordinates of the target. The indirect method involves registering and overlaying brain anatomical atlases according to the patient's scans. From here the target position is inferred or calculated based on established formulas that return the target coordinates based in the distance to well identified landmarks. Indirect targeting while using atlases is both depen-

dent on the accuracy of the given atlas and the topographical correlation between the atlas and the patient's neuroanatomy. The final coordinates may not be the spot where the stimulation will produce the optimal symptomatic relief, but it is still regarded as an initial target. In the surgery we attended, the neurosurgeons team extrapolated the initial coordinates based in known distances from of well identified landmarks.

1.3.2 Intraoperative

A stereotactic frame is mounted in the stereotactic ring, which orients the electrode cannulae along the trajectory provided by the image planning software. Since the skull has approximately a spherical form, the use of the arc stereotaxic frame set with spherical coordinates is advantageous because it allows the selected trajectory to be concentric with the hole rims in both skull outer and inner limits. The settings for the stereotactic frame are directly provided by the imaging planning software.

After setting the stereotactic frame spherical coordinates using mechanical screws, the frame will orient the driver along the defined electrode insertion trajectory. Then, the driver itself is responsible for lowering the electrode's depth. Each input coordinate in the stereotactic frame is confirmed via a millimetric scale engraved between moving components, thus its precision is directly dependent on the human visual precision.

Inside the operating room, while the patient is being prepared, the target coordinates are simulated by means of a phantom device. This phantom has similar sockets to the stereotactic ring fixated on the patient's skull. For each electrode or target coordinate, the x, y and z cartesian components have to be introduced in the phantom and other 3 spherical coordinates this time are set in the frame, so that in the end, the probe's tip and the phantom's tip are coincident, guaranteeing this way that the electrode reaches the target position with a defined orientation, figure 1.6.

The stereotactic frame is then detached from the phantom, attached to the

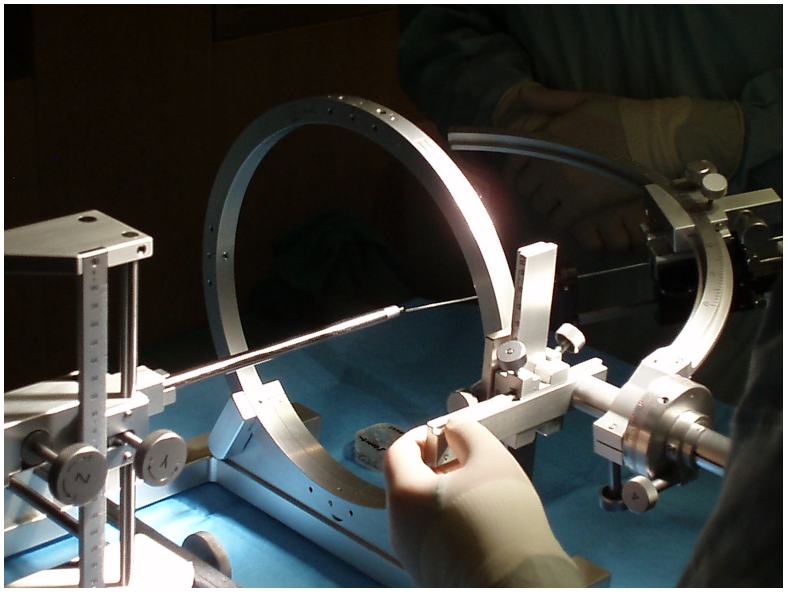


Figure 1.6: Setting phantom coordinates.

stereotactic ring on the patient's head and once again it is set along the desired trajectory. A thin straight metal rod placed in the driver, marks the entry position in the patient's scalp, figure 1.7.

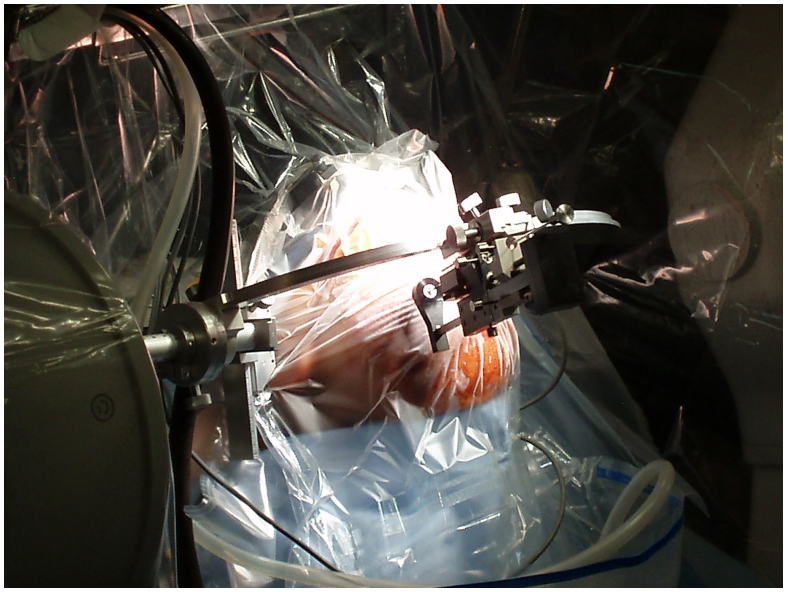


Figure 1.7: Marking the entry position.

Marked the entry position, the stereotactic arc is set aside to make the scalp incision and to drill the hole in the skull to access the brain, figure 1.8.

Then the driver is placed again, along with the frame to guide the electrode

(a) Making the scalp incision. **(b)** Hole drilling in the skull.

Figure 1.8: Surgical steps.

cannulae collinear to the defined trajectory. Firstly, a set of microelectrodes responsible for recording cortical electrical signals are positioned 15mm above the target coordinates defined at the image planning software, along the trajectory and the patient's sedation lowered. The medical team used 5 electrodes cross positioned, set apart 1mm from each other to cover a wide area and thus increase the probability to find the optimal stimulation site, figure within the nucleus (fig. 1.9). The electrodes are lowered millimeter by millimeter in an iterative process until reaching 5mm from the target. Upon reaching this point the electrodes advance now half a millimeter between iterations, and by each step the signal is recorded. At each iteration the neurophysiological readings are saved, compared and analyzed to assess the most proximal locations to the sensorimotor region, or in other words, the coordinates where the electrodes retrieve an electrophysiological record closer to the expected.

At the coordinates where the electrodes recorded the best signals, these recording microelectrodes are replaced by stimulation macro/microelectrodes. For each of this sub-solutions, the electrical current and depth of leads are changed individually in iterative steps. At each step the neurologist team qualitatively evaluates the patient's symptoms as responses to the stimulus applied, and once again they save each in order to find the optimal electrode trajectory, depth and current, figure 1.10.

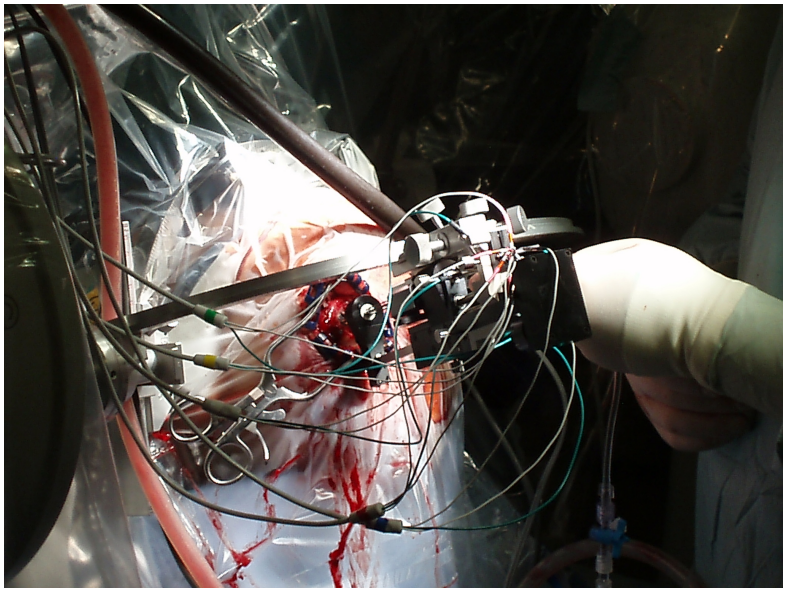


Figure 1.9: Electrode placement, signal registration and stimulation.

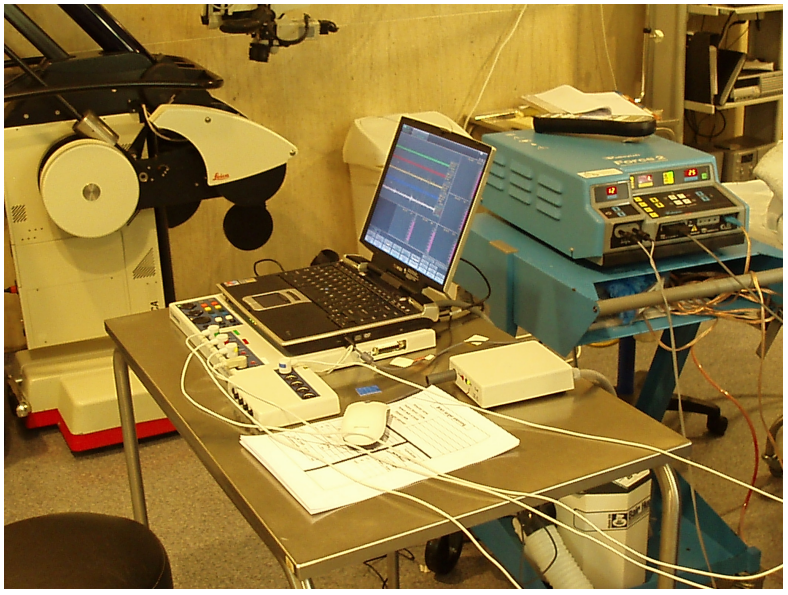


Figure 1.10: Calibration of stimulation parameters.

Depending in which nuclei we are stimulating there are different current limits and levels of responsiveness, in other words, the GPi in comparison to the STN needs higher current amplitudes to achieve the required therapeutic effect. This becomes an important fact because if the patient complains about side-effects when stimulation parameters should not provoke any, it may indicate a bad electrode positioning. Another important fact to be aware of when introducing multiple elec-

trodes is that it may cause edemas or bleedings which will change the interstitial impedance allow the signal to spread to other structures and cause undesirable effects. This kind of problem is one of the reasons that leads some neurosurgical teams to avoid using recording microelectrodes and implant the stimulation electrode alone [42].

Upon finding the ideal stimulation signal properties and position, the electrodes are replaced by a definitive quadripolar macroelectrode, that should later on be connected to an Implanted Pulse Generator (IPG). If a bilateral brain stimulation is needed, all intraoperative process must be repeated for the other side [13].

1.3.3 Which tasks can be accomplished by a robotic manipulator?

After doing some research about robotic systems oriented to stereotactic keyhole surgery, more specifically DBS surgery, the information found almost exclusively describes the features of the system and displays qualitative data about its efficiency. Notwithstanding the factual importance of such details, which shall be discussed in chapter 2, there is quite scarce information about the practical and clinical advantages of including such technology within the operating room. How does it improve the work conditions for neurosurgeons, for neurologists and to other staff? What tasks could be achieved by the robotic system? What are the benefits for the patient?

As stated previously, a typical DBS surgery lasts several hours through which the surgical team must remain utterly focused while performing each step to avoid any failure. Due to its characteristics, robotic systems could provide a substantial contribute in consistency, steadiness and precision and therefore improve at the same time, the working conditions for neurosurgeons and the surgical outcome for the patient [43]. After watching DBS surgery and upon a meeting between neurosurgeons and robotic experts, we concluded that a simple robotic system could upgrade the standard procedure in various aspects:

[13] **subsection 1.3.2** The procedure here described lasted 18 hours.

1. Enable coordinates and trajectory information retrieved from the image planning software to be managed by a communication protocol that would link this program to a robotic controller software. The trajectories information would be handled through software instead of having the medical team manually register and insert it both in stereotactic frame and the phantom screws.

2. Avoid the slow process of, mounting and setting the phantom, frame and driver coordinates to test the position and trajectory generated by the image planning software. Dismount the frame, attach it to the patient stereotactic ring, input coordinates again (just to mark the entry position); disassemble, make incision and drill burr hole, and assemble again to finally introduce each set of electrodes. This procedure repeated for each target.

   A robotic manipulator would position itself according to the desired trajectory in a few seconds and would also open the possibility for frameless stereotactic procedures with guaranteed precision.

3. Avoid stereotactic frame and driver mechanical slacks or loose parts, which may require preoperative calibration. In figure 1.6, it is possible to see that after setting each coordinate for both the phantom and the frame, there is a slight mismatch in their tips meaning that when placing the electrodes in the patient's brain, they will not reach the target location but a nearby one, which may lead to decrease in the treatment efficiency and increase of side-effects.

4. Enable robotic manipulator to handle multiple end-effectors and surgical instrumentation to execute consistent skull drilling based on force feedback, swift positioning and driving of various sets of electrodes with improved precision. The manipulator could swiftly and precisely constrain drilling and insertion tasks to a predefined trajectory, instead of executing them based in a marked entry position.

5. Enable medical teams to easily take control over the task of advancing the depth of electrodes while evaluating the patient's symptoms by simple inter-

acting with the robot controller interface, therefore aiding neurosurgeons on that task.

6. Enable an online monitoring of the instrumentation tips absolute coordinates based on their physical dimensions and on the manipulator position relative to the base referential, using geometric direct kinematics. However, this feature can only be used if the instrumentation is moved along the trajectory by the manipulator.

7. Providing assistance to younger and less experienced neurosurgeons as an assistive and training platform and enabling senior highly experienced neurosurgeons, who might lost some dexterity, to continue performing surgeries later in their careers. [44].

8. Reduce the risk of data loss or human errors.

Cooperation between neurosurgeons and robots opens a new horizon of possible improvements to quality healthcare and is therefore recognized as the final step in the transition of surgery from the Industrial to Information Age [45].

Chapter 2

Robotic Systems for Deep Brain Stimulation Neurosurgery

In this chapter we will briefly talk about neurosurgical robots, point out their advantages and current problems when compared to human standard performances, with a focus in DBS neurosurgery. We will also review the available robotic systems oriented or possibly adaptable to DBS procedures, designating the category they fall in and their main features.

2.1 Motivation

As mentioned in the last chapter, neurological disorders have a tremendous impact in today's society, and burden millions of patients, families and caretakers. According to patient selection rules, not all patients affected by the previous enumerated neurological disorders are eligible for DBS, but a significant part is [4] [27]. On the other hand, DBS has proven to be a top notch treatment that despite having no effect in disease progression, it shows very good results as a palliative care by reducing the symptoms and allowing the patient and family to carry on a normal life. Alongside the growing awareness for such procedure, also the number of actual prescribed DBS surgeries is increasing [46]. Furthermore, DBS neurosurgeries tend to be rather long procedures as stimulation parameters are yet to be programmed

and the duration can drag even further if the surgery is bilateral. Consequently, it is a very demanding procedure for the medical team involved, both physically and psychologically.

Having presented the advantages and difficulties of DBS therapy, how would it benefit from including robotic technology into the procedure? A successful outcome of DBS is largely dependent of an accurate localization of target and a precise guiding of electrodes to it [47]. General surgery and more specifically neurosurgery has been evolving towards increasing minimalist techniques, consequently achieving lower recovery times and improved patient outcome. The continuous search for a higher quality surgical practice, magnification of the operating field and tool miniaturization brought healthcare professionals to the verge of natural manual dexterity and spatial orientation [48]. Robots on the other hand can easily surpass humans in precision, procedural consistency and by introducing features like tremor filtered tool handling and movement amplification [49]. A robotic manipulator could takeover some menial and repetitive tasks, (but always under supervision of the surgeon) thus reducing the physical effort required.

Keeping in mind that DBS is a relatively recent therapy, and that robotic neurosurgery has not long ago released the first products into the market, there are yet a lot to be made in the Robotic DBS field. After searching on this topic we found scattered systems and ideas but not a review that compiled information exclusively about robotic assisted DBS. Thus, one of the goals of this chapter is to provide a complete review about what can be done, what is expected, current challenges and future trends.

2.2 Robotic Systems for Neurosurgery

The concept of robotic manipulator has been defined as "A reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks" by the Robot Institute of America in 1979. Robots can be divided in classes and types depending on its function, shape or control mode, and cover a

wide variety of tasks from human replacement in hazardous missions, repetitive and stressful jobs and even menial tasks. Notwithstanding its shape or function a robot architecture is generally divided into [50]: (i) a mechanical subsystem composed of both rigid and/or deformable bodies, (ii) a sensing system, (iii) an actuation subsystem, (iv) a controller and (v) an information-processing subsystem [51].

For DBS surgery, the most common robotic body among the commercially available is the serial arm because it is better fitted for tool handling and manipulation. In theory, the robotic solution should be able to reach a desired three dimensional position in space with a specified orientation. Six degrees of freedom (DOF) manipulators are rather recurrent as they offer flexibility in both positioning and orientation, while keeping kinematic equations simple. Increasing the number of joints would result in kinematic redundancy, which would increase the number of possible configurations to reach the same position and orientation and so the final arm pose can be decided upon user defined conditions ([52], chapter 1). As neither execution time nor large forces are pre-requisites in neurosurgery, robot kinematics can be conditioned to function with lower joint speed and forces. Consequently increasing the stability and reducing the risk of failure since the manipulator movements are less demanding for actuators [53].

The robotic system can be decomposed in: **sensors** that provide the controller with feedback of the environment and itself, from which it is possible to extrapolate the robot spatial configuration through direct kinematics; **actuators** that supply power to joints, responsible for moving each link to a final arm posture and end-effector position; and finally in a **controller** and **information-processing** subsystems which are accountable for controlling the inputs of each actuator, based both in the controller algorithm and the received data from sensors [53] [47].

The advantages and limitations of robots over humans have been addressed for a long time and despite the specificity of DBS surgery, the same arguments can be used, (see table 2.1) [54] [53]. The healthcare institution must always weight the benefits of a robotic system over the existing solution, and should also consider the financial impact and the need to adapt or change methodologies. The problem

Table 2.1: Advantages and disadvantages of human and robots capabilities.

	Surgeons	**Robots**
Strengths	Flexible, adaptable to tasks	Stable, untiring with repeatability
	Strong hand-eye coordination	Good geometry accuracy
	Dexterity above millimeter scale	Motion scaling, tremor filtering
	Judgement experience	Use diverse types of sensors in control
	Powerfull sensor capability	Manage multiple, simultaneous tasks
	Handle qualitative information	Work in dangerous environments
Limitations	Tremor, Fatigue	Limited dexterity, hand-eye coordination
	Ineffective at sub-millimiter scale	Large, Cumbersome and Expensive
	Variability in skill, age, state of mind	Unable to process qualitative information
	Difficulties in handling quantitative data	Technology still under development
	Imprecision	Not versatile

with most of the robotic neurosurgical systems is the fact that they are difficult to include in the *modus operandi*, technically demanding to use and still exceedingly expensive. On the other hand, the precision, steadiness and tirelessness so characteristic of robotic systems are a major contribute for improving the final outcome of the treatment [55].

Presently, robots are recognized as a cooperative tool that functions under the supervision of a surgeon instead of a replacement. Consequently, the first step towards improving surgical robot systems is to establish a base architecture, essentially a well structured universal core from which any company could assemble an oriented solution for the specific problem of DBS surgery. Keeping this in mind, we dared to point out some guidelines about general features of a robotic system for DBS surgery, relative to the desirable upgrades from the standard procedure previously referred.

The first desired upgrade consists in developing an interface to connect the robotic controller software to the imaging planning software used by the medical, so they can share the information about the planned trajectories. A simple improvement like this, would relieve the medical team from personally managing and interpreting the relationship between the coordinates extracted from imaging data and the chosen settings for the guiding hardware. Whether to devise an imaging software to be used in parallel to the robotic controller system, or develop an interface able to communicate with both existing planning software and manipulator, is a choice of the project leader.

The robotic behavior for DBS surgery should be semi-active or passive, so the robot can be manipulated online or moved accordingly to preoperative directives. Having the manipulator perform independent actions is not a requirement nor a desired feature in this case, since all motions are relatively simple and can be previously coordinated. Two of the most desired progresses are the ability to passively hold and manipulate instrumentation in defined positions and orientations, for electrode guiding or motion restriction in hole drilling. The knowledge required of direct and inverse kinematics both geometric and differential for serial manipulators, are already fully understood and studied. Therefore the biggest challenge remains in conceiving a system able to adapt to the institution methodologies and standards rather than understanding the mathematical equations and principles involved.

As any other hardware within the operating room, also the robot must obey antiseptic policies. The most common and recommended procedure defends that all the parts of the robot should be covered with sterile drapes or pre sterilized bags except for the end-effector that should be reusable and sterile [56]. It is desired for the manipulator to handle different instrumentation, the developer should once again decide between using an adaptable end-effector to hold each different tools and consequently avoid the need for its replacement when changing tasks, or use simple end-effectors for each task [57]. When projecting these components, one must bear in mind that the solution should be able to undergo sterilization processes like autoclave or be cheap enough to be disposable.

It is recommended that the robotic system is mobile and able to be easily moved around the operating room, because normally the same environment is used for other neurological surgeries as well. A portable structure implies that both size and weight of the manipulator and controller should be restricted. The assigned tasks in DBS surgery are not demanding in terms of weight nor they require a large workspace and so it is rather easy to find a light and compact manipulator to fulfill that duty, refer to chapter 3. Another concern regarding the portability of the robotic system is its capacity to attach its structure to the surgical table, to guaranteed that in all moments the transformation between the manipulator referential and the referential used for surgery is fixed. The surgery base referential is recurrently situated at the center of the reference system of the stereotaximeter, so the displacement and orientation between itself and the robot base needs to be acquired previous to the surgery. Once again this problem can be tackled from different angles, either using optical registration to match the operating field to preoperative exams, using fiducial markers or direct kinematics to recognize key points in the stereotactic reference system and then extrapolate the surgery referential.

The simplicity of the required movements, demands an equally pragmatic and intuitive user interface. Besides positioning along the desired trajectory and possibly execute collinear movements, the robot may also be manipulated so that its body does not interfere with the neurosurgeon's workspace. Adding an intuitive control over the arm like floating mode or moving it with a six DOF controller, would only make it more easily accepted in the medical community.

Safety is a paramount concern when developing a robotic system, and should be addressed in every phase of the system conception because a small failure can lead to drastic consequences or place bad labels on the product. There is a great concern involved in assuring patients safety from undesired movements of the manipulator, and same can be said for the medical team that shares the workspace. Fei [58] splits the universe of possible system errors into four categories: pure hardware, pure software, hardware triggered by software and software triggered by hardware. Then he proposes a systematic method to analyze and evaluate safety issues based

in seven core steps:

1. *Definitions and requirements*, of the surgical robot functionalities and working environment constraints;
2. *Hazard identification*, of robot actions relatively to patients, other persons or surroundings;
3. *Safety insurance control*, in which implementation guidelines are specified and established a monitoring mechanism to eliminate or reduce hazard;
4. *Safety critical limits*, sets each subsystem safety parameters thresholds;
5. *Monitoring and control*, is an assessment of the Safety insurance control success;
6. *Verification and validation*, of the final product compared to the requirements;
7. *System log and documentation*, to record system status and interface instructions.

Also, if there is any malfunctioning or the manipulator just stops working it must be assured at all moments a retract mechanism that allows a safe removal of the instrumentation from the end-effector and the resumption of a standard stereotactic surgery [59]. The predictability and simpleness of movements should not be an excuse for neglecting safety protocols.

2.3 State of the Art Robotic Systems

Since the first report of a robotic neurosurgical system in 1985, a wide range of neurosurgical solutions have been brought to stage [60]. In this section, we will focus on the most relevant robotic devices for DBS surgery and their key features. We chose to present not only the systems built specifically oriented for DBS surgery, but also systems developed for general neurosurgery which can be adapted to it, (see table 2.2).

Table 2.2: Main neurosurgical robotic projects able to perform DBS and principal features (adapted from [1]).

	Project	Phase	Category	Institution	Main features
ORIENTED	Neurobot	Experimental set-up	Passive, automated	Imperial College of Science, Technology and Medicine, London	4 DOF Rigid platform for tool holding, moving endoscope along defined trajectory and restrain instrumentation workspace
	NeuroMate	Commercial use	Passive, automated	IMMI / ISS / Schaerer Mayfield NeuroMate Sarl; Lyon, France	5 DOF arm image-guided, computer-controlled robot for stereotactic procedures
	Pathfinder	Commercial use	Active, automated	Prosurgics Ltd.; High Wycombe, UK	6 DOF manipulator for neurosurgical procedures using fiducials to mark the surgery field
	Robocast	Experimental set-up	Active, automated	Neuroengineering and medical robotics Laboratory, Politecnico di Milano; Milan, Italy	Parallel, Serial and Linear multi-robotic 13 DOF system for precise probe alignment for keyhole neurosurgery procedures.
	Rosa	Commercial use	Semi-active, automated	MedTech SAS; Montpellier, France	6 DOF manipulator for frameless stereotactic robot-guided deep electrode placement.
ADAPTABLE	Evolution1	Commercial use	Semi-active, automated	U.R.S. Universal Robot Systems; Schwerin Germany	4 DOF hexapod based on parallel actuator configuration for drilling applications and tool handling.
	Minerva	Experimental use	Active, automated	Laboratory of Microengineering, Swiss Federal Institute of Technology	5 DOF for stereotactic instrument guiding within CT scanner (discontinued)
	NeuroArm	Experimental set-up	Semi-active, tele-operated	University of Calgary; Canada	MRI-compatible ambidextrous robot.

2.3.1 NeuRobot

The robotic surgical simulator NeuRobot[1] born from the European Community funded project ROBOSCOPE to provide a joint solution for common problems in Neurosurgery. The system consists of a robotic arm and a simulator image-guided system, ROBO-SIM. First the patient's physiology is captured as a 3D MR image data-set, which is used by the surgeon to identify regions of interest like brain lesions, ventricular system or vascular structures. The planning trajectory is then calculated based in this preoperative information [64] [65].

The NeuRobot has 4 active DOF to manipulate the instrumentation around the entry point in the burr hole, and 3 passive axes relative to each spatial coordinate. The robot needs to be manually positioned in a desired XY location inside the operating room while the patient in the surgery table is driven up or down according to the robot's workspace. Then the probe orientation is controlled by Yaw, Pitch, Endoscope rotation and depth DOF. Although the control can be configured to operate autonomously, it would arise several concerns such as "who is in-charge" of the surgery and whether the responsibility should befall in the surgeon or the robot manufacturer [64].

The final product comprises a control mechanism developed from a flight-simulator experience by Fokker control systems, enhancing precise motion and force-control using low force inputs [65]. A special attention was given to safety issues, and thus one of the embedded features is the active constrain of movement outside the safe operating region, which are preoperatively defined based in MRI segmentation data. To solve the tissue deformation problem during the procedure, the probe position is dynamically tracked in real-time with ultrasound imaging, which despite having low spatial resolution has one of the best temporal

[1] **subsection 2.3.1** Do not misunderstand by other systems called NeuRobot [61] [62] that is a telecontroled micromanipulator system with a master-slave control hierarchy to perform minimally invasive procedures using an endoscope and three robotic arms. This system is not relevant for DBS surgery, as the only manipulation required in DBS procedures is done outside the surgical field along a predefined trajectory. There is also another system also called NeuroBot, which is used in skull-based surgeries [63].

resolutions [64].

A stereotactic frame is used to register the external world coordinates to the patient's skull and thus relate the intraoperative manipulator location to the patient's preoperative data. 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 electrodes for deep brain stimulation. This project still needs to attain safety verification according to the legislation of medical robots. The need to adapt the operating room to the robot's workspace, due to its limited flexibility can be considered as a disadvantage.

2.3.2 NeuroMate

The NeuroMate robotic system, which is FDA approved and now commercially available at Renishaw company, is an image-guided and robotic assistive system for stereotactic procedures in neurosurgery, (figure 2.1). It includes a kinematic positioning software, as well as a 5 DOF arm that achieves an accuracy of 0.7mm and a precision of 0.15mm, guaranteeing payload stability up to 7kg. Additionally NeuroMate has an embedded state of the art planning and visualization system able to work with computer tomography, magnetic resonance and angiographic images. Furthermore, it has been designed with the possibility to either use conventional stereotactic localizer frames or an exclusive frameless method based in an ultrasound registration system to localize the robot's position relative to the patient's skull. Being developed strictly towards neurosurgery, the NeuroMate possesses some singular features that distinguish it from industrial robots, like low speed, redundancy and safety devices [66] [67].

Li et al. [66] considered NeuroMate to be an extent to human capabilities, as it reduces human errors and saves time in long term surgeries or biopsies that target multiple structures, in which the human tiring and loss of attention might affect the final outcome. In [66] the authors present the result of a study regarding the

[2]**subsection 2.3.2** Available: `http://www.futura-sciences.com/uploads/RTEmagicP_Renishaw_neuromate_surgical_robot_txdam20315_07b6d2.jpg`. (Accessed: 04-Mar-2012)

Figure 2.1: NeuroMate[2].

accuracy of different localization systems:

1. Standard procedure;
2. Infrared tracking system using the frame for fiducial registration;
3. Frame for fiducial registration;
4. Infrared tracking system using screw markers for registration.

In the end, the last method achieved better results, but a robotic frame based configuration also presented better precision than the standard procedure[3]. The frameless approach scored the worst precision among the studied methods, but still managed to attain an average error below $2mm$ for target tracking in three dimensions [66].

The NeuroMate works as a passive assistant for holding, supporting and stabilizing tools controlled by the surgeon, it also increases surgical safety, improves the efficiency and cost savings to the institution. This stereotactic oriented robotic sys-

[3]**subsection 2.3.2** check Table 2 from, *The application accuracy of the NeuroMate robot–A quantitative comparison with frameless and frame-based surgical localization systems.*

tem uses its own imaging software for registration and trajectory planning, which can also be an obstacle since it is difficult to adapt to other implemented imaging systems in the healthcare institution. Furthermore, its static base can also discourage its acquisition because it permanently restrains the space within the operating room.

2.3.3 Pathfinder

The Pathfinder system (figure 2.2) developed by Prosurgics Ltd. is a robot built for neurosurgical procedures as a response to the miniaturizing of instrumentation and the increase of required accuracy that will soon transcend even the most skilled surgeon capabilities. A 6 DOF robotic arm is installed on a mobile and stable platform that allows it to be easily moved around the operating room and firmly fixed to the patient head clamp during the surgery.

Figure 2.2: Prosurgics, Pathfinder[4].

One of the remarkable improvements in Pathfinder is the introduction of reflectors attached to the patient's head used as fiducial markers to identify the

[4]**subsection 2.3.3** Available: `http://www.designworldonline.com/uploads/ImageGallery/prosurgics%20%20robot.jpg`. (Accessed: 08-Mar-2012)

surgical field. The markers are tracked by a camera system integrated at the end effector, rather than other preoperative image-guiding. The neurosurgeon can either choose to bound them to the scalp or rigidly attach them to the skull, while keeping them uniformly spaced around the head and from each other to facilitate the registration process. These markers consist of a black titanium sphere coated in a reflective material to be easily seen in CT scans and by the robotic camera, respectively [68] [69].

The Computed Tomography exam is used to pinpoint the markers positions relative to the surgical volume, while the MRI image set is required to segment the target brain structures. The CT and MRI datasets are then matched to overlay both target and fiducial markers locations and after by the Mayfield Aciss II^{TM} planning software to calculate the probe's trajectory, which may ultimately be edited by the surgeon. The robot is then positioned sideward from the patient position to provide the maximum DOF without getting in the way of the surgeon when it is not being used. Several integrity tests are then performed to check whether the robot is correctly connected to the controller workstation, to confirm the surgical plans or to assess the correlation between preoperative and live information. Using external fiducial markers allows the system to iteratively track its position relatively to the patient, thus solving one of the biggest issues with preoperative image guided robots, and relieving the need for intraoperative online image scans [68].

It achieves a registration accuracy comparable or superior to a stereotactic frame without the need for it, with all the advantages of a frameless system and permits moving the robot without the need for rescanning or replanning [70]. Furthermore, this tracking system allows for a consistent precision and repeatability below $0.5mm$ in all the surgical volume even in the deep targets, unlike other frameless methods [69]. The most common problem with the Pathfinder approach is the possible skin movements between the preoperative scans and during the procedure and registration failures caused by misidentification of the markers due to abnormal lighting conditions [68].

2.3.4 Robocast

The Robocast acronym for Robot and Sensor integration for Computer Assisted Surgery and Therapy project (FP7 ICT-2007-215190) aims to create a modular system that integrates image guided navigation and robotic devices for keyhole surgery. Therefore the project developers pictured 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 [71] [72].

Robocast architecture consists of optical sensors, an electromagnetic tracking system, ultrasound and three robotic actuators with haptic devices. There is a serial arm called *gross-positioner* with 6 DOF, a *fine-positioner* parallel robot with also 6 DOF to further improve accuracy and a *linear piezo-actuator* to grant a linear insertion of the electrode or biopsy probes. The optical sensors are used to register the intraoperative environment according to the preoperative plan. Probe's tip position is constantly tracked with the electromagnetic system, while the ultrasound device is used for updating the preoperative plan during the procedure. A 3 DOF haptic feedback actuator is used to move the probe inward [73].

The software part can be divided in six subsystems: preoperative planning, human computer interface, sensor manager, high level controller, haptic controller and safety check. The preoperative planning GUI autonomously calculates the lower risk optimal entry point and trajectory, after the surgeon selection of both the target and entry area [72]. 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 it to the system control center. The high level controller receives the information from the preoperative planning and the sensor manager subsystems and iteratively calculates the gross positioner and fine positioner kinematics. The haptic controller interfaces the linear actuator robot with the haptic device, transmitting a force-feedback reaction to surgeon for moving the probe. Finally, the safety check module runs regular state verifications in each subsystem and if it catches any failure it immediately stops the probe

movement [71].

As the Robocast is still at the development stage, we can only speculate about possible drawbacks. Having to control 3 coupled different actuators with several DOF to move the end-effector requires a consistent communication protocol and a complex controlling software which may be more vulnerable to failure. On the other side and by all the described features, ROBOCAST project is expected to have a positive impact both in this field of neurosurgical robotics as well as in the medical community.

2.3.5 Rosa

Rosa robot developed and commercially available by Medtech is the latest generation of neurosurgical computer controlled robot indicated for use as a stereotactic instrument, (figure 2.3). Rosa system comprises a mechatronic part consisting of a 6 DOF serial manipulator and a control part formed by neurosurgical oriented navigation and registration software [74].

The robot is intended to be used within the operating room for spatial positioning and orientation, precise targeting and dexterous handling of minimally invasive instrumentation. These parameters are calculated based in preoperative imaging data of the patient and the intraoperative information of the robot position relative to the physiological target. Spatial registration of the patient's head during the operation can either be done using fiducial markers, for which the surgeon manually moves the tip of the arm and the probe pinpoints the location of each marker or using a laser telemeter to digitally register anatomical landmarks [75] [74].

Medtech conducted a study to evaluate the in vitro accuracy in 45 measurements for each technique using a phantom head with 9 targets. In the end it was concluded that the system localization using fiducial markers and optimal registration achieved both a mean accuracy below $2mm$. The robot is immobilized relative to the patient during the procedure, however the instruments can not be

[5] **subsection 2.3.5** Available: `http://www.medtechsurgical.com/var/ezwebin_site/storage/images/mediatheque-galerie/galerie-photos/robot-rosa/photo-2/720-2-fre-FR/Photo-2_tof.jpg`. (Accessed: 04-Mar-2012)

Figure 2.3: Rosa Robot[5].

adjusted intraoperatively and thus must be calibrated from factory [75].

The Rosa robotic solution combines the planning and registration functionalities of the StealhStation and VectorVision systems and the mechanical guiding and immobilization of Neuromate. The unavailability of technical information and the lack of scientific papers about this system difficults its review.

2.3.6 Evolution 1

Evolution 1 is a robotic system developed by Universal Robotics Systems especially designed for neurosurgical and endoscopic applications at a micro scale for brain and spine surgeries. It falls out of the pack as a 4 DOF hexapod or Stewart-platform based on a parallel actuator that combines high accuracy with great payload capacity. Its 6 mechanical parallel axes work as a spherical joint to move a platform that holds a slider joint, resulting in extremely precise movements in

3D space with an absolute positioning accuracy of 20μm and motion resolution of 10μm even under loads of up to 500$Newton$ [76] [77].

Figure 2.4: Universal Robots Systems, Evolution1.

The Evolution 1 system has a height of only 500mm thus being smaller than most surgical robots. It is able to compute the movement of all axes in less than 120μsec. It comprises an universal adapter so it can incorporate different types of surgical instrumentation like endoscopes and high speed drills. However and due to the rather small working range, it must be pre-positioned approximately 5cm above the entry position in the desired orientation. Its user interface was implemented recurring to a touch screen and a master joystick device where one can for instance, select the speed and control the movement execution of the end-effector [76] [77].

The trajectory followed by the end-effector instrumentation is set preoperatively by a planning software system *VectorVision*, that uses imaging data acquired via a MRI 1.5 Tesla scanner. Intraoperatively, the patient's face is scanned for surface recognition using infrared flashes or laser surface scanning. This information is then used to match the MRI scans and, consequently guarantees a matching between the coordinates of the surgical field and the three dimensional

imaging data [77].

The main advantage of using a system like Evolution 1 is to achieve high precision holding and positioning while manipulating the endoscope, or handling other tools with smooth and slow movements within critical anatomical regions. This system can be adapted to use instrumentation for DBS surgery however, a high payload capacity is superfluous as both the instrumentation and the tasks are not weight demanding. It thus briefs the need for a parallel actuator that would occupy and greatly restrain the neurosurgeon workspace, (figure 2.4).

2.3.7 Minerva

The Minerva system was designed with the purpose of operating within a CT scanning machine combining the advantages of a robotic tool and the dynamic track of instrumentation with tomography images. It is mounted on a passive translation guide fixed in horizontal rails that move on rails parallel to the CT table. The manipulator has 5 DOF and 6 actuated joints, being the first 4 joints for positioning the manipulator's working axis along the defined trajectory and the last two, to approach the patient. The last joints are collinear sliders, however only the last one remains operative during the surgery, as the previous ones are disconnected after positioning the instrumentation in the correct direction and close to the surgical field. The $6th$ joint is used to guide the end-effector instrumentation in or outward from the patient head in a straight trajectory [78] [79].

Additionally, the robotic system relies on a Brown-Roberts-Wells reference stereotactic frame attached to the robot gantry and coupled to the motorized CT table by two spheroidal joints, to match its base referential with the referential used in the imaging software. As the robot operates within the gantry it must carry out all the procedure steps such as probe setting and orientation, skin incision, bone drilling, dura perforation, probe handling and probe exchange [79].

This robotic system was indeed used in 2 surgeries back in 1993 at the CHUV Hospital in Switzerland, but it was discontinued. The main reason behind this decision was the robot's limited DOFs and the fact that it did not get rid of the

stereotactic frame rendering it to be cumbersome. And finally, as Minerva was fixed to the CT it turned out to be unviable for neurosurgical procedures because of the longer operation times and neither the scan was available for diagnostic imaging during the procedure [78].

Aside from these disadvantages, this system could be adapted to assist DBS operations. Despite this fact and adding to the previously stated drawbacks, having a manipulator operate within a CT machine is not necessarily an improvement since MRI provides greater contrast to segment anatomical structures in soft tissues when compared to CT. Furthermore, if we consider that the brain shift in a keyhole procedure like the DBS surgery is negligible and that the used electrodes and canullas keep the linear profile when being inserted, there is no need for an online tracking of the electrodes. It's autonomous functioning nature can also work as an inconvenience, because Minerva system withdraws the neurosurgeon's decision capacity in the outcome of the procedure.

2.3.8 NeuroArm

The awarded system NeuroArm developed by Dr. Garnette Sutherland from the University of Calgary and engineers from Macdonald Dettwiler and Associates (MDA) was introduce in 2002 as an extent of human abilities, representing a potential dramatic change in the way surgery is performed. The Neuroarm project aims to take advantage of a MR-environment as well as recent advances of technology like haptic feedback, 3D image reconstruction and hand-controller design. It claims the title of the first image-guided, MR-compatible surgical robot capable of microsurgery and stereotaxy. It is constituted by two 7 DOF manipulators semi-actively actuated in a master-slave control type and commanded by hand controllers at a remote workstation. The human-robot interface filters undesired hand tremors and can scale the movement of the controllers relatively to end-effectors [48] [80].

The NeuroArm is built towards neurosurgery precision tasks, so each arm has a limited payload of 0.5 Kg, force output of 10 N, a tip speed that ranges from

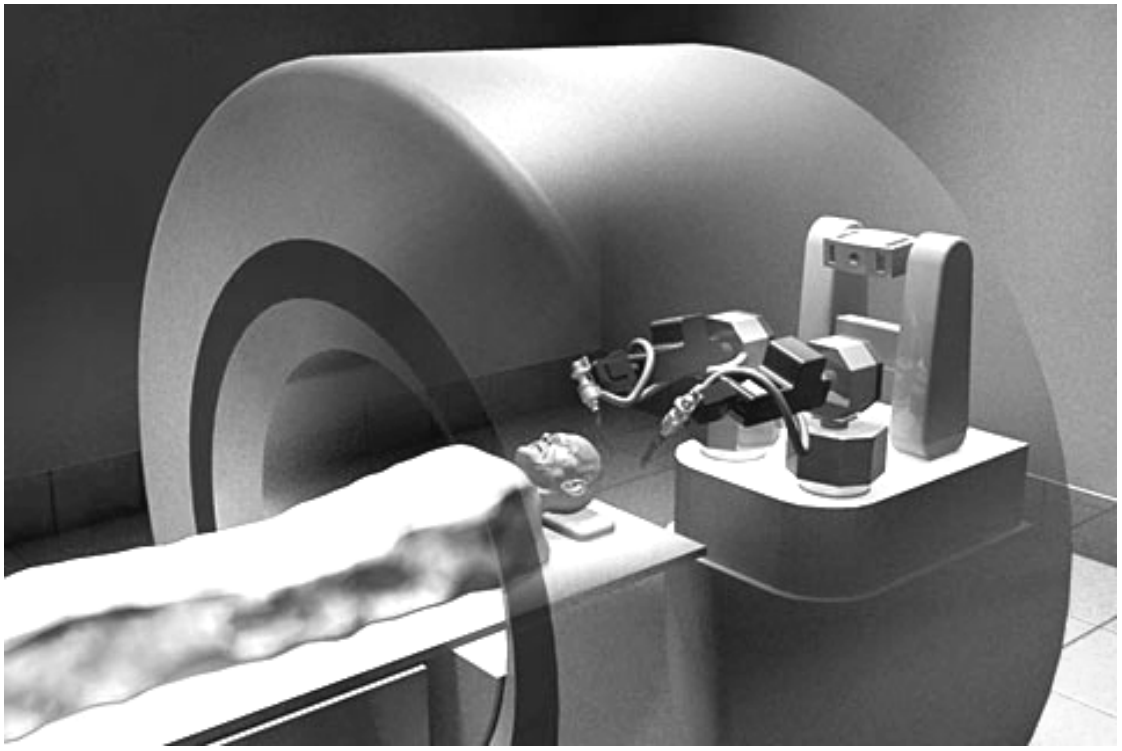


Figure 2.5: NeuroArm concept[6].

0.5 to $5mm/sec$ and a submilimetric 3D spatial accuracy. Patient safety was a paramount concern throughout the development of the robotic solution. For instance if the robot happens to leave the safe zone, a routine of *active workspace constraining* would trigger to avoid possible complications. This policy granted this system a Canadian Standards Association approval in 2007, Institutional Ethics and Investigational Testing approval by University of Calgary and Health Canada in 2008, (figure 2.5).

This robotic system is both capable of microsurgery and stereotaxy which granted it the place among the robotic platforms able to assist a neurosurgeon in a DBS procedure [81]. Despite increasing the surgery time, its precision, steadiness and compatibility with a planning software resulted in reduced trauma and blood loss, according to Pandya et al [48]. The end-effector positioning could be verified by overlaying 2D and 3D MRI information of preoperative and intraoperative, respectively. After positioning, a Z-Lock feature is used to restrict the tool

[6]**subsection 2.3.8** Available: `http://img1.qq.com/tech/pics/6270/6270273.jpg`. (Accessed: 03-Mar-2012)

movement along the defined longitudinal trajectory. The foresaw drawbacks of this robotic system are the necessity of an available MRI scanning machine during the whole surgery, the maintenance and acquisition costs, which are further increased due to the manufacturing requirements of producing a manipulator with exclusively non-ferromagnetic materials, primarily titanium and polyetheretherketone [80].

2.3.9 Other robotic systems for stereotactic procedures

Arata and his colleagues [82], proposed a robotic system for neurosurgery that unlike the serial manipulators previously described can be mounted in the stereotactic frame to aid surgeons in brain tumor removal surgeries. Although such system is still in a research stage, it offers a different approach for a similar problem and in our point of view this ideas should not be neglected. Brain tumor resection briefly resembles DBS procedures, it is also a difficult and time-consuming practice where even the slightest error in tumor resection can dramatically lower the clinical outcome [83].

Other robotic teleoperated systems like Da Vinci can successfully manipulate instrumentation while scaling down surgeon's movements and filtering tremor however, in brain tumor removal it is critical to locate the lesion. The aimed solution must therefore include an imaging navigation system to aid the neurosurgeons detect the target and evaluate if the tumor was totally resected. Then a precise robotic arm controlled by the surgeon would enhance his/her motion precision to effectively remove the tumor, which should ultimately lead to improved clinical outcome.

The robotic system proposed by Arata et al. can be divided into a surgical motion base and a surgical tool manipulator, controlled by a master device. The surgical tool manipulator incorporates a 3D endoscope, optical lights, an irrigation system and the volume control suction tool. However, what distinguishes this

[7] **subsection 2.3.9** Available: `http://arata.web.nitech.ac.jp/fig/motionbase.jpg`. (Accessed: 09-Mar-2012)

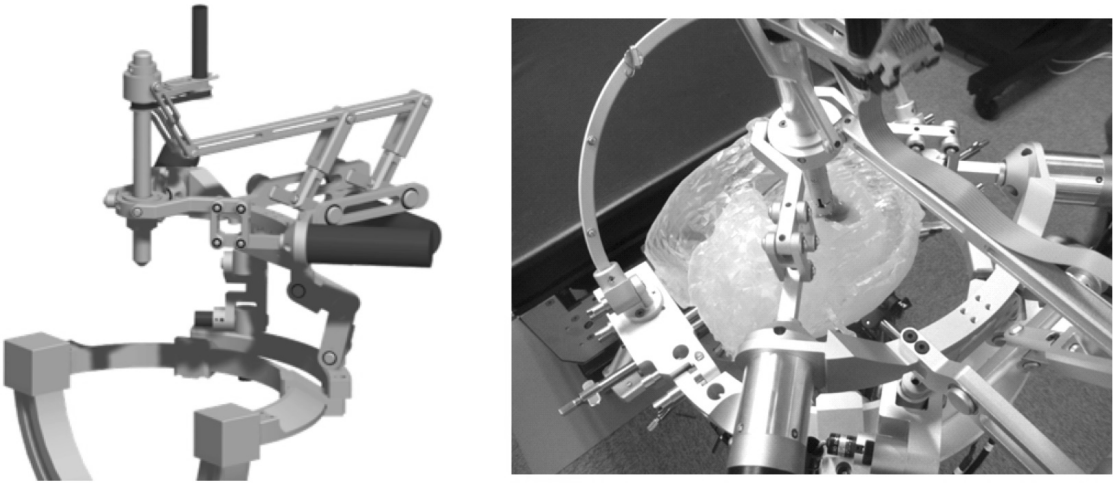


Figure 2.6: Brain tumor resection surgical robot[7].

system from other solutions is its surgical motion base, (figure 2.6). It is a 7 DOF robotic platform responsible for positioning the tool manipulator. It is mounted in a conventional head frame and through its parallel mechanism it can achieve high rigidity and accurate movements. The robot physical dimensions are 365 x 300 x 290 mm and its weight is circa 3kg with all the actuators included.

The surgical base achieved an accuracy of 0.04 mm along all axes, with a rigidity of 6.5N/mm. The surgical tool manipulator registered an average accuracy of 0.015 deg, and an average rigidity of 28N/mm. This surgical robot shows promising results and could eventually be adapted to deep brain procedures. For the best of our knowledge it is still in a research state and a practical evaluation in animal/cadaver models is still needed to infer its advantages and potential problems.

2.4 Current Challenges and Directions

In the previous section, it was already mentioned some current problems with state of the art solutions. From the variety of robotic systems for DBS, the few that are neither in a research phase nor adapted solutions are yet a considerable expense for a healthcare institution. The current neurosurgical robotic systems are only accessible to high-end healthcare facilities, which denies such foremost

instrumentation to the majority of the population. This fact becomes even more concerning, if we consider the epidemiology data of DBS sensitive diseases that indicate higher incidence rates in lower income countries with further financial strain.

In conclusion, neurosurgeons long for a low budget and simple platform that can fulfill their requirements in terms of handling, guiding precisely instrumentation and be able to aid the surgeon without the need of a cumbersome stereotactic frame. It should optimally have a mobile base, a pragmatic and reliable method to attach itself to the operating table and register its positioning relative to the patient. Also, as most institutions have already fully implemented navigation software, the ideal solution would be to adapt the system to it, instead of imposing a new one. Along the desired features come most of the current challenges, presented below.

Safety

We will start by addressing *Safety*, because it plays a critical role in neurosurgery assistive behavior and is the most cited reason behind the medical team's apprehension to use robotic neurosurgery equipment. In the scope of a DBS assistive role, safety definition is molded by high precision standards with consistent repeatability, being able to provide a stable and steady platform for instrumentation positioning. The robot should not budge while performing a task and there must exist a watch mechanism to prevent external inputs to interfere with the robot's performance. According to this requirements, the robot is recommended to move in low velocity in order to avoid further stress/overheating the joints, and also facilitate a motion stop action since the movement inertia is fairly less [84].

The robot position should be frequently assessed to check if the robot is capable of restraining its end-effector movements within the desired trajectory. If the positioning error deviates from the ideal position more than a defined threshold, the system must warn the user and enter a safety subroutine either to rectify its position or to stop all the manipulator actions. In any case, the robotic system should be prepared for these eventualities and at any time allow the surgeon to

remove the robotic equipment and carry on the surgery manually.

The final issue we wanted to address in terms of safety, is the need to provide the robot with collision avoidance algorithms. Even moving at low speeds, the robot collision must avoid equipment or people to prevent possible damages. Moreover, the robot motion path and performance is directly affected/altered by collisions. Regarding this issue, the desired collision avoidance routine should consider the static environment where the robot is placed, but also the dynamical component of the workspace or the unpredictable factors. The motion planning core of the robotic control should stop or adapt the robot posture if any obstacle interferes with the planned trajectory.

Haptic feedback

The capacity to handle and guide electrodes along a predefined trajectory, is a feature common to almost all robotic solutions for DBS surgery. However, using the same manipulator to through a commanded input, trepan a burr hole, requires the capacity to handle higher payloads and the addition of an haptic feedback. The fidelity in reproducing the sense of pressure while drilling is crucial to judge when to stop and avoid potential lesions [73]. Haptic feedback involves force and tactile pressure perception however, a tactile feedback feature is neither a requirement nor an advantage in a neurosurgical robot oriented to stereotactic electrode placement procedures. Notwithstanding the factual value of this feature in a surgical robot, haptic feedback is still subject of ongoing research and despite proving that it could achieve better results than a system without feedback in telecontroled manipulation, it does not reliably represent yet the pressure felt by the manipulator [85]. Moreover, the tasks performed during DBS neurosurgeries more than a sensitive control architecture, they require precision and steadiness. Such requirements could be achieved without depriving the neurosurgeon from the direct control of instrumentation, simply by using the robot as a cooperative and motion rectifier.

Intraoperative registration

The mobility and flexibility to mount and dismount any equipment used inside the operating room is a premise for its acceptance. A robotic system for assistance in DBS surgery is usually large and despite its applicability in biopsies or other minimally invasive procedures its use does not extend to the majority of neurosurgical practices. Having an equipment like that stationary inside an operating or imaging room would restrict it to other practices, resulting in delays and financial losses to the institution. However, having to mount an equipment each time before the surgery, requires a consistent method to relate the preoperative coordinates to the robotic referential. The state of the art systems presented, tackle this problem from different angles however none of the solutions seem to be spotless. Frame based configurations are cumbersome, frameless approaches lack the necessary precision, fiducial markers detection can be mislead by lighting conditions and optical surface recognition accuracy is still far from ideal [1], [66], [48], [80].

Image guided robotic systems

A major problem with neurosurgery or any other soft tissue surgery, is the deformability and mobility of tissues, in this particular case within the skull. There are many approaches to account for this brain shift problem, intra-operative imaging, boundary condition methods, indentation and mathematical modeling. The question we need to answer here is whether it is beneficial or not to use image guided robotic systems, or resort to other methods to minimize the impact of brain shift. Using MRI-guided instrumentation, one is capable of dynamically keep track of the brain structures and of electrodes, and possibly make adjustments. On the other hand, it implies building a robot from non-ferromagnetic materials exclusively, which further enlarges its price and may affect its final precision. Also the fact that the robot is assigned to a MR machine, means that it requires for the institution to ensure a MRI machine, associated equipment and conditions, which beyond the costs involved prevents the use of that equipment for any other practices.

Surgical simulation

Neurosurgical simulation has been referred by several authors as an important tool to test the tools used at the procedure and to practice surgery or train new surgeons, without causing any damage to humans in the process. This technology could also serve to evaluate the surgeon's skill and report errors and possible ways to improve it. There are some commercially available systems for surgical simulation like: RoSS™ oriented for stand-alone surgery and provides the experience of controlling a da Vinci® robot, or *NeuroTouch* more specifically oriented to brain surgery with the *Cranio* version. Also in this area, it is expected a simulator as close as possible to reality that can ideally provide multi sensorial instant feedback and accurately emulate the patient reaction along the procedure.

Other research fields

Besides the investigation in unresolved issues of current solutions, other research fields like nano robotics are giving the first steps towards providing nano robots with the ability to monitor neural electrical activity in a non-destructive way. The results of this study could in the future give rise to the development of nano actuators to perform non-invasive surgery within the brain [86]. Other non-invasive methods like transcranial Direct Current Stimulation (tDCS) allows for interacting with brain endogenous activity and performance by applying electric fields. As DBS is all about stimulating neural circuitry, this non-invasive method seem to fulfill the needs however, its efficiency drops when targeting deep structures and it can not stimulate as accurately as a cortical placed electrode yet [87].

Part II

Developed Solution

Chapter 3

Industrial Robot System Search

In pursue of a robotic system to satisfy our needs, we made a thorough market search on industrial robotic manipulators. Why is the choice of the manipulator important for the final aim of our project? The robotic system concept for neuro-surgical purposes has a singular set of characteristics that are rather specific and often deviate from the common scope of industrial oriented robots. Acknowledging the variety of tasks a robot could perform within an assembly line, most robot manufacturing companies today present a wide range of solutions with different sets of features for specific goals:

- High-payload capacity;
- Fast cycle task execution times;
- High movement stiffness, rigidity;
- Light and/or compact body and controller;
- High movement flexibility;
- Large/Small workspace;
- Mounting flexibility, environment tolerance;
- Hardware/Software adaptability to tasks;
- Interface user-friendliness;
- Custom specifications.

3.1 Robot features

Here are some examples of commonly sought features. As the manipulator choice is not straightforward and can not be based on our impulse, in this section we will explain the relevance of each feature in the context of our problem and the thought process behind the decision of the robotic systems. The information here presented for each robot was entirely gathered from the manufacturer datasheets. Robot characteristics with little or no relevance to our problem will be disregarded. We are not affiliated to any of these companies, hence we have no underlying interest on this review other than justifying the choice of the selected robot systems for our project.

3.1.1 Mechanism type

The first question we need to answer is whether to choose a serial or a parallel manipulator. In a manipulator with a parallel mechanism, each joint has an independent contribution to the end-effector motion and its links connect the base to the end-effector platform. A parallel manipulator provides rigid and stable movements in the platform, and is (normally) assigned to heavy duty tasks, due to its high payload capacity. On the other hand, these manipulators have a very limited workspace and their kinematics are in general more complex and are frequently undetermined systems.

Serial manipulators are described as a sequence of rigid bodies connected by means of actuated joints that go from the base to the end-effector. The movement performed by the end-effector depends on the consecutive displacements of each joint, meaning that it accumulates errors from every single joint displacement. A serial manipulator has less movement rigidity and steadiness then a parallel mechanism, which can still be overcome by using more accurate motor-actuators. Another disadvantage of serial robots, is their limited payload capacity. However, they are still the most common manipulators used in industry due to its high flexibility and broader dexterous workspace that allows it to avoid obstacles and grasp objects from different angles and with different poses.

When we contextualize this information in the scope of our problem we notice that, the robot will not be required to manipulate heavy instrumentation and despite the needed movement and positioning precision, it is also required some dexterity to position the arm according to defined trajectories without crossing the surgeon line of sight. Therefore we aim for a serial manipulator.

3.1.2 Degrees of Freedom

The DOFs in a serial manipulator, defines the number of independent motions that can be performed by a robotic manipulator. Also the distribution and typology of joints along the manipulator defines its flexibility and limits the dexterous workspace. When selecting a robotic arm, the number of DOF, their type and sequence along the arm should be chosen to match the procedure requirements.

The robotic system is expected to position and hold instrumentation along defined trajectories. Thus we established that the manipulator should reach a cartesian three dimensional position with a desired end-effector orientation, within its dextrous workspace. To satisfy this condition, the manipulator should have at least 6 DOF. With more than 6 DOF, the manipulator becomes kinematically redundant as there would be a non direct relation between the 6 target coordinates (position and orientation) to more than 6 joints. Robots with more than 6 DOF, have additional sources of error from the extra joints, but their redundancy actually enhances the arm flexibility. One can determine the robotic manipulator spatial configuration from an infinite number of solutions to avoid collisions with intraoperative personnel or instrumentation.

According to the presented information, we chose to test our solution using 6 DOF and 7 DOF robotic manipulators, which shall be described in section 4.5.

3.1.3 Rigidity

Camarillo et al. [53] addressed another important set of characteristics namely movement inertia, joint stiffness and speed/force tradeoff. The inertia of a robot is defined by its mass and density of the building material. Having higher inertia

implies lethargic movements because it is harder to accelerate or decelerate a larger mass. A higher inertia is normally directly related to a higher payload capacity, however within a neurosurgery context the payload is not a restrictive requirement. Furthermore a higher inertia causes the robot to build up more kinetic energy as it moves, which raises more safety concerns.

Stiffness on the other hand stands for the capacity of a robot to maintain a determined position when nudged by an external force. The stiffness is granted by the material and geometric disposition of the manipulator. This is one of the most sought features in a surgical assistive robot as it allows a superior control, accuracy and steadiness. However, joint stiffness is rarely displayed in the manipulator datasheets and we can only infer about this information.

The motor actuators can also be design towards force and speed driven tasks. In these cases, the robot is optimized to perform repetitive tasks at very high cycle times, thus enhancing productivity. However, productivity is not a concern in surgery procedures. In light of our objectives, such features can be easily replaced over precise and stiff task execution.

3.1.4 Workspace

Workspace is the volume around the manipulator that can be reached by the end-effector. It is determined by the manipulator links dimensions, joint mechanical limits and restricted by configurations that result in collisions between robotic body links. Even among the serial manipulators there are several types of structures, designed for different purposes: cartesian, scars, cylindrical, spherical and anthropomorphic.

Cartesian and scara robot types are both designed for pick and place, assembly and packing tasks. A cartesian robot moves along the 3 cartesian axes XYZ and it is known by the rigidity and stability of motion ideal for straight-line higher payload tasks. On the other hand the scara robot type, does not require a rail structure to operate which makes it more portable and capable of operating in a smaller scale. Scara robots have a cylindrical workspace envelope and 4 actuated

joints, which provides an extra DOF and thus more motion flexibility compared to the cartesian.

Cylindrical and spherical robot types are built towards specific workspace requirements. As their names suggest they have cylindrical and spherical envelope workspaces. One of the most remarked disadvantages has to do with the manipulator fixed configuration, which prevents it from reaching a target that is in a straight-line behind an obstacle.

According to the intraoperative conditions, space limitation, mobility requirements and task specifications, we concluded that the anthropomorphic configuration would be the most appropriate. Its structure resembles the human arm, and among the configurations is the one with most dexterity since all joints are revolute. In a typical 6 DOF anthropomorphic manipulator, the arm is formed by a cylindrical and two consecutive revolute joints, being the last called elbow, which connects the arm to the forearm. The forearm is formed by a cylindrical, a revolute and another cylindrical joint that connects to the end-effector, (check figure 4.11 in section 4.3).

3.1.5 Force and Position Control

Another robot feature has to do with the type of control exerted on the manipulator. Having already selected the structure, we still need to decide how to operate it.

The control application can be developed to move the robotic manipulator based on direct joint displacements or by indirectly selecting a position and orientation for the end-effector to reach. Whatever the case, there must be a communication protocol between the robotic actuators and the control software, in which the current joint positions and target joint displacements are interchanged. Since we aim to keep the robot control as simple and intuitive as possible, we devised a control architecture based in world coordinates for position and orientation that should indirectly control the joint displacements.

As the manipulator actions will occur in a controlled environment where the

free space is defined from the beginning, it is not required of the robotic system to dynamically adapt to the changes in the environment. Moreover, it is intended for the manipulator to have a predictive and well defined behavior for each input, so that the neurosurgeon can also foresee its movements and feel comfortable with it. Consequently, it is possible to abbreviate the motion planning problem by manipulating the robot through point-to-point movements instead of computing the desired path.

Finally, it is also important to decide whether the robot should be controlled through position and/or force, knowing that such decision is primarily based in task requirements. In the proposed solution, the robot should be manipulated through both control types for different tasks. This issue will be further explained in chapter 4.

3.1.6 Precision and Repeatability

Precision and repeatability are unquestionably desired robot specifications for almost any industry. The ability to consistently execute sub millimeter scaled tasks is one of the advantages that sets robots apart from human labor. Although important some industries tend to overlook this aspect in favor of other priorities.

When we narrow our scope to neurosurgery and particularly to this project, the precision and repeatability are the most critical features. At the same time, they are also often omitted on the manufacturers data sheets. One of the reasons behind this decision relates to the fact that the robotic manipulator precision is not a constant value throughout the workspace. It varies for example with the displacement of each joint actuators relative to their home position, since in most cases, more rotation means less precision.

It is difficult to anticipate how each manipulator behaves and what level of precision it achieves without practical tests and real robot simulations. Consequently the simulation will play an irreplaceable role in the development of a robotic system of this nature.

3.2 Available robotic systems

After fully understanding the purpose of a robotic system for an assistive role in deep brain stimulation surgeries and after selecting a set of desirable characteristics, we proceeded to do a market search. However some of the discussed features like: joint stiffness, movement inertia, backdrivability[1], can not be evaluated based on the information gathered from the robotic systems datasheets.

The prize of the complete a robotic system is hardly accessible at the manufacturer website, but is also particularly difficult to obtain. After contacting the respective companies, and upon some conversation we managed to get an average price of a complete robotic system of around 20 000 €, which is significantly less than the average price of any available neurosurgical robot (more than 20 times) and is even cheaper than the mechanical surgical stereotactic frame used in standard DBS surgical procedures.

Some information presented in the datasheets can not be directly used as a term of comparison between systems. One example is the maximum joint displacement, velocity or acceleration. These values are often very similar among all systems, but even when they differ there is no easy way to anticipate their implication in a practical environment. So this is another reason to test each solution in a simulated environment before acquiring the system.

We compiled and listed all the serial manipulator robotic systems, with anthropomorphic configuration and 6 DOF from well-known and market established companies like: **Abb**, **Adept**, **Epson**, **Fanuc**, **Kuka**, **Mitsubishi**, **Motoman**, **Nachi**, **Schunk**, **Stäubli**, **Toshiba** and **Universal Robots**. A 7 DOF robotic manipulator (Schunk Amtec LightWeightArm) was also evaluated for two reasons: to assess its feasibility as we are adding and extra DOF and to test our approach in a real environment, as this particular robot manipulator is available in our laboratory for testing. Despite lacking the required precision which limits its applicability in an operating room, it suffices for assessing the kinematic equations

[1]Feature that allows the user to maneuver the manipulator by grabbing and moving the end-effector, also known as floating mode.

and test the control architecture.

The first characteristic to be compared between all the available robots is the weight of both the manipulator and the controller, (figures 3.1 and 3.2). As the robot must be carried in and out of the operating room for each surgery, it must be as mobile and portable as possible therefore its weight is a limiter factor. In figure 3.1, we can see that most robotic manipulators fit inside a 20 to 40 kilogram range, which is acceptable. Above 40 kilograms, the solution becomes overly cumbersome.

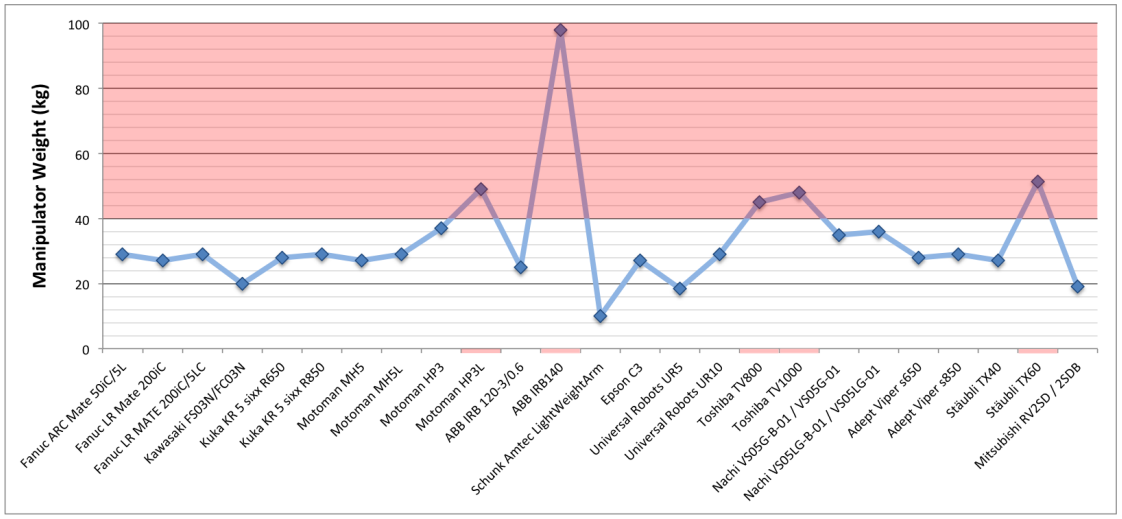


Figure 3.1: Manipulator weight.

The same can be said about the weight of the controller that must be brought together with the manipulator to the operating room. However in this topic, the differences are more pronounced, (figure 3.2). Also for the controller weight we dare to set a maximum limit of weight of 50 kilograms, which leaves out some choices.

The positioning and guiding assistive behavior do not require a high payload capacity and as we can see in figure 3.3, the selected robots present payload values recommended for the expected use. However, to avoid possible malfunctions and to safeguard the consistency of the system from unpredicted circumstances we discourage the choice of manipulators whose payload capacity is inferior to 3kg.

Across the features, the horizontal reach has a particularly significance in our solution, (figure 3.4). As seen in the operating room, the neurosurgeon team

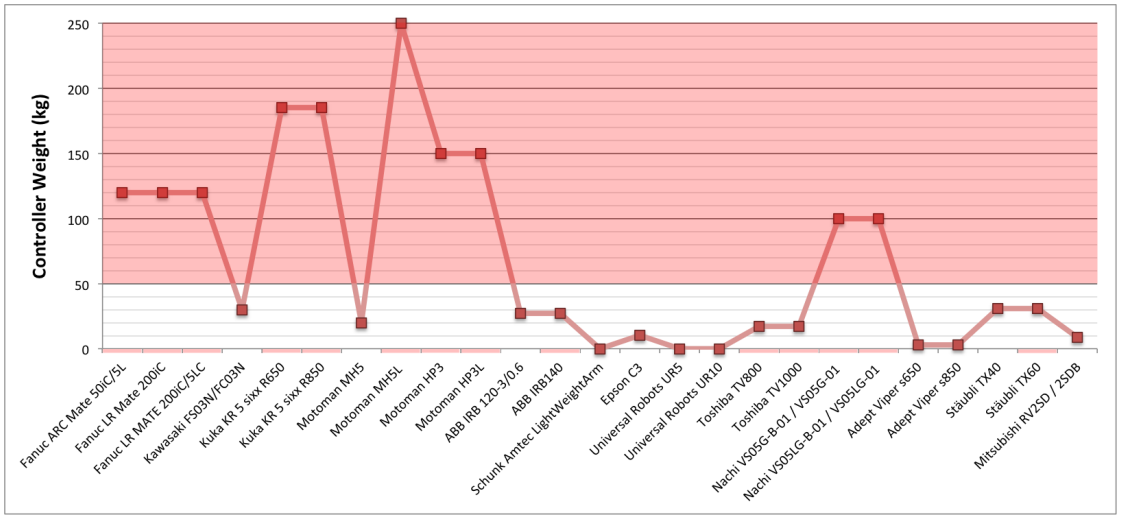


Figure 3.2: Controller weight.

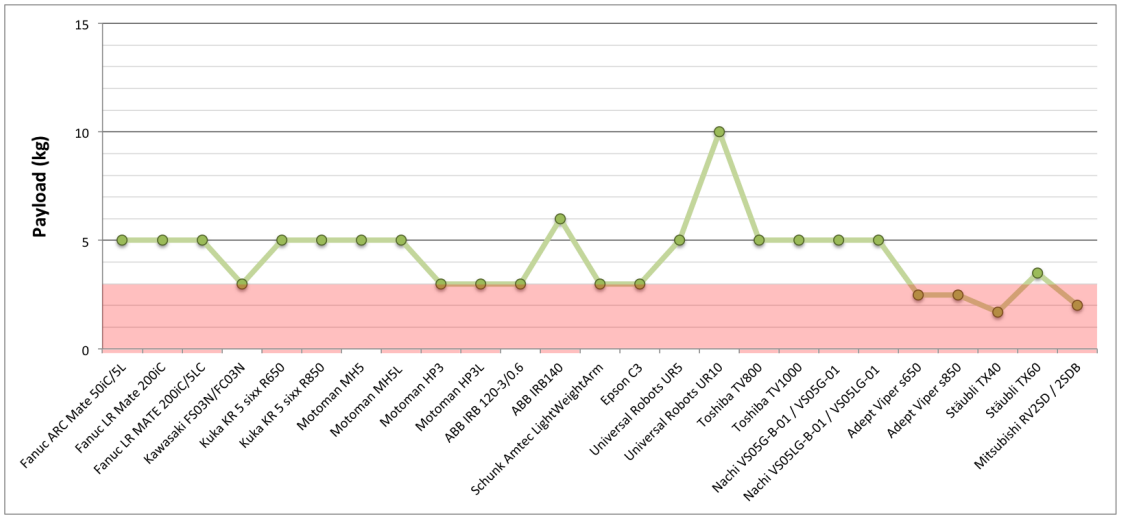


Figure 3.3: Payload.

operates within a restrictive and relatively small area, which will be shared by the robot. The manipulator can not be so large that it will not fit the existing space nor so small that it has no flexibility in its positioning and does only reach the target coordinates from specific spots. Such inflexibility can be compromising if the viable manipulator poses block the surgeon's view or workspace. The horizontal reach values of the candidate systems fall within an acceptable range ($500 \sim 1000$ mm), but can only be truly evaluated in simulation.

Lastly but perhaps the most vital in view of our project, is the repeatability of movements that stands for the maximum expected deviation of position for the

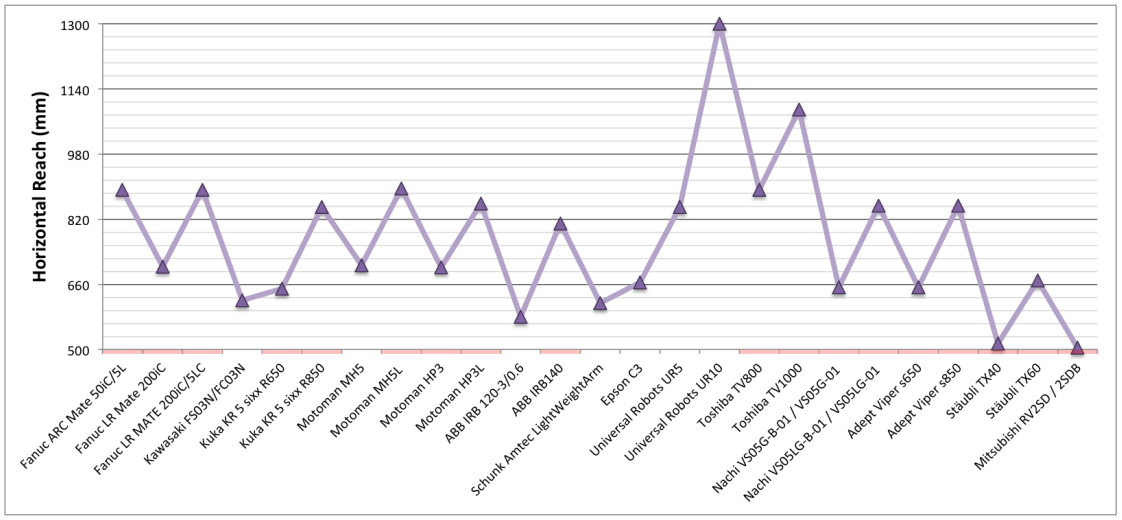


Figure 3.4: Horizontal reach.

same input using the same device, (figure 3.5). More than repeatability the most interesting characteristic to differentiate systems would be the point accuracy but it is rarely displayed in the information provided by the manufacturer. Also for this feature, the short range of values shows the similarity of the candidate systems. Thus, we are looking for the system with least repeatability, preferably inferior to 0.1 mm to best the repeatability achieved by the standard stereotactic frame.

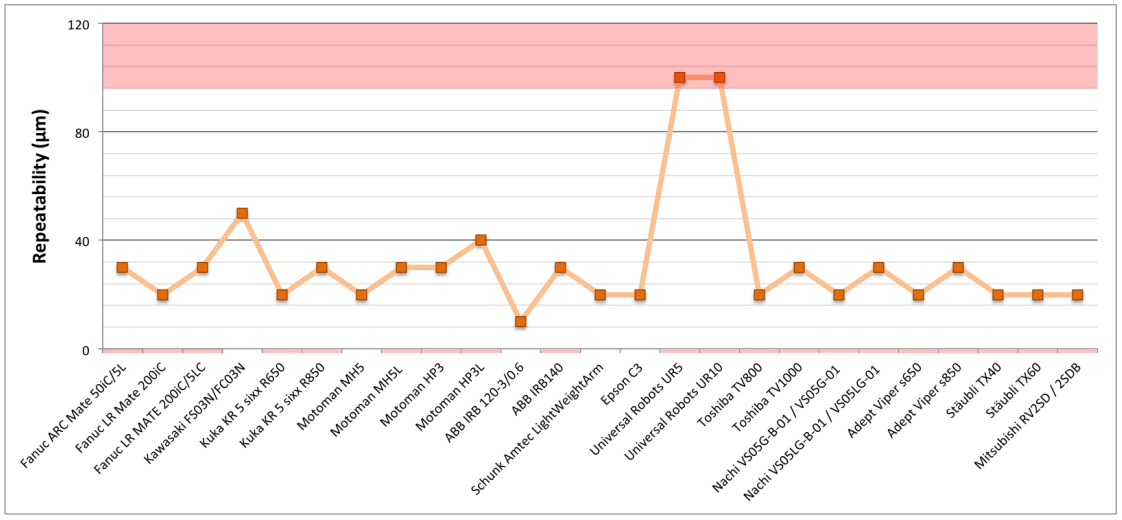


Figure 3.5: Repeatability.

After reviewing the robot characteristics we looked for the companies of the best candidate robotic systems ideally with branch offices in Europe to avoid the

customs costs. We finally decided for the Motoman MH5, Abb Irb 120 and Schunk Amtec LightWeightArm II. The last one because it is available for testing in our laboratory.

Chapter 4

Manipulator Kinematics

In this chapter we will introduce the fundamentals of kinematics to study the geometry of manipulators using mathematics to describe positions and transformations in tridimensional space. We decided to present the theoretic background on manipulator kinematics and the specific solutions/features for the selected robotic systems. Background information about this section can be found at [88] [52] [89].

4.1 Kinematics Problem

Robotic tasks are performed by completing a sequential set of predefined motions, directed by the controller subsystem that provides input to the robot actuators while taking into account the values read from the sensors. However, specifying the correct input to each actuator involves the analysis of the robot mechanical model to obtain the desired output.

The first step toward setting the controller algorithm is to outline a mathematical model of the robot that describes how the body moves according to given directives and thus learn its input/output characteristics. Kinematics is the area of mechanics that studies the laws of motion of bodies, independently of the causes. Only purely geometric aspects of the movement are considered, such as the position or higher order derivatives of position like velocity or acceleration relative to an independent variable such as time. Manipulators can be described as a set of rigid links connected by actuated joints that can either be revolute or pris-

matic producing respectively angular or offset displacements relative to neighbor links. These links can be serial or parallel to each other depending on the desired behavior and robotic features.

One can describe the position of each structural element of the manipulator, relatively to the joint variables ($\mathbf{q}$) or to the tridimensional space ($\mathbf{x_e}$) in which the robot is inserted. *Joint Space* stands for the set of variables that parameterize each joint variable commonly a (n $\times$ 1) joint vector,

$$\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \tag{4.1}$$

while *Cartesian space* describes the spatial coordinates and orientation of the end-effector. The position can be represented by means of a XYZ (3 $\times$ 1) vector $\mathbf{p_e}$, and the orientation is briefly specified in this dissertation by Euler Angles $\gamma_\mathbf{e}$ vector (3 $\times$ 1),

$$\mathbf{x_e} = \begin{bmatrix} \mathbf{p_e} \\ \gamma_\mathbf{e} \end{bmatrix}. \tag{4.2}$$

The problem of *Geometric Direct Kinematics* consists in computing the tool frame position and orientation relative to the base reference from a set of joint angles and knowing the structural model of the robot. In other words, it means to compute the *Cartesian space* representation, knowing the *Joint space* variables,

$$\mathbf{x_e} = f(\mathbf{q}) \tag{4.3}$$

On the other hand, if we want to calculate all the possible joint angle solutions to attain a given end-effector position and orientation, this problem is called the *Geometric Inverse Kinematics* problem. Since in our case the task is assign in a workspace because it is easier to the user, there is great interest in translating the coordinates and orientation from the *Cartesian space* to the *joint space*,

$$\mathbf{q} = f^{-1}(\mathbf{x_e}) \tag{4.4}$$

This latter problem opposite to direct kinematics, does not always have a possible and determined solution due to the non-linearity of the inverse kinematics equations, and might even be unreachable if the target position falls outside the workspace, collides with the robot body or reaches a singularity.

Likewise geometric kinematics equations that establish the relation between joint angles and end-effector position and orientation, there is the differential kinematics, which relate the joint displacements/velocities to end-effector displacements/velocities. The *Differential Direct Kinematics* problem computes the tool velocities from each joint velocity,

$$\mathbf{v_e} = g(\dot{\mathbf{q}}) \tag{4.5}$$

and from the inverse relation, one can calculate each joint velocity to achieve a final end-effector linear and angular velocity through *Differential Inverse Kinematics*,

$$\dot{\mathbf{q}} = g^{-1}(\mathbf{v_e}) \tag{4.6}$$

Differential kinematics allow the user to easily control and have feedback on joint/end-effector velocities, forces or torques and permit the manipulation of instrumentation through incremental movements. Furthermore, it allows the user to control the manipulator through back drive (in a free-mode) or along a desired trajectory, which are some sought features for this project.

4.2 Spatial Descriptions and Transformations

A consistent method to define the position and orientation of an entity in tridimensional space is one of the premises to study kinematics. We will represent this information relative to the universe coordinate system, or other Cartesian referentials relative to it. The origin frame is the coordinate system $\{O\}$ formed by $\hat{x}, \hat{y}$ and $\hat{z}$ normed and orthogonal vectors that stand as the frame axes. To describe a point in this referential we need a (3×1) vector ($^A\mathbf{P} \in \mathbb{R}^3$), which we will call a *Position Vector* and will be represented as

$$^{A}\mathbf{P} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T . \tag{4.7}$$

To specify an entity orientation in space, it is assigned a coordinate system $\{A\}$ to the body in question, whose axis are then described relatively to the reference frame. If we express the unit vectors of the object axes $\hat{x}_A, \hat{y}_A$ and $\hat{z}_A$ relatively to $\{O\}$ we get,

$$^{O}\hat{x}_A = \hat{x}_{A,X} \; . \; \hat{x} + \hat{x}_{A,Y} \; . \; \hat{y} + \hat{x}_{A,Z} \; . \; \hat{z} \tag{4.8}$$

$$^{O}\hat{y}_A = \hat{y}_{A,X} \; . \; \hat{x} + \hat{y}_{A,Y} \; . \; \hat{y} + \hat{y}_{A,Z} \; . \; \hat{z} \tag{4.9}$$

$$^{O}\hat{z}_A = \hat{z}_{A,X} \; . \; \hat{x} + \hat{z}_{A,Y} \; . \; \hat{y} + \hat{z}_{A,Z} \; . \; \hat{z} \tag{4.10}$$

which built into a 3×3 matrix, results in what is called a *Rotation Matrix* of $\{A\}$ relative to $\{O\}$, conventionally named after the notation $^{O}\mathbf{R}_A$

$$^{O}\mathbf{R}_A = \begin{bmatrix} & & \\ ^{O}\hat{x}_A & ^{O}\hat{y}_A & ^{O}\hat{z}_A \\ & & \end{bmatrix} = \begin{bmatrix} \hat{x}_{A,X} \; . \; \hat{x} & \hat{y}_{A,X} \; . \; \hat{x} & \hat{z}_{A,X} \; . \; \hat{x} \\ \hat{x}_{A,Y} \; . \; \hat{y} & \hat{y}_{A,Y} \; . \; \hat{y} & \hat{z}_{A,Y} \; . \; \hat{y} \\ \hat{x}_{A,Z} \; . \; \hat{z} & \hat{y}_{A,Z} \; . \; \hat{z} & \hat{z}_{A,Z} \; . \; \hat{z} \end{bmatrix} . \tag{4.11}$$

Since both reference frames are orthonormal, the column vectors from the respective rotation matrix are mutually orthogonal. This property suggests that the inverse of this matrix is equal to its transpose.

$$^{O}\mathbf{R}_A = ^{A}\mathbf{R}_O^{-1} = ^{A}\mathbf{R}_O^{T} . \tag{4.12}$$

If we consider the frame {A} to be superimposed to frame {O}, by applying a rotation[1] to {A} around each of the {O} coordinate axes, we can describe the new frame unit vectors component's relatively to the reference axes according to (4.11) as follows. If we rotate θ around the x axis we get,

[1] **section 4.2** By convention we consider counterclockwise rotations positive.

$$Rotation_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4.13}$$

if we rotate ψ around the y axis we get,

$$Rotation_Y(\psi) = \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \tag{4.14}$$

and if we rotate ϕ around the z axis we get,

$$Rotation_Z(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.15}$$

The characterization of the orientation of a solid body in a tridimensional space can be specified by no less than 3 parameters, usually called *Euler angles* [90]. There are exactly 12 possible *Euler angles* combinations fulfilling the requirement of no consecutive rotations around the same angle. For the coherence of our approach we will only use the X-Y-Z fixed angles orientation also referred as *Roll, Pitch, Yaw*. Considering $\gamma = [\ \phi \quad \psi \quad \theta\ ]^T$, as the set of angles that describe the orientation around ZYX axes respectively and that all rotations occur about the reference frame, we can write the combined matrix of each contribution (matrices 4.15, 4.14 and 4.13) by premultiplying them as follows[2],

$$RPY(\gamma) = Rotation_Z(\phi) \quad Rotation_Y(\psi) \quad Rotation_X(\theta)$$

$$= \begin{bmatrix} c(\phi)c(\psi) & -s(\phi)c(\theta) + c(\phi)s(\psi)s(\theta) & s(\phi)s(\theta) + c(\phi)s(\psi)c(\theta) \\ s(\phi)c(\psi) & c(\phi)c(\theta) + s(\phi)s(\psi)s(\theta) & -c(\phi)s(\theta) + s(\phi)s(\psi)c(\theta) \\ -s(\psi) & c(\psi)s(\theta) & c(\psi)c(\theta) \end{bmatrix} \tag{4.16}$$

[2]**section 4.2** The Roll-Pitch-Yaw matrix (4.16) is written in a compressed form to fit the printing area. Therefore, c() stands for cos() and s() stands for sin().

As referred previously, all the vectors represented in a rotation matrix are unitary axes of an orthogonal frame, thus being mutually perpendicular. Consequently, the following six constrains arise from the nine rotation matrix elements,

$$\begin{aligned} |\hat{x}| = 1, \quad |\hat{y}| = 1, \quad |\hat{z}| = 1, \\ \hat{x} \cdot \hat{y} = 0, \quad \hat{x} \cdot \hat{z} = 0, \quad \hat{y} \cdot \hat{z} = 0\cdot \end{aligned} \tag{4.17}$$

The problem of finding the $\gamma = [\ \phi \quad \psi \quad \theta\ ]^T$ set of angles from a rotation matrix built as (4.16), is then abbreviated from solving nine to three equations as we have six dependencies (4.17). Considering a rotation matrix

$$R_{XYZ}(\phi, \psi, \theta) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.18}$$

the solution for ψ in the range $(-\pi/2, \pi/2)$ is

$$\begin{aligned} \phi &= \arctan_2\ (r_{21}, r_{11}) \\ \psi &= \arctan_2\ (-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \theta &= \arctan_2\ (r_{32}, r_{33}) \end{aligned} \tag{4.19}$$

whether if ψ is within the $(\pi/2, 3\pi/2)$ range, then

$$\begin{aligned} \phi &= \arctan_2\ (-r_{21}, -r_{11}) \\ \psi &= \arctan_2\ (-r_{31}, -\sqrt{r_{32}^2 + r_{33}^2}) \\ \theta &= \arctan_2\ (-r_{32}, -r_{33}) \end{aligned} \tag{4.20}$$

In the case that $\psi = \pm\ \pi/2$, the solutions (4.19) and (4.20) degenerate, as it is only possible to determine the sum or difference of ϕ and θ. Therefore to solve this cases, one can fix a value like $\phi = 0$, and find the solutions for $\psi = \pm\ \pi/2$,

$$\begin{aligned} &\phi = 0, && \phi = 0, \\ &\psi = +\pi/2 && \psi = -\pi/2 \\ &\theta = \arctan_2\ (r_{12}, r_{22}) && \theta = -\arctan_2\ (r_{12}, r_{22}) \end{aligned} \tag{4.21}$$

Now that we can fully describe a frame position and orientation in space relative to another, we use what is called a *homogeneous transform* to compactly represent

both rotation and translation. Being {O} our base referential, and {B} another referential in the same tridimensional space, translated by ${}^{O}\mathbf{P}_{B}$ and rotated by ${}^{O}\mathbf{R}_{B}$, we can locate the same point with both frames following the equation

$$ {}^{O}\mathbf{P} = {}^{O}\mathbf{R}_{B}\,{}^{B}\mathbf{P} + {}^{O}\,\mathbf{P}_{B} \tag{4.22} $$

The ${}^{O}\mathbf{R}_{B}\,{}^{B}\mathbf{P}$ part represents the target point described as ${}^{B}\mathbf{P}$ for the {B} frame, in an intermediate referential whose origin is coincident to {O} but with the orientation of {B}, achieved by premultiplying ${}^{B}\mathbf{P}$ by ${}^{O}\mathbf{R}_{B}$. Then we add ${}^{O}\mathbf{P}_{B}$ as the vector that locates {B}'s origin relative to {O}. In the end we want to transcribe this transformation as only one operand,

$$ {}^{O}\mathbf{P} = {}^{O}\mathbf{T}_{B}\ {}^{B}\mathbf{P} \tag{4.23} $$

The transformation matrix suggested by the operand ${}^{O}\mathbf{T}_{B}$ was stablished as the 4×4 matrix

$$ {}^{O}\mathbf{T}_{B} = \left[\begin{array}{ccc|c} & {}^{O}\mathbf{R}_{B} & & {}^{O}\mathbf{P}_{B} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{4.24} $$

Unlike the rotation matrix whose inverse is equal to the transpose (4.12), the *homogeneous transformation* matrix ($\mathbf{T}$) is not orthogonal and thus,

$$ \mathbf{T}^{-1} \neq \mathbf{T}^{T} \tag{4.25} $$

This operand will be rather useful along this section, as it serves to compactly describe the spatial transformations between frames, and consequently help understand the robot's behavior.

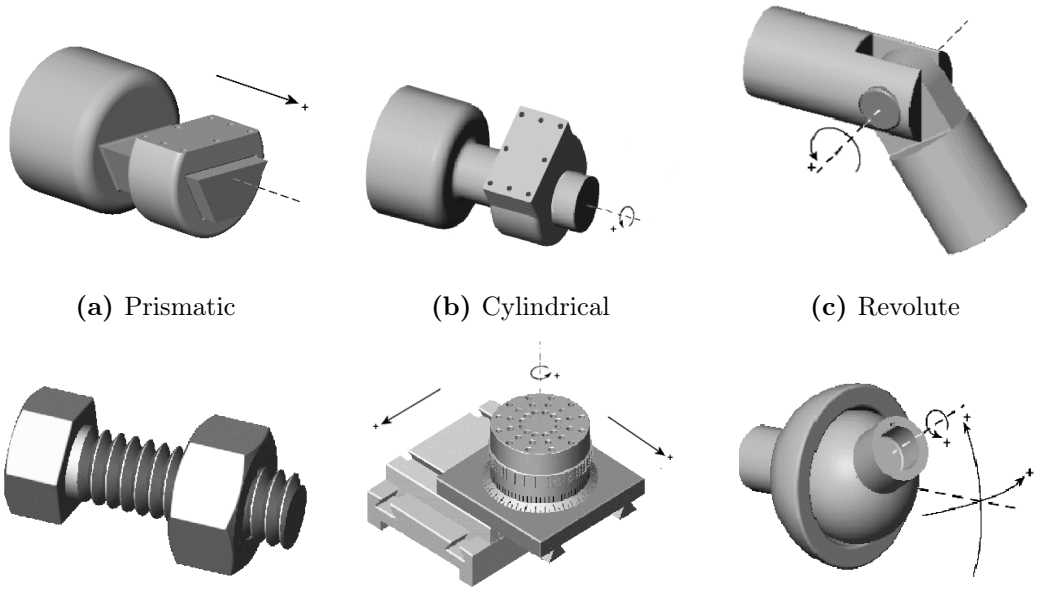
4.3 Geometric Kinematics

In this section, we will study the position and orientation of chain linked serial manipulators and how this parameters change during movement. We will address

both *Direct* and *Inverse Kinematics* geometric problems because they are essential to controlling and understanding the system we are trying to develop.

A robotic manipulator can be described as a set of rigid links connected through mechanically driven joints, known as a *kinematic chain*. The manipulator can be considered as *open-chained* or *close-chained*, depending whether the configuration has one sequence of links with two endings[3] or if the sequence of links forms a loop. In this dissertation we are particularly interested in open-chained robots.

Open-chained manipulators are formed by $n+1$ links connecting n joints, where the base link is usually immobilized. There are several types of joints that directly affect how the links move relatively to the previous one. Craig [88], listed six types of joints commonly used in industry, (figure 4.11).



(a) Prismatic **(b)** Cylindrical **(c)** Revolute

(d) Screw **(e)** Planar **(f)** Spherical

Figure 4.1: Joint types[4].

The robots studied in this dissertation belong to the R-R-R type, that stand for revolutive wrist, revolutive shoulder and revolutive elbow. Thus, our focus will be in revolute and cylindrical joints for the manipulator, and a prismatic joint for the end-effector actuated mechanism.

[3]**section 4.3** from the base to the end-effector.

The structure and functionality of the manipulator is strongly influenced by the type, number and order of joints as well as the properties of each link. Manipulators are usually built with one degree of freedom joints, hence the common assumption of the number of DOF from the number of joint articulations. Based in the selected industrial robotic solutions, we will focus 6 DOF and 7 DOF anthropomorphic serial manipulators.

4.3.1 Geometric Direct Kinematics

Regardless of the complexity of mechanical design of each link, the material properties, its shape and weight, it is enough to consider it as a rigid entity represented by a line in space when we are elaborating the kinematic relations.

To describe how the configuration of the manipulator changes, we need to assign a frame to each joint, to the base and to the end-effector. *Homogeneous transforms* are a tool to express and manipulate the relation between frames and to explain how they change over movement. Considering Frame 0 as base and Frame n as the end-effector, the transformation from the base to the end-effector is expressed as ${}^0\mathbf{T}_n$ and can be obtained through,

$$ {}^0\mathbf{T}_n = {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 ... {}^{n-1}\mathbf{T}_n \tag{4.26} $$

We used the *Denavit-Hartenberg* convention to calculate the direct kinematics of an open-chain manipulator. The convention sets a systematic method to describe the relative position and orientation of neighboring links, through four parameters $\alpha_{i-1}, a_{i-1}, \theta_i, d_i$. We start by setting the manipulator to its home position, and assigning a frame to each joint, using the methodology[5]:

1. For a joint (i), the origin of the {i} frame is set by the intersection of two lines coaxial with $link_i$ and $link_{i+1}$;

[4]**section 4.3** Available at: `http://www.mathworks.com/help/toolbox/physmod/mech/ug/f2-182101.html`

[5]**section 4.3.1** For further information regarding atypical cases, consult [52] pages 62 and 63.

2. Having the origin defined, the $\mathbf{Z}_i$ axis is defined along the joint motion axis;

3. The $\mathbf{X}_i$ is set along the direction from the origin of {i} to {i+1};

4. Finally the $\mathbf{Y}_i$ axis is fixed to complete a right-handed frame.

Upon assigning the frames to the respective joints, we define the four parameters that identify the relation between joints,

- α_{i-1}, angle of rotation around $\mathbf{X}_i$ axis;
- a_{i-1}, distance of translation along the $\mathbf{X}_i$ axis;
- θ_i, angle of rotation around $\mathbf{Z}_i$, after the rotation around $\mathbf{X}_i$ axis;
- d_i, distance of translation along $\mathbf{Z}_i$, after the rotation around $\mathbf{X}_i$ axis.

The *homogeneous transform* translates the position and orientation from frame {i-1} to {i} and is defined by the result of all four transformations,

$$^{i-1}\mathbf{T}_i = Rotation_X(\alpha_{i-1}) \cdot Translation_X(a_{i-1}) \cdot Translation_Z(d_i) \cdot Rotation_Z(\theta_i)$$

$$^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.27}$$

In the end, according to (4.26) the transformation from the base to the end-effector is expressed as the product of each link transformation matrix, thus resolving the problem of *Geometric Direct Kinematics* for 6 and 7 DOF manipulators since the method followed is the same for both.

$$^0\mathbf{P} = {}^0\mathbf{T}_n \, {}^n\mathbf{P} \tag{4.28}$$

According to (4.28), one can compute any point coordinates relative to the base frame, when the position relative to the end-effector is known.

4.3.2 Geometric Inverse Kinematics

When we turn to the *Geometric Inverse Kinematics* problem, the solution depends on the non-linearity of the equations that may lead to multiple solutions, infinite number of solutions[6], or no valid solutions when all the solutions exceed the at least one joint limit. The existence of a set of joint variables that allow the end-effector to reach a determined position and orientation, depends if the target falls within the dextrous workspace.

Unlike *Geometric Direct Kinematics*, the inverse kinematics solutions for 6 and 7 DOF manipulators are different. We start by addressing the solution for 6 DOF manipulators, and then present the differences to the method used for the 7 DOF.

6 DOF To make our method easier we decouple the manipulator in two parts, from the base frame {B} to the wrist {W}, which we call arm and is mainly responsible for positioning. And from the wrist {W} to the end of the robot's last link {T} called forearm, where the orientation is defined. The algorithm proposed uses a geometric approach to determine the lower arm joint values, and an algebraic method to determine the last three joint variables. It can be divided in five elementary steps,

1. Calculate the wrist position {W};
2. Solve the inverse kinematics for the arm, $(\theta_1, \theta_2, \theta_3)$ (see figure 4.2);
3. Compute ${}^0\mathbf{R}_3(\theta_1, \theta_2, \theta_3)$;
4. Find ${}^3\mathbf{R}_6(\theta_4, \theta_5, \theta_6)$ (figure 4.4) from, ${}^3\mathbf{R}_6 = {}^0\mathbf{R}_3^{-1}\ {}^0\mathbf{R}_6$;
5. Solve the inverse kinematics for the spherical wrist.

The user inputs the desired position and orientation for the end-effector as $\mathbf{x_e} = (p_{e,x}, p_{e,y}, p_{e,z}, \gamma_{e,\theta}, \gamma_{e,\psi}, \gamma_{e,\phi})$, according to (4.4). Knowing $\gamma_\mathbf{e}$, one can compute Roll-Pitch-Yaw matrix (4.16). The wrist position, denoted as $\mathbf{p_w} = (p_{w,x}, p_{w,y}, p_{w,z})$,

[6] **subsection 4.3.2** In cases of kinematically redundant manipulators.

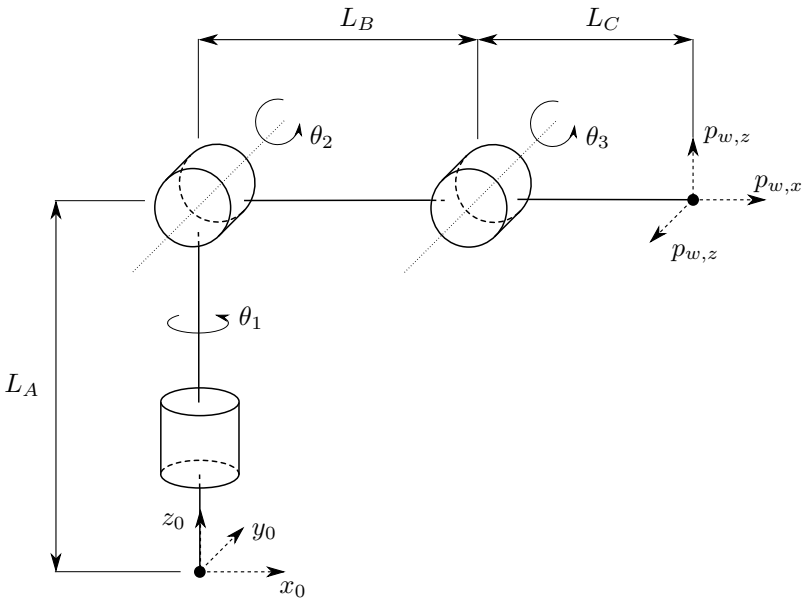


Figure 4.2: Anthropormorphic arm.

is calculated by subtracting the length of the wrist to the end-effector (L_w) (figure 4.4) from the final position specified $\mathbf{p_e}$, taking into account the orientation, $\hat{x}_{RPY}$.

$$\mathbf{p_w} = \mathbf{p_e} - L_w \; . \; \hat{z}_{RPY} \tag{4.29}$$

Then we check if the wrist position falls within range of the arm. Being L_B the length from the shoulder to the elbow and L_C the length between the elbow and the wrist, $\mathbf{p_e}$ can only be reached with $\gamma_\mathbf{e}$ orientation if[7],

$$|L_B - L_C| \leq \|\mathbf{p_w}\| \leq L_B + L_C \tag{4.30}$$

If this condition is verified, we proceed to the next step of the algorithm to calculate $(\theta_1, \theta_2, \theta_3)$. As we are dealing with a standard anthropomorphic lower arm, the only joint capable of positioning itself in the xy plan is θ_1 (figure 4.2) and so it is calculated as follows,

$$\theta_1 = \begin{cases} \arctan_2(p_{w,y}, p_{w,x}) & \text{if } p_{w,y} \neq 0 \\ 0 & \text{if } p_{w,y} = 0 \end{cases} \tag{4.31}$$

[7]**subsection 4.3.2** $\mathbf{p_e}$, must consider the distance from the base frame to the center of the shoulder (L_A).

We can simplify the problem of calculating the θ_2 and θ_3, by reducing the problem from three to two dimensions using a plane formed by the z axis and the direction of θ_1 set on the xy plane, figure 4.3.

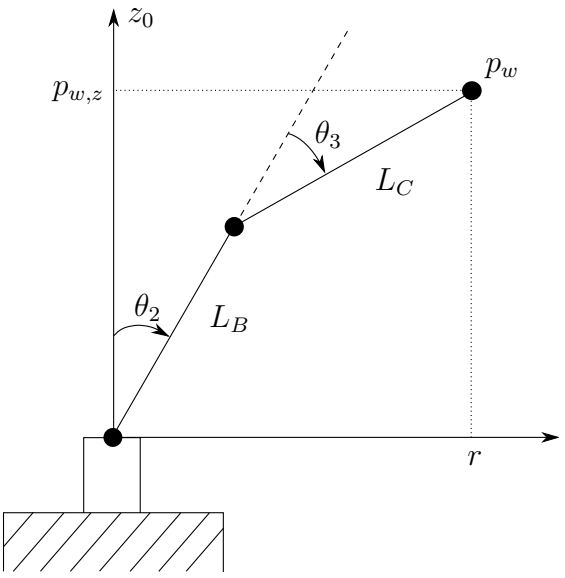


Figure 4.3: Plane projection formed by links 2 and 3.

Considering $r = \sqrt{{p_{w,x}}^2 + {p_{w,y}}^2}$, we easily prove that,

$$\begin{cases} r = L_B \sin\theta_2 + L_C \sin\left(\theta_2 + \theta_3\right) \\ p_{w,z} = L_B \cos\theta_2 + L_C \cos\left(\theta_2 + \theta_3\right) \end{cases} \tag{4.32}$$

which allow us to solve this problem as a planar 2 DOF arm. With both expressions we reach the following equation,

$${p_{w,x}}^2 + {p_{w,y}}^2 + {p_{w,z}}^2 = {L_B}^2 + {L_C}^2 + 2\ L_B\ L_C \cos\theta_3 \tag{4.33}$$

from which,

$$\cos\theta_3 = \frac{{p_{w,x}}^2 + {p_{w,y}}^2 + {p_{w,z}}^2 - {L_B}^2 - {L_C}^2}{2\ L_B\ L_C} \tag{4.34}$$

and by the *Pythagorean trigonometric identity*

$$\sin\theta_3 = \pm\sqrt{1 - {\cos\theta_3}^2} \tag{4.35}$$

With the cosine and sine of θ_3 we can finally calculate θ_3 using,

$$\theta_3 = \arctan_2(\sin\theta_3, \cos\theta_3) \tag{4.36}$$

As we can see in (4.35), we can have two solutions for the same position. Knowing the value of θ_3 we can determine θ_2 by means of,

$$\theta_2 = \arctan_2(r, p_{w,z}) - \arctan_2(L_C \sin\theta_3, L_B + L_C \cos\theta_3) \tag{4.37}$$

After solving the inverse kinematics for $(\theta_1, \theta_2, \theta_3)$ we proceed to compute ${}^0\mathbf{R}_3$, using the homogeneous transforms from geometric direct kinematics. It is now possible to calculate ${}^3\mathbf{R}_6$ through,

$${}^3\mathbf{R}_6 = {}^0\mathbf{R}_3^T \; {}^0\mathbf{R}_6 \tag{4.38}$$

as both ${}^0\mathbf{R}_3^T$ and ${}^0\mathbf{R}_6$ are known, being the latter the Roll-Pitch-Yaw matrix computed earlier with the orientation parameters $\gamma_\mathbf{e}$. To simplify the notation we denote the ${}^3\mathbf{R}_6$ matrix as,

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.39}$$

From here, and due to the spherical wrist properties we directly compute θ_5 from,

$$\theta_5 = \arctan_2(\pm\sqrt{1-(r_{23})^2}, r_{23}) \tag{4.40}$$

which similarly to the elbow (4.35) has two solutions, which allow the manipulator to reach the same position and orientation with 2 different possible configurations. In the end, we may reach the same wrist position with 4 possible configurations, hence the need of a decision criteria. This criteria must firstly consider the limits of each joint, since they can restrict the options to one configuration over the other. However if more than one solution is possible, the spatial distribution of the arm or the strain applied to each joint can bias the decision.

A particularity of the spherical wrist, is that it reaches an infinite number of solutions if $\theta_5 \in \{0, \pi\}$, as a result of the collinear axes of joints 4 and 6. To solve

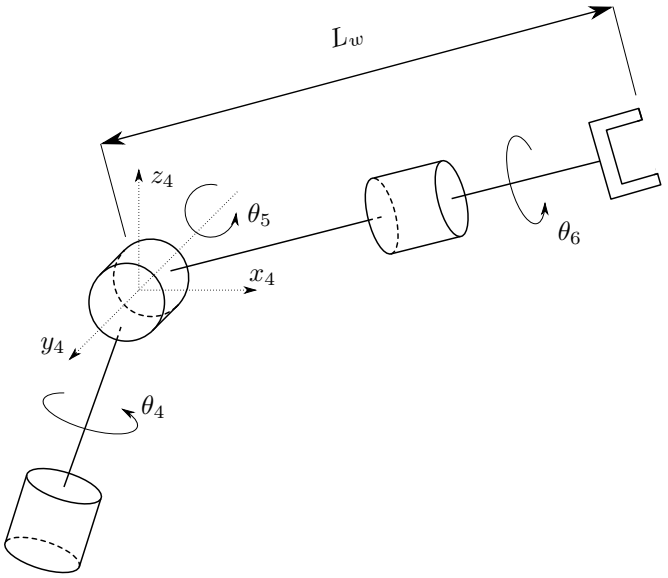


Figure 4.4: Spherical wrist.

this cases of singularity we chose to fix the 4th joint and only rotate the 6th, what results in

$$\begin{cases} \theta_4 = 0 \quad \wedge \quad \theta_6 = \arctan_2(r_{12}, r_{32}) & \text{if } \theta_5 = 0 \\ \theta_4 = 0 \quad \wedge \quad \theta_6 = -\arctan_2(-r_{12}, -r_{32}) & \text{if } \theta_5 = \pi \end{cases} \tag{4.41}$$

Otherwise, θ_4 and θ_6 can be computed by,

$$\begin{cases} \theta_4 = \arctan_2(r_{33}, -r_{13}) \\ \theta_6 = \arctan_2(-r_{22}, -r_{21}). \end{cases} \tag{4.42}$$

With this last step, we have calculated the set of joint variables $\mathbf{q}$, corresponding to a specified end-effector position and orientation as (4.4), thus solving the problem of *Geometric Inverse Kinematics* for 6 DOF manipulators.

7 DOF The method to compute the geometric inverse kinematics in a 7 DOF is a bit more complicated as we want to control seven joints $\mathbf{q} = [\theta_1, \theta_2, \cdots, \theta_7]^T$ based in six position and orientation coordinates (4.2). This leads to a case of redundancy since we have more variables than equations. To develop the geometric inverse kinematic equations for the 7 DOF robotic manipulator, we resorted to the

method proposed by Eliana in her PhD thesis [91]. To simplify the notation the manipulator distances will be displayed as,

- L_1, Base to Shoulder;
- L_2, Shoulder to Elbow;
- L_3, Elbow to Wrist;
- L_4, Wrist to End-effector.

Similarly to 6 DOF manipulators, we can calculate the wrist position by subtracting the wrist length from the target position along the desired orientation, (4.29). Also the joint variables of the arm can be computed if we have the shoulder position as we already know the base coordinates. The elbow that connects the arm to the forearm has however no restrictions, which means that[8] it can orbit perpendicularly to the shoulder-wrist line without changing the end-effector position or orientation.

It was added an extra manipulator configuration constraint that defines the minimum angle between the Shoulder-Elbow-Wrist plan and the floor, which we will henceforth call α, (figure 4.5).

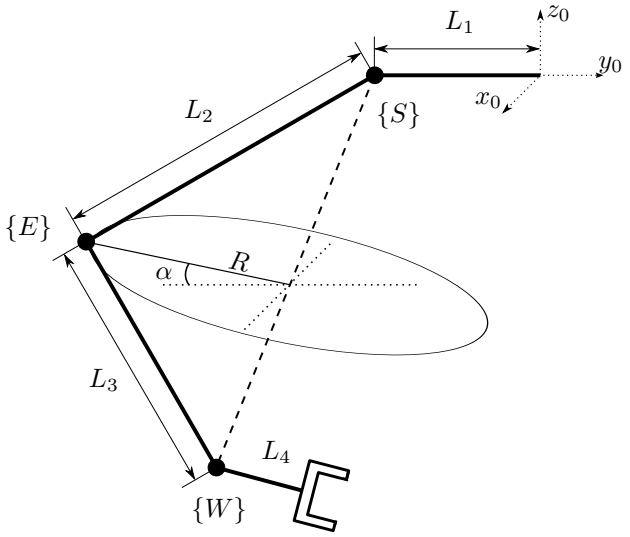


Figure 4.5: Elbow redudancy.

Thus, the geometric inverse kinematics solution given in (4.4), now has 7 variables to 7 equations and takes the new form,

[8] **subsection 4.3.2** unless the target position requires for the manipulator to be fully stretched

$$\mathbf{q} = f^{-1}(\mathbf{x_e}, \alpha) \tag{4.43}$$

We start by calculating the wrist position using (4.29). Since the shoulder is connected to the base through the first link and considering the world frame to be represented by $x_0\ y_0\ z_0$ as in figure 4.5, the shoulder is located at $S = [0, -L_1, 0]^T$. With the angle $\left(\frac{\pi}{2} - (-\theta_4)\right)$ (figure 4.6), we can compute the distance from the shoulder to the wrist using the cosines law,

$$L_{SW} = {L_2}^2 + {L_3}^2 - 2\ L_2\ L_3 \cos\left(\frac{\pi}{2} + \theta_4\right) \tag{4.44}$$

which leads to,

$$\theta_4 = \arcsin\left(\frac{L_{SW} - {L_2}^2 - {L_3}^2}{2\ L_2\ L_3}\right) \tag{4.45}$$

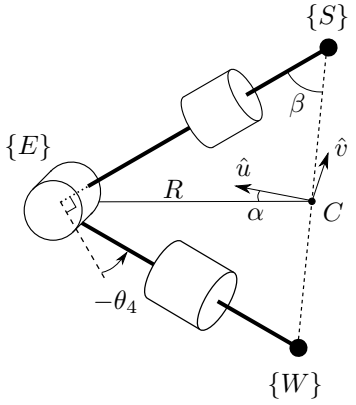


Figure 4.6: Elbow angle θ_4.

Having the value of the elbow angle computed, it can still take any orientation along the circular orbit depict in figure 4.5. Thus, the next step of the algorithm is to find the elbow position taking into account the angle α.

Using the cosines law once again it is possible to extrapolate the coordinates of the point C. We calculate the $L_2 \cos(\beta)$ (figure 4.5) which is the distance from the shoulder to C,

$$\overline{SC} = L_2 \cos(\beta) = \frac{L_{SW}^2 + L_2^2 - L_3^2}{2L_{SW}} \tag{4.46}$$

Considering $\hat{n}_{SW}$ to be the unit vector with the direction set from the shoulder to the wrist, C is calculated by adding $\overline{SC}$ along $\hat{n}_{SW}$ to the shoulder position,

$$C = S + \overline{SC} \cdot \hat{n}_{SW} \tag{4.47}$$

With coordinates of C and by the *Pythagorean trigonometric identity* we compute the radius (R) of possible elbow positions,

$$R = L_2\sqrt{1 - \cos^2(\beta)} \tag{4.48}$$

Next, to obtain the final elbow position as a function of α we will use some auxiliary vectors. Firstly, we define $\hat{u}$ as an unit vector normal to $\hat{n}_{SW}$ projected in the xy plan, or in other words, parallel to the ground. Then $\hat{v}$ is defined as the unit vector normal to $\hat{u}$ and $\hat{n}_{SW}$, which is obtained by,

$$\hat{v} = \hat{u} \times \hat{n}_{SW} \tag{4.49}$$

The elbow position is now computed by the following expression,

$$E = C + R(\hat{u}\cos(\alpha) - \hat{v}\sin(\alpha)) \tag{4.50}$$

With the elbow position we can now compute the matrix that defines the rotation between the base referential to the elbow referential. For this, we assume that the elbow rotates along $\hat{z}$, perpendicularly to the plan formed by the shoulder, elbow and wrist. Being $\hat{v}_{EW}$, $\hat{v}_{SW}$ and $\hat{v}_{CE}$ the unit vectors with the direction of elbow-wrist, shoulder-wrist and center-elbow, respectively. The frame axes are,

$$\begin{aligned}\hat{y}_4 &= \hat{v}_{EW} \\ \hat{z}_4 &= \hat{v}_{SW} \times \hat{v}_{CE} \\ \hat{x}_4 &= \hat{y}_4 \times \hat{z}_4\end{aligned}$$

Therefore we can express the rotation matrix from the base to the elbow as,

$${}^0\mathbf{R}_4 = \begin{bmatrix}\hat{x}_4 & \hat{y}_4 & \hat{z}_4\end{bmatrix} \tag{4.51}$$

As we know beforehand the value of θ_4 and thus the rotation ${}^3\mathbf{R}_4$, the matrix ${}^0\mathbf{R}_3$ is simply,

$$ {}^0\mathbf{R}_3 = {}^0\,\mathbf{R}_4\ ({}^3\mathbf{R}_4)^T \tag{4.52} $$

With the ${}^0\mathbf{R}_3$ matrix, it is possible to extrapolate the θ_1, θ_2 and θ_3 using the algebraic method presented for the spherical wrist in the 6 DOF manipulators, (see equations (4.40), (4.41) and (4.42)).

Finally, for the last three joints the algorithm repeats as we know the rotation matrices, ${}^0\mathbf{R}_4$ and ${}^0\mathbf{R}_7$, so it is possible to find ${}^4\mathbf{R}_7$ by,

$$ {}^4\mathbf{R}_7 = ({}^0\mathbf{R}_4)^T\ ({}^0\mathbf{R}_7) \tag{4.53} $$

Once again it is used the algebraic method from the spherical wrist to compute the angle of the last joints.

We have thus solved, problem of *Geometric Inverse Kinematics*, for both 6 and 7 DOF manipulators as proposed.

4.4 Differential Kinematics

The differential kinematics equations establish the relation between the *Joint* and *Cartesian space* velocities, forces or torques. As in the previous section, we will start by explaining how to compute the *Differential Direct Kinematics*, whose algorithm is similar for both 6 and 7 DOF manipulators, and then talk about *Differential Inverse Kinematics*.

4.4.1 Differential Direct Kinematics

The functions that return the end-effector linear and angular velocities when given the joint velocities are represented as a matrix named *Geometric Jacobian*, which depends on the manipulator configuration. It is however possible to compute the Jacobian matrix resorting to the differentiation of the geometric direct kinematics functions relative to the joint variables, if the end-effector position and orientation

are described by a minimal representation, such as the in Denavit-Hartenberg convention. The Jacobian matrix obtained this way differs from the *Geometric Jacobian* and is known as the *Analytical Jacobian*, which shall be denoted as $\mathbf{J}(q)$ or simply $\mathbf{J}$. This matrix is particularly helpful to analyze the manipulator redundancy, to find singularities and to correlate not only the velocities but also the forces and torques in the *Joint* and *Cartesian Space*.

In analogy to the geometric kinematics problem, the end-effector pose instead of being represented by a position and orientation is, in the differential kinematics context, expressed by a linear ($\dot{\mathbf{p}}_\mathbf{e}$) and an angular velocity ($\omega_\mathbf{e}$),

$$\mathbf{v_e} = \begin{bmatrix} \dot{\mathbf{p}}_\mathbf{e} \\ \omega_\mathbf{e} \end{bmatrix} = \mathbf{J}(q) \ . \ \dot{\mathbf{q}} \tag{4.54}$$

The Jacobian matrix, can be split into two (3 × n) sub matrices[9] that provide the relation between each joint velocity and the end-effector linear ($\mathbf{J}_P$) and angular velocity ($\mathbf{J}_O$) components, respectively,

$$\mathbf{J}(q) = \begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix} . \tag{4.55}$$

Acknowledging that the robotic manipulators used are anthropomorphic and thus have only revolute and cylindrical joints, the full *Analytic Jacobian* (with both sub-matrices) can be obtain from the expression [52],

$${}^0\mathbf{J}(q)_n = \begin{bmatrix} {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{1} \times ({}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{n} - {}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{1}) & {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{2} \times ({}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{n} - {}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{2}) & \cdots & {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{n} \times ({}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{n} - {}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{n}) \\ {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{1} & {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{2} & \cdots & {}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{n} \end{bmatrix} \tag{4.56}$$

where ${}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{n}$ is the (3×1) position vector of the transformation matrix from the base of the manipulator to the end-effector, ${}^\mathbf{0}\hat{\mathbf{p}}_\mathbf{i}$ (3×1) position vector and ${}^\mathbf{0}\hat{\mathbf{z}}_\mathbf{i}$ (3×1) rotation vector component from the partial premultiplying transformation matrix from the robot base to each joint given by [92],

[9] **subsection 4.4.1** being n, the number of joints

$$
{}^{0}\mathbf{T}_i = \prod_{j=1}^{i} {}^{j-1}\mathbf{T}_j = \left[\begin{array}{cccc|c} & & & & \\ {}^{0}\hat{\mathbf{x}}_\mathbf{i} & {}^{0}\hat{\mathbf{y}}_\mathbf{i} & {}^{0}\hat{\mathbf{z}}_\mathbf{i} & & {}^{0}\hat{\mathbf{p}}_\mathbf{i} \\ & & & & \\ \hline 0 & 0 & 0 & & 1 \end{array}\right] \tag{4.57}
$$

With the Jacobian matrix defined, and the (4.54), one can compute the end-effector linear and angular velocities, thus solving the *Differential Direct Kinematics problem* for either 6 or 7 DOF manipulators.

4.4.2 Differential Inverse Kinematics

As seen in the geometric inverse kinematics problem, there is a complex and non-linear relation between the *Joint* and *Cartesian space* variables due to the manipulator configuration, redundancy or specific poses. However, when we turn to differential kinematics, the equation represents a linear mapping relating both spaces that varies with the current configuration [52].

6 DOF According to [52], given that the number of DOF is equal to the number of operational space variables needed to execute a certain task (in our case 6 space variables, thus 6 DOF), the differential inverse kinematics can be simply solved by using the inverse Jacobian matrix,

$$
\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\ \mathbf{v_e} \tag{4.58}
$$

As the surgical tasks assign to the robot require the control of the manipulator along all position and orientation coordinates, we can use the equation 4.58 to solve the differential inverse kinematics problem for the 6 DOF robots.

7 DOF In the case of the 7 DOF manipulator, the Jacobian is a (6×7) matrix and thus is not directly invertible. Additionally, there is an infinite number of solutions or joint velocity profiles for the same operational space requirements. Thus, the approach suggested by [52], involves the formulation of a linear optimization problem for the joint velocities, which should work as a secondary objective.

Being $\mathbf{J}$ a regular Jacobian matrix where the number of columns (m) is greater than the number of rows (n), it will be used the right inverse matrix ($\mathbf{J}_r$) so that,

$$\mathbf{J}\ \mathbf{J}_r = \mathbf{I}_m \tag{4.59}$$

where $\mathbf{I}_m$ is the identity matrix. If the $(m \times n)$ Jacobian matrix $\mathbf{J}_r$ has rank (m), the right pseudo-inverse can be calculated as,

$$\mathbf{J}_r^{\dagger} = \mathbf{J}^T(\mathbf{J}\ \mathbf{J}^T)^{-1} \tag{4.60}$$

Such solution minimizes the norm of joint velocities. Thus the equation 4.6, for a 7 DOF manipulator can be written as,

$$\dot{\mathbf{q}} = \mathbf{J}_r^{\dagger}(\mathbf{q})\ \mathbf{v_e} \tag{4.61}$$

Nevertheless, one can also add a further linear optimization problem as stated before by making use of the manipulator redundancy. Therefore,

$$\dot{\mathbf{q}} = \mathbf{J}_r^{\dagger}(\mathbf{q})\ \mathbf{v_e} + \mathbf{J}_{r,null}^{\dagger}(\mathbf{q})\ \dot{\mathbf{q}}_\mathbf{0} \tag{4.62}$$

is also a solution for the differential inverse kinematics problem, being the $\dot{\mathbf{q}}_\mathbf{0}$ a vector of arbitrary joint velocities (the secondary objective). It is possible to guarantee that the manipulator would move at the desired linear and angular velocities and still endow behaviors like obstacle avoidance, by making use of the *null* space[10].

The $\mathbf{J}_{r,null}^{\dagger}(\mathbf{q})$ matrix is a projection of the $\mathbf{J}_r^{\dagger}(q)$, in the *null* space and can be computed as,

$$\mathbf{J}_{r,null}^{\dagger}(\mathbf{q}) = (\mathbf{I} - \mathbf{J}_r^{\dagger}(\mathbf{q})\mathbf{J}(\mathbf{q})) \tag{4.63}$$

The flexibility provided by the extra DOF, in the context of our problem, can be used to guarantee that the robot posture does not collide with intraoperative

[10]**subsection 4.4.2** a Jacobian subspace, of joint velocities that reproduce no variation in the end-effector velocity.

elements, personnel or machinery. Despite secondary, this feature can greatly add to the acceptance of a robotic system by the operating team.

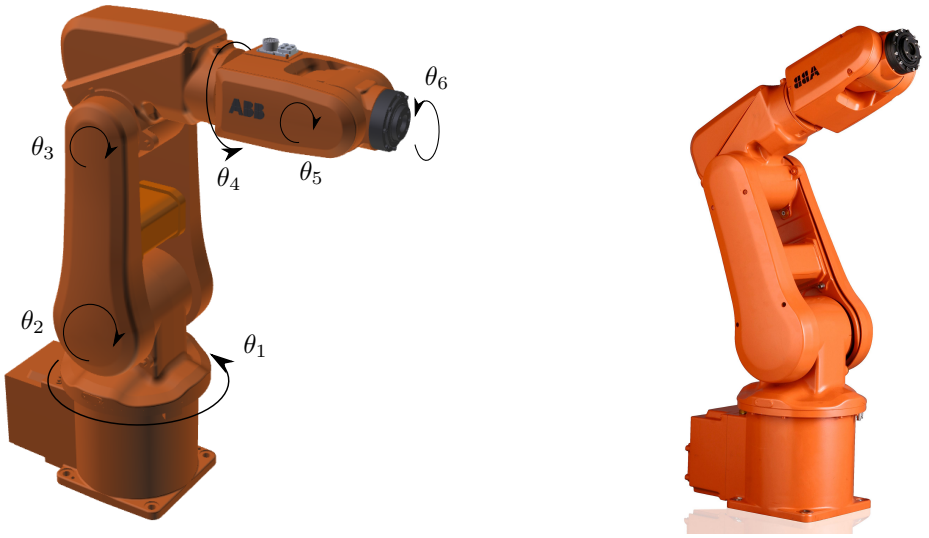
4.5 Selected Robots Specifications

After presenting the fundamentals of manipulator kinematics for 6 and 7 DOF manipulators, in this section we aim to describe the specific characteristics for each of the selected robotic systems, along with their limitations. The information and images of each manipulator are retrieved from the correspondent manufacturers. Although it might seem repetitive, we chose to exhibit each robotic manipulator in separated subsections to facilitate a quick access to the information regarding each system.

4.5.1 ABB IRB 120

The ABB IRB 120 is the smallest robot of the ABB line and is recommended for material handling and assembly applications. Having a weight of only 25 kg and with a reach of 580 mm this manipulator excels in flexibility and efficiency specially within limited spaces, like the available workspace within the operating room. The modularity of the system, allows for the robot to be mounted in any angle, however due to the location of the available workspace and the space restrictions caused by the surgical lights at the ceiling, it was decided to mount the robots' in a mobile platform(cf. chapter 6).

Other captivating features of the manipulator are the smooth surfaces and the internally routed cables, which greatly helps in the robot's integration as it needs to be covered with sterile drape for each procedure. The manipulator light body is built in an aluminum structure that incorporates precise and reliable motors that further enhance the robot's agility and accuracy of movements. The robotic manipulator is controlled with the IRC5 ABB compact controller which provides increased accuracy and motion control previously only possible in larger installations. It adds to the robot's portability and mobility around the operating room,



(a) 3d model with each joint and correspondent rotations represented.

(b) Photo of the real manipulator taken from the manufacturer website.

Figure 4.7: ABB IRB 120 manipulator.

and could even be incorporated in the robot mobile platform base.

The robot can operate in environments from $5°C$ to $45°C$ and in conditions of humidity up to 95%. The robotic system includes safety and emergency stops, has supervision circuitry and is shielded for electromagnetic compatibility.

The ABB IRB 120 manipulator joint angles and velocity limits are presented in the table 4.1. Its initial posture and each joint rotation direction is depict in figure 4.7a. The real robot[11] body is presented in figure 4.7b.

Robot kinematics considers the geometry and spatial configuration of the robot, to describe relations between a chained joint action and its spatial implication, either in position, velocity or acceleration. The links are simply considered to be rigid bodies moved by means of actuated rotational or translational joints. Therefore, one can represent the whole structure of the robot through a simplified schematic (figure 4.8), in which, links are lines and joints are different symbols standing for cylindrical[12] or revolute[13], since all the joints in the Abb manipulator

[11]**subsection 4.5.1** For additional information regarding this manipulator please consult: `http://www.abb.com/product/seitp327/be2eef38406eaca4c125762000319182.aspx`.

[12]**subsection 4.5.1** Inverted triangles

[13]**subsection 4.5.1** Bold circle, concentric with a semi-circumference

Table 4.1: ABB IRB 120 manipulator limits.

Joints	Joint Limits (deg)	Velocity Limits (deg/s)	Acceleration Limits (deg/s^2)
θ_1	[-165, 165]	250	-
θ_2	[-110, 110]	250	-
θ_3	[-90, 70]	250	-
θ_4	[-160,160]	320	-
θ_5	[-120, 120]	320	-
θ_6	[-400, 400]	420	-

are rotative.

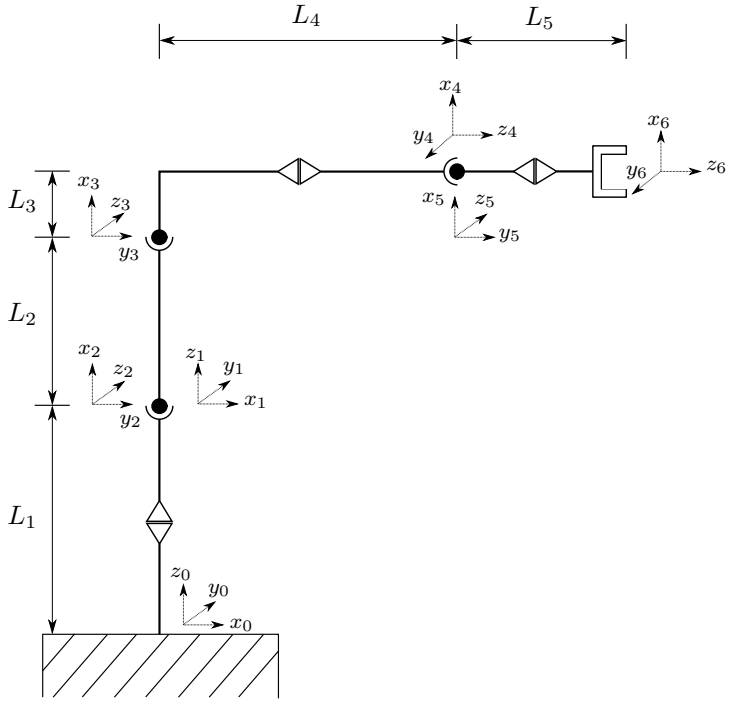


Figure 4.8: ABB IRB 120 manipulator at starting position with the frames assigned to each link, according to Denavit-Hartenberg convention.

The lengths depict in figure 4.8, are presented in table 4.2.

Table 4.3, presents the Denavit-Hartenberg parameters for the ABB manipulator. Each transformation presented in this very table is depict in the schematic figure 4.8. Due to space limitations each frame should be considered to be assigned

Table 4.2: ABB IRB 120 arm segments length.

Segments Length (mm)	Description
$L_1 = 290$	Distance from base frame to the 2^{nd} joint.
$L_2 = 270$	Distance from the 2^{nd} joint to the 3^{rd} joint.
$L_3 = 70$	Eccentricity from 3^{rd} to the 4^{th} joint along the base frame z-axis.
$L_4 = 302$	Distance from the 3^{rd} to the 4^{th} joint along the base frame x-axis.
$L_5 = 72$	Distance from the 4^{th} to the end-effector.

to the closest joint, or in case of the $0^{th} frame$ to the base and the $6^{th} frame$ to the end-effector. Having the Denavit-Hartenberg parameters, each transformation can be represented by a matrix that is computed according to 4.27 and the *Geometric Direct Kinematics* for this manipulator is simply the product of those matrices from the base to the end-effector.

Table 4.3: Denavit-Hartenberg parameters for the ABB IRB 120 manipulator.

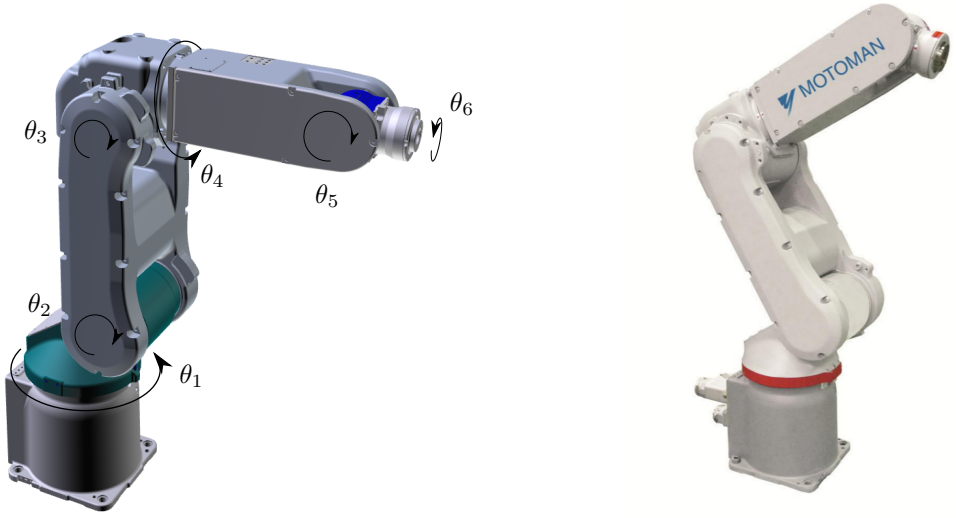
${}^{i}T_{i+1}$	α_{i-1} (deg)	a_{i-1} (mm)	θ_i (deg)	d_i (mm)
$\{0\} \rightarrow \{1\}$	0	0	θ_1	L_1
$\{1\} \rightarrow \{2\}$	-90	0	$\theta_2 - 90$	0
$\{2\} \rightarrow \{3\}$	0	L_2	θ_3	0
$\{3\} \rightarrow \{4\}$	-90	L_3	θ_4	L_4
$\{4\} \rightarrow \{5\}$	90	0	θ_5	0
$\{5\} \rightarrow \{6\}$	-90	0	θ_6	L_5

The *Geometric Inverse Kinematics* can be calculated with the algorithm presented in section 4.3 for the 6 DOF manipulators. There is however a slight difference caused by the offset between the 3^{rd} and $4^{th} joint$, which means that these two joints are not connected by a straight line. The solution outlined, involved using an auxiliary line to directly connect each joint instead of the actual link layout.

For the *Differential Kinematics*, the only information we need from the manipulator are the Denavit-Hartenberg parameters here displayed. The algorithm and further explanation are found in the previous section 4.4.

4.5.2 Motoman MH5

The MH5 Motoman manipulator, similarly to the previous robot from ABB has a compact design and is aimed towards high performance material handling, packing, assembly among other tasks within a restricted floor space and workspace. Focusing in precision and dexterity, this robot has a maximum payload capacity of $5kg$, but on the other hand it can easily achieving repeatability results below $\pm 0.02mm$.



(a) 3d model with each joint and correspondent rotations represented.

(b) Photo of the real manipulator taken from the manufacturer website.

Figure 4.9: Motoman MH5 manipulator.

To improve the robot's repeatability and steadiness, it can adapt its performance to the load, which has effective results in payloads inferior to $1kg$. Similarly to ABB it can be mounted in the floor, walls or the ceiling, but once again, the operating room disposition and space limitations will force us to fix it in a mobile platform. It can operate at temperatures ranging from $0°C$ to $45°C$ and

Table 4.4: Motoman MH5 manipulator limits.

Joints	Joint Limits (deg)	Velocity Limits (deg/s)	Acceleration Limits (deg/s^2)
θ_1	[-170, 170]	376	-
θ_2	[-65, 150]	350	-
θ_3	[-136, 255]	400	-
θ_4	[-190,190]	450	-
θ_5	[-125, 125]	450	-
θ_6	[-360, 360]	720	-

at a maximum relative humidity of 90%.

The manipulator has internally routed cables and hoses to maximize the system portability and reliability, and has also built-in collision-avoidance routines. It comes with a small size controller DXM100, certified and compliant to internationally recognized safety standards, which uses a programmable, user-friendly interface. The controller's interface also incorporates straightforward commands that allow the user to easily start, hold, stop, teach, play, or execute emergency routines. After contacting the Yaskawa Motoman representatives we found on one of the most desired features for our project, that is the floating mode, or backdriving trait. Such characteristic is exclusive of this manipulator from the selected set. It would certainly be a major contribute for the acceptance of a robotic system within the medical community as the surgeons should be able to freely move the manipulator by hand to the proximity of the entry point, from where it could precisely adjust to the closest pre-planned trajectory.

Joint, velocity and acceleration limits are found in table 4.4. The initial manipulator posture is depict in 4.10, and the specific arm segment lengths can be found in table 4.5.

The *Geometric Direct Kinematics* are computed by multiplying the transformation matrices generated from the Denavit-Hartenberg parameters (see matrix 4.27) from the base to the end-effector. Similarly to the ABB manipulator, the

Table 4.5: Motoman MH5 arm segments length.

Segments Length (mm)	Description
$L_1 = 330$	Distance from base frame to the 2^{nd} joint.
$L_2 = 88$	Eccentricity from base to the 2^{nd} joint along the base frame x-axis.
$L_3 = 310$	Distance from the 2^{nd} joint to the 3^{rd} joint.
$L_4 = 40$	Eccentricity from 3^{rd} to the 4^{th} joint along the base frame z-axis.
$L_5 = 305$	Distance from the 3^{rd} to the 4^{th} joint along the base frame x-axis.
$L_6 = 86.5$	Distance from the 4^{th} to the end-effector.

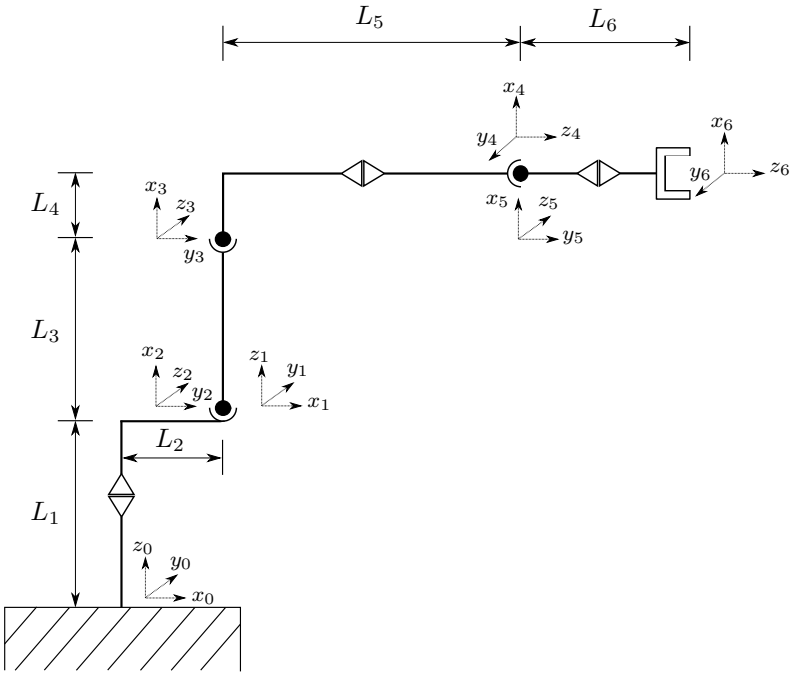


Figure 4.10: Motoman MH5 manipulator at home position with the frames assigned to each link, according to Denavit-Hartenberg convention.

MH5 model is endowed with an offset in the 3^{rd} to the $4^{th}joint$ link, moreover it has an additional eccentricity on the link from the 1^{st} to the $2^{nd}joint$. To guarantee that each link is represented by a straight line, so we can apply the algorithm for *Geometric Inverse Kinematics* drafted in section 4.3, for each eccentricity instead of the actual link layout we used the shortest line segment between both joints.

Table 4.6: Denavit-Hartenberg parameters for the Motoman MH5 arm.

${}^{i}T_{i+1}$	α_{i-1} (deg)	a_{i-1} (mm)	θ_i (deg)	d_i (mm)
$\{0\} \rightarrow \{1\}$	0	0	θ_1	L_1
$\{1\} \rightarrow \{2\}$	-90	L_2	$\theta_2 - 90$	0
$\{2\} \rightarrow \{3\}$	0	L_3	θ_3	0
$\{3\} \rightarrow \{4\}$	-90	L_4	θ_4	L_5
$\{4\} \rightarrow \{5\}$	90	0	θ_5	0
$\{5\} \rightarrow \{6\}$	-90	0	θ_6	L_6

As presented in section 4.4, the *Differential Kinematics* calculation need only the Denavit-Hartenberg parameters for this manipulator.

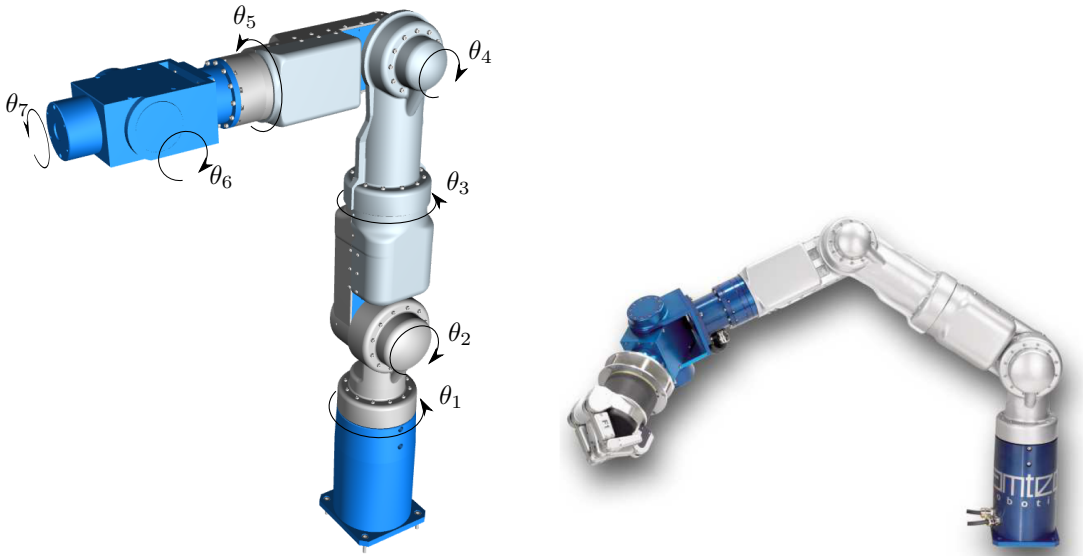
Having displayed the characteristics of the Motoman MH5 manipulator, the algorithms to achieve the kinematics relations are also fully described in section 4.4.

4.5.3 Schunk LightWeightArm II

The Light Weight Arm II by Schunk is a 7 DOF manipulator with a human-like architecture focused in having a light body with a flexible and mobile structure. The robotic controller is incorporated in the arm, thus relieving the need to transport a cumbersome external controller case.

The manipulator real appearance is shown in figure 4.11b, while in figure 4.11a it is displayed the initial position of each joint and the respective rotation direction used to compute the kinematic relations. The robot's cables and hoses are internally routed, in an integrated arm structure that weights only $12kg$ and has a maximum payload of $3kg$. The manipulator joint, velocity and acceleration limits can be found in table 4.7.

The schematic figure 4.12, depicts the manipulator initial position with the revolute and cylindrical joints disposed along the arm. The frames assigned to each *key-point*, either at revolute joints, at the manipulator base or at the end-



(a) 3d model with each joint and correspondent rotations represented.

(b) Photo of the real manipulator taken from the manufacturer website.

Figure 4.11: Schunk Light Weight Arm manipulator.

Table 4.7: Schunk Light Weight Arm manipulator limits.

Joints	Joint Limits (deg)	Velocity Limits (deg/s)	Acceleration Limits (deg/s^2)
θ_1	[-165, 165]	52.2	208.8
θ_2	[-105, 91]	52.2	208.8
θ_3	[-165, 165]	52.2	208.8
θ_4	[-25,196]	41.2	164.8
θ_5	[-165, 165]	41.2	164.8
θ_5	[-120, 120]	240	960.0
θ_7	[-165, 165]	360	1440.0

effector were sequentially placed according to the Denavit-Hartenberg convention, whose parameters are displayed in table 4.9. Arm segment lengths, denoted as L_x, specific of the Schunk LWA II manipulator are presented in table 4.8.

Algorithms to compute both *Geometric Kinematics* and *Differential Kinematics* for 7 DOF manipulators with a similar structure similar to this robot are fully explained in sections 4.3 and 4.4.

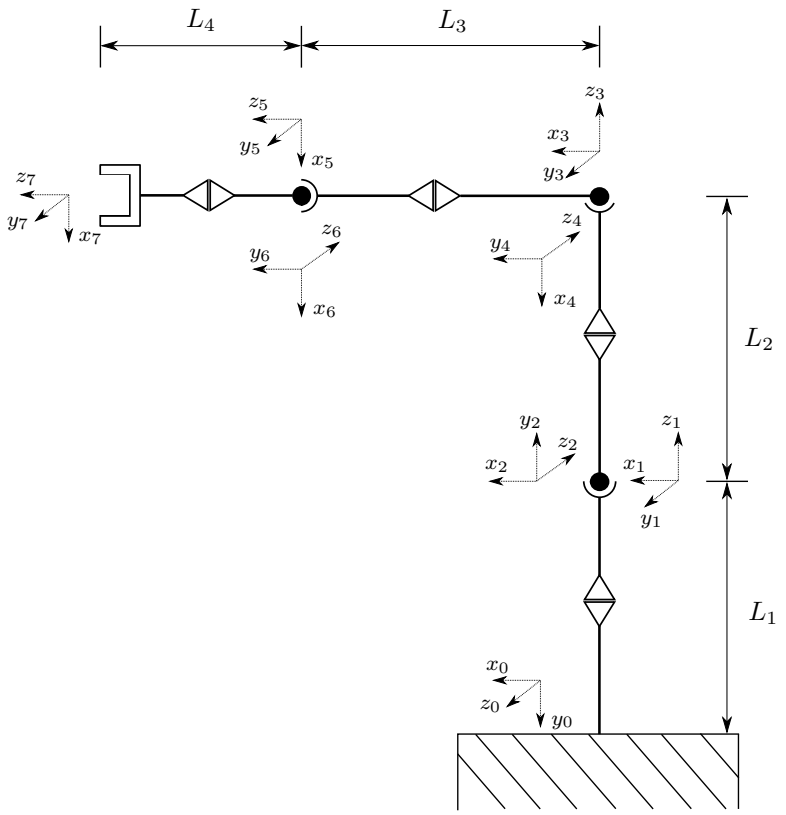


Figure 4.12: Schunk Light Weight Arm at starting position with the frames assigned to each link, according to Denavit-Hartenberg convention.

Table 4.8: Schunk Light Weight Arm segments length.

Segments Length (mm)	Description
$L_1 = 400$	Distance from base frame to the 2^{nd} joint.
$L_2 = 395$	Distance from the 2^{nd} joint to the 4^{th} joint.
$L_3 = 370$	Distance from the 4^{th} joint to the 6^{th} joint.
$L_4 = 95$	Distance from the 6^{th} joint to the end-effector.

Table 4.9: Denavit-Hartenberg parameters for the Schunk Light Weight Arm.

${}^{i}T_{i+1}$	α_{i-1} (deg)	a_{i-1} (mm)	θ_i (deg)	d_i (mm)
$\{0\} \rightarrow \{1\}$	90	0	θ_1	L_1
$\{1\} \rightarrow \{2\}$	90	0	θ_2	0
$\{2\} \rightarrow \{3\}$	-90	0	θ_3	L_2
$\{3\} \rightarrow \{4\}$	90	0	$\theta_4 - 90$	0
$\{4\} \rightarrow \{5\}$	-90	0	θ_5	L_3
$\{5\} \rightarrow \{6\}$	90	0	θ_6	0
$\{6\} \rightarrow \{7\}$	-90	0	θ_6	L_4

Chapter 5

Simulator

Being our purpose the development of a full robotic architecture and system towards stereotactic neurosurgery, it is rather difficult to create and build a product just from scratch and put it to use. Each step towards the objective brings new challenges, and the path is marked by a continuous struggle to solve upcoming problems. Therefore, regardless of how good the blueprints are, there must always be a test phase which usually reveals implementation issues [93].

Computer robotic simulators are programs that replicate a robotic system's behavior in an environment as close as possible to reality. It allows the user to thorough test the robotic emulated hardware/software or control application, make adjustments, devise new features, solve unforeseen problems or simply collect data in a controlled context without the risk of damaging expensive machinery or jeopardize the safety of users. Moreover, computer simulation is always accessible while real testing environment might not, the simulation runs are usually quicker than real ones and are also less demanding in terms of personnel and resources. The steady progresses in computer technology and processing power now permits state-of-the-art simulators to replicate not only complex system models but also physical and graphical environments with high accuracy [94] [95].

For these reasons and due to the inaccessibility to the desired robotic resources in light of this project objectives, simulation was a paramount element in devising a solution, which explains the importance given to it along this dissertation. The

simulator hereby presented can also be used for the robotic system end-users, to become accustomed to the robotic capabilities, movements and to the UI (User-Interface) used to maneuver it. It can also be used intraoperatively to predict and evaluate the robotic manipulator movements before executing any actions, (figure 5.1).

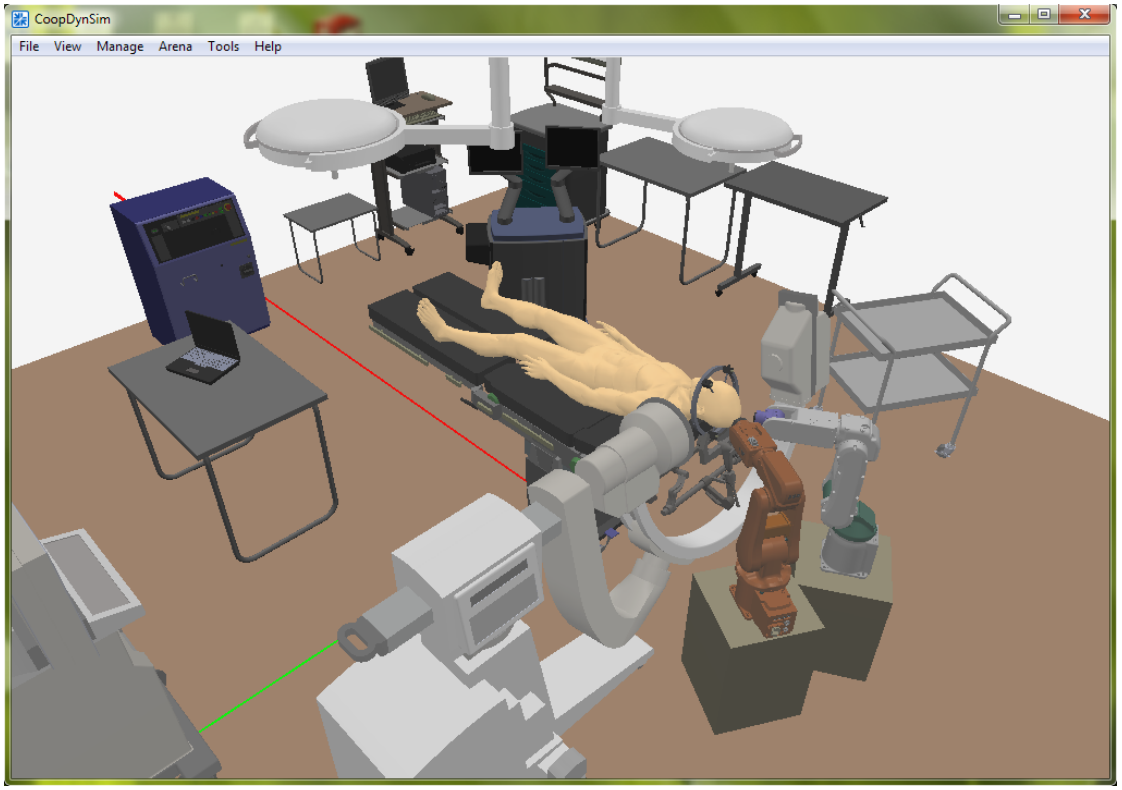


Figure 5.1: CoopDynSim interface window (operating room environment).

According to our work, the computer developed simulator is expected to replicate the behavior of a robotic platform, commanded through a high level controller via a third party communication layer. As stated in chapter 3, there are several variables that account to choose the most fitted robotic system and while some robotic features might be straightforwardly better or worse, other information such as joint limits, horizontal reach and degrees of freedom require practical testing, to understand their repercussions.

The simulator chosen was closest to a robotic simulator rather than a surgical simulator. In other words, it is meant to test how the robot will fit in the workspace, how it will interact with the user and to further elaborate on its motion

planning. Interaction with tissues, surface compliance, forces exerted during the assigned tasks and physiological implications were not included in the objectives of the sought simulator.

5.1 CoopDynSim 3D robotics simulator

Upon evaluating the most renowned robotic simulator softwares, we decided for the *CoopDynSim* 3D robotics simulator [96]. The thought process behind this selection had as a primarily focus the ability to emulate specific environments (operating room) and robotic systems. We sought an *open source* software, flexible enough to add and edit custom equipment and implement specific manipulator functionalities (adapted to DBS surgical procedure).

Furthermore, the simulator should have physical and graphical representations of a tridimensional world with several objects and robotic manipulators. There are several solutions with both graphical and physical tridimensional representations like *Player-Stage-Gazebo*, *USARSim*, *SimRobot*, *Microsoft Robotics Developer Studio* and *Webots*. However, we sought a simulator either free or available at our laboratory, to which we were familiar with. This way we can avoid further commitment to a software that would not meet our ultimate goals and also shorten the learning curve associated to working with the new tool. For additional information regarding available robotic simulators please consult [96] [97] [98].

Upon deciding for the *CoopDynSim* home made simulator, it was necessary to become acquainted with the software, understand how was the program structured, its basic functionalities and finally how could it be customized to include our robots and environments. We will now explain what was done previous to the developed work and the basic structure of the simulator which was somewhat respected throughout the upgrades.

CoopDynSim, or in its extended version Cooperative Dynamics Simulator, is a robotics simulator developed in C++, originally conceived to emulate multiple mobile robots or teams of robots in a world with basic obstacle and targets, (figure 5.2). As stated previously each element comprehended in the simulator has a

graphical representation rendered using OpenGL [99], and a physical counterpart emulated with Newton Game Dynamics engine [100]. Being a software developed in the laboratory from scratch, it has a slightly limited user interface and has a strict number of included robot entities large enough to cover the investigation needs. However and at the same time, its flexibility to accommodate custom made components, worlds and third party control applications is unmatched.

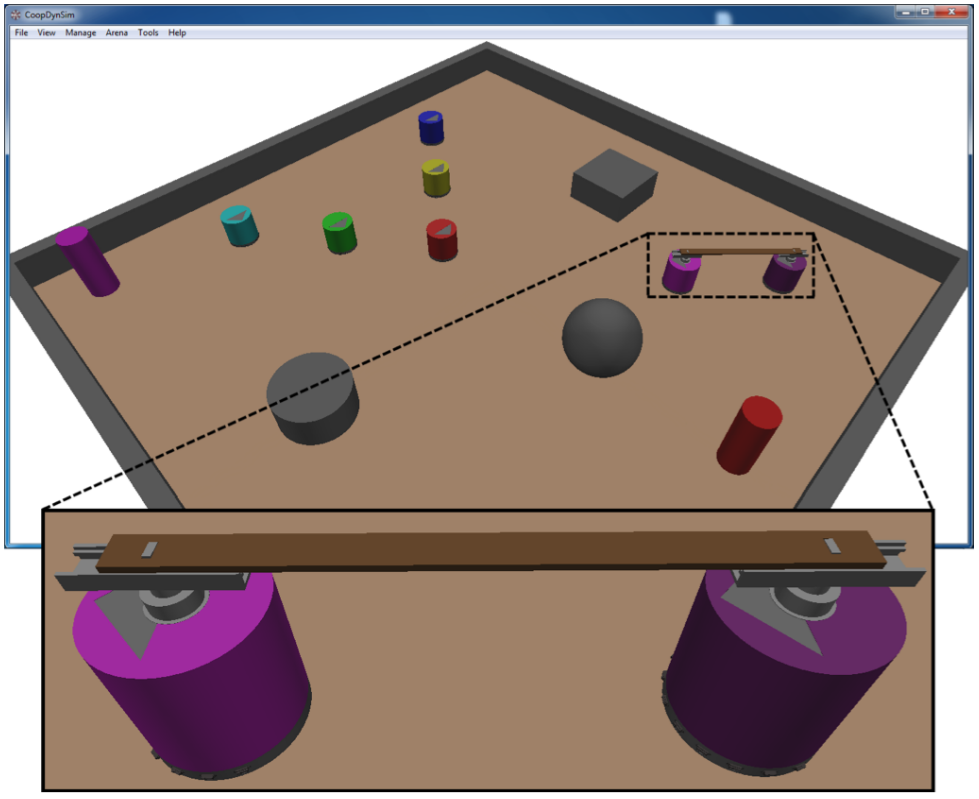


Figure 5.2: CoopDynSim interface window (mobile robotic simulator).

The simulator basic architecture can be partitioned in six main cores schematically related as in figure 5.3.

Starting with **Dialog**, these classes comprise functionalities and properties that form the UI. It was created so that the user could easily setup a world with different objects, targets or robots in specific positions and orientations, and then update or remove them at will. The main UI window (figure 5.1 or 5.2) displays the world and allows the user to navigate in all 3 dimensions. However, this window does not establish any interaction between the user and the virtual world other

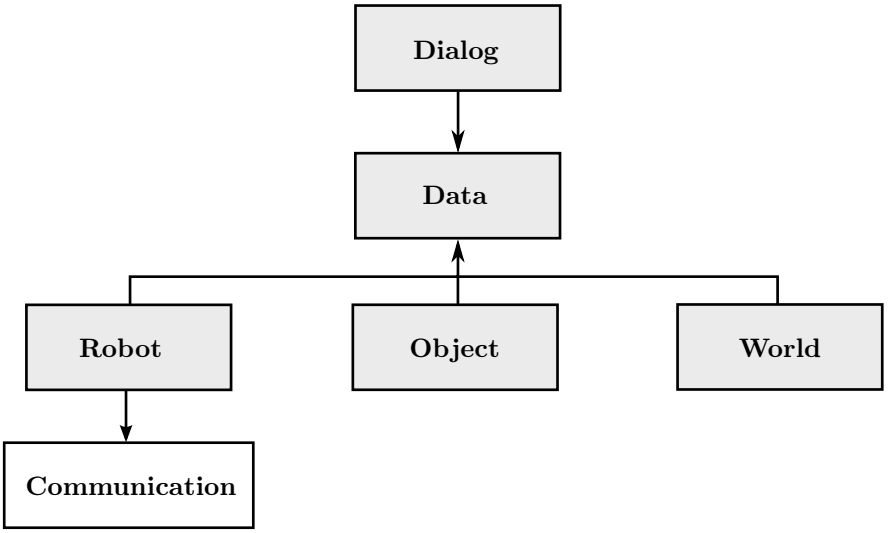


Figure 5.3: Diagram of *CoopDynSim* basic architecture.

then visualization. It is possible to simulate a world by setting each entity[1] to its assigned position and orientation, or by importing script wise files containing the information about each object, target and robot. The simulator input stream recognizes two types of script files:

1. ***.world** files - specify floor dimensions and each object type, dimension, position, orientation, color and mass;
2. ***.scenario** files - sets up a testing environment by specifying targets with designated colors and positions; robots with a type, name, position, orientation and color respectively; and the world where the simulation will take place (associated ***.world** file).

Additionally the UI allows the user to toggle several view options either to track the robot's path, see the world axes, display the physical contours of each element or to show the distance ray of robot embedded sensors.

However, the dialog classes only establish the bridge between the user intentions and the implemented code. The code that represents and manages all the objects within the simulator is the **Data** class, which stands as the *CoopDynSim* main core. As the user presses any button in the UI, an associated function from Data

[1] **section 5.1** object, target or robot

is called. Insert or remove functions call the specific entity's class constructor or destructor respectively, while update functions call entity's class methods to change its location. Data also includes functions to select the virtual world for simulation, to manage the graphic/physic threads access to entities and even to set the graphics refresh cycle as it calls the draw functions for all inserted entities.

The **World** and all incorporated entities can be break down to a group of **Objects** with a 3D graphical representation created from a set of vertices, normals, materials, etc. and a physical associated shape either represented by elemental entities (box, cylinder or sphere) or a compound of entities. A physical representation is associated to a mass, mass distribution, friction coefficients and other physical properties. When an object is created in the simulator, it is associated to a memory block that shares both graphic and physic information, which is accessed in an intercalary fashion, (figure 5.4).

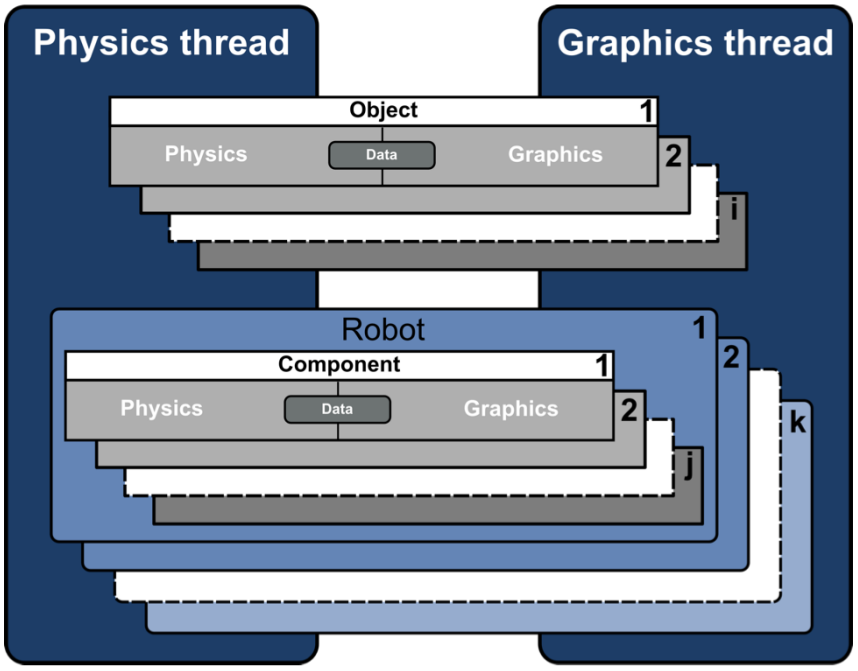


Figure 5.4: Object's properties in the simulated world.

Each *CoopDynSim* thread running either physics, graphics or the UI window threads are independent and managed separately. The graphics thread has an update cycle of $100ms$ while the physics thread has an update cycle of approximately

$10ms$, since it requires a higher refresh rate. However, both values can be adjusted to meet the requirements sought for simulation.

Since all entities can be separated in several objects, also the included **Robot** platforms are a set of associated objects grouped rigidly or linked through joint connections that can be actuated at each physics thread update. Being the initial focus of this simulator mobile robotics, the virtual robots emulated were created as close as possible to the real ones in terms of dimensions, weights, actuators, wheels and sensors. The only difference lied in the vision system, which instead of visually identifying targets through a camera input, the vision simulated module simply returns the target's position.

Similarly to other threads, also mobile robot threads are updated regularly in an interval defined by the developer regardless of having or not a new **Communication** event. Each robot instantiated is assigned to a new thread responsible to update the robot's modules, according to communication received directives. Robot classes are implemented as close as possible to their real forms, which means that its interface follows a server-client model, communicating via a middleware-layer. This architecture implies that each robot's hardware module is associated to a server that communicates to a client responsible for managing the received information and commanding the respective module, figure 5.5. The robot's name should be an unique char sequence as it will be used to identify the module's server ports.

Since the communication is simple and abstract, the user can directly transpose the implemented code from the simulator to a real robotic platform. The abstraction middleware was implemented using YARP[2] [101] an open-source software library oriented to humanoid robots. It provides a wrapper for the communication of each API to the control software based on a socket interface topology. To guarantee the integrity of the communication, it was devised a protocol that defines the communication rules between the control application and any hardware module, simulated or real. The message format is as follows, figure 5.6.

The *Error Code* is an integer that represents the success or not of the command

[2]**section 5.1** Yet Another Robot Platform

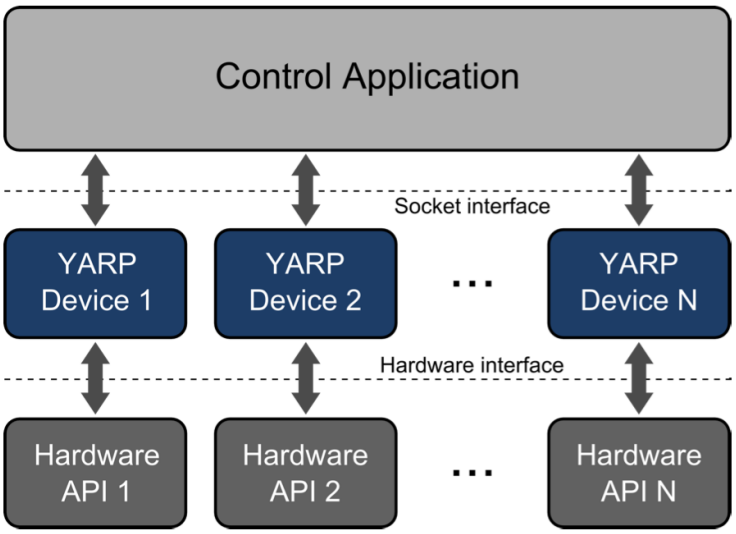


Figure 5.5: Middleware abstraction layer modularity.

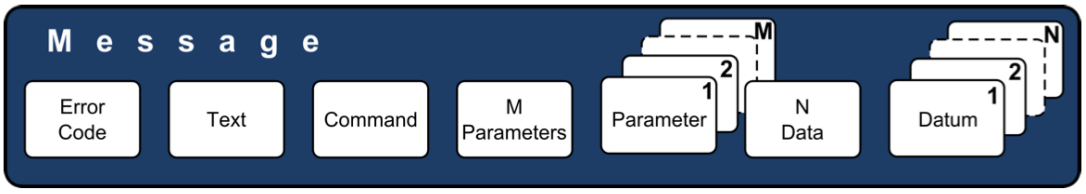


Figure 5.6: Communication message protocol.

sent, while the *Text* field carries a string type variable. The *Command* field is an unique key used for Remote Procedure Call either on the server or client application as specified by the developer. The message's numerical data is transmitted through a M number of integer *Parameters* and a N number of float *Datum*. This level of abstraction permits the communication to be accepted cross-platform using different languages, attribute that is taken into account and put to use in our project.

In addition to the standard features referred so far, the simulator also counts with a playback mode that allows the user to save in a log file the robot's position at each iteration as long as the selected scenario. Upon finishing the experiment, the user can replay or pause the saved simulation with a visual feedback. It provides a significant help in debugging a control application, specially if there are several robots or variables updating simultaneously.

5.2 Adaptation of CoopDynSim

Having as a starting point the *CoopDynSim* 3D robotic simulator oriented for mobile platforms, it was not ready yet to test robotic manipulators, even less to emulate an operating room environment. After selecting the robotic systems to be tested from the market supply (chapter 3), visiting a Neurosurgery operating room in Coimbra University Hospitals and attending to a real procedure in the same room, all the requirements were met to start upgrading the simulator and to devise a control application.

The objective is to create a virtual environment as close as possible to the real operating room, with all the equipments included in their right positions and dimensions, with a physical body associated. Therefore the developer can stipulate where to place the robot, how the robot should move inside the available workspace, understand possible motion restrictions and based on this data elaborate the best possible control solution. Having a visual feedback close to the operating room also turns the simulator more appealing and provides an extended sense of reality.

5.2.1 Objects and World

In this first subsection, it will be described how were the new simulator objects created, how were they processed to be included in the simulator, how are they managed inside the simulator project and summarily depict and list some of them.

As the objects in the virtual world have physical and graphic shape, we started by gathering 3D models of the equipment found inside the operating room. The first attempt to obtain these graphical representations lead us to 3D model libraries like:

- 3D Content Central [102]
- GrabCAD [103]
- RevitCity [104]

Some models found matched the characteristics of the intraoperative equipment, however the medical equipment in the Neurosurgery Service of Coimbra

University Hospitals operating room had singular shapes not matched by the models available. In order to create the desired environment, we had to design some of the medical equipment and furnitures ourselves. Such task was accomplished with the help of the 3D computer assisted drawing software *SolidWorks*® and Deep Exploration™ a product by Right Hemisphere.

On our first visit to the neurosurgery service, we were authorized to take measures of most objects, note the equipment brand and model and take pictures of the operating room from various angles. The photos taken during the surgery day served to identify the location of each equipment.

The design of each 3D model followed a more or less extensive sequence of commands and transformations. Each model was either integrally designed in a solid works "*.part" file, or divided in elemental components drawn in separate "part" files and then grouped in an "*.assembly" file. It is common to include in the assembly, conditions that restrict the interaction between parts. Moreover, in an assembly file the user can position each part in a desired position and orientation relative to the Solidworks origin referential, while in a part file the object placement in the world is defined from the moment it is drawn. This feature comes really handy, since the object origin in the simulator is the same as in Solidworks.

Several equipment related to the operating room were added (e.g. figures 5.7 and 5.8), to create a virtual environment as close as possible to the real operating room, and thus have a closer perception of the robotic systems' performance in a customized workspace.

Upon finishing the 3D models it was necessary to introduce them in the virtual world of CoopDynSim. This was achieved by using a conversion tool provided by Deep Exploration™, figure 5.9. The software handles several types of 3D CAD or DCC files, allows the user to edit the model and export the result into 2D or 3D supported graphic formats.

The models were exported as OpenGL CPP code files. The converter identifies and lists all the *materials*, *face indices*, *vertices* and *normals* contained in a 3D model. *Material* is an array of structures each one containing several color/lighting properties to be emulated like the ambient, diffuse, specular, emission and alpha

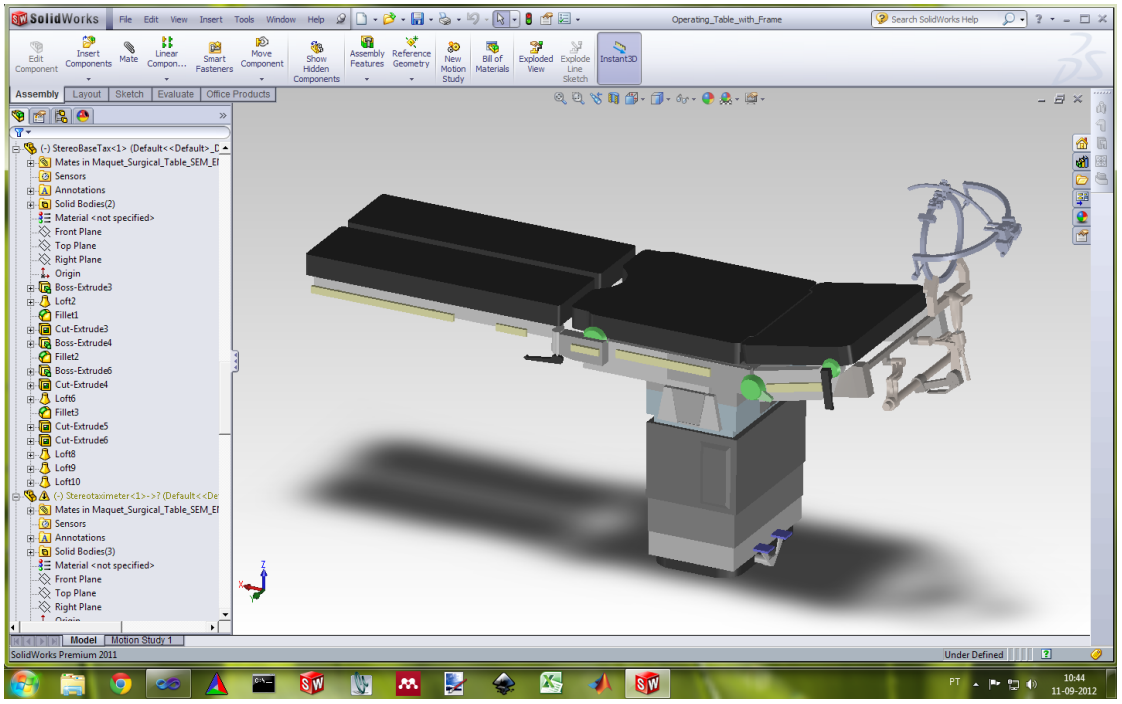


Figure 5.7: Operating Table 3D Model with Stereotactic Frame in Solidworks.

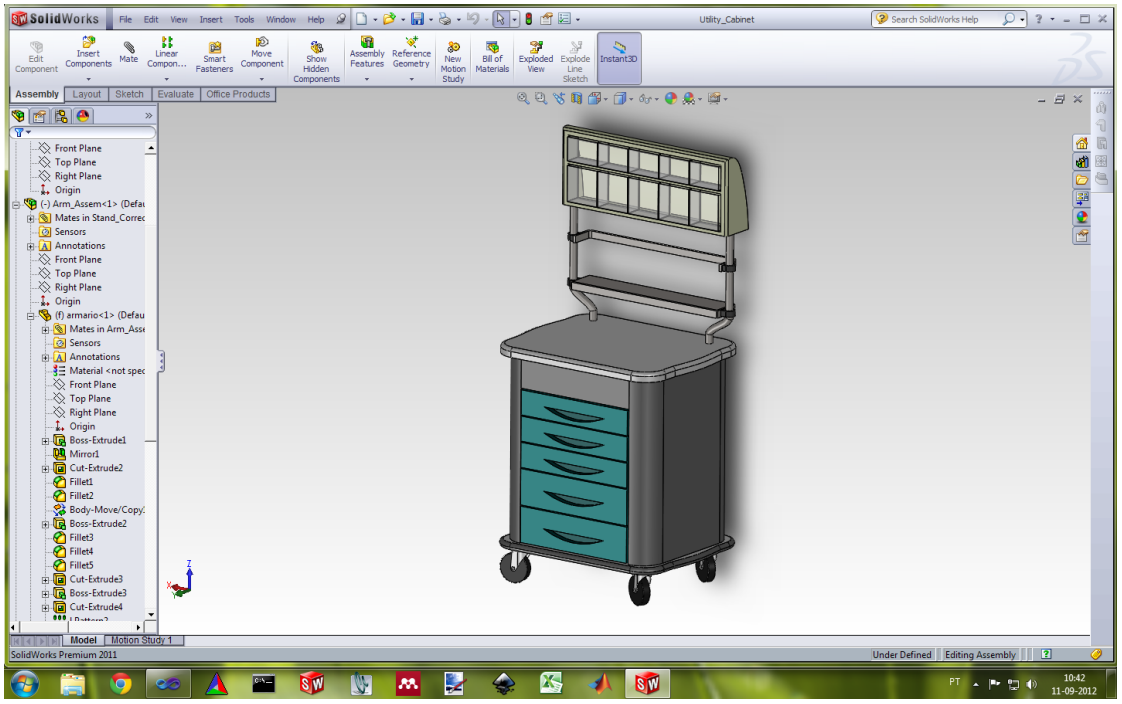


Figure 5.8: Utility Cabinet 3D Model in Solidworks.

or opacity. The objects can also be represented by a texture. Being objects in OpenGL rendered as a set of triangles, *face indices* is the bi-dimensional array

that represents the indices of the vertices to form triangles, while *vertices* array defines the position of each vertex and *normals* array defines the normal vector to each triangle.

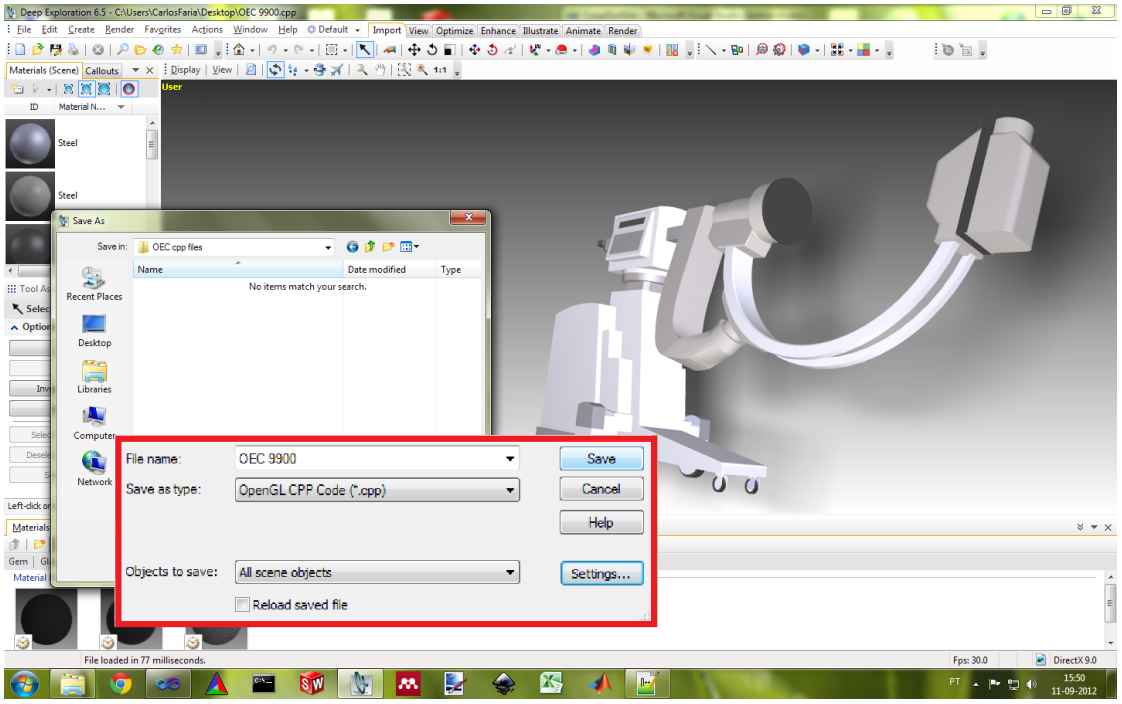


Figure 5.9: 3D model conversion to OpenGL readable cpp files.

The program is structured so each graphical object has its own class, derived from the **CObjectSim** base class, (figure 5.10). In its turn the **CObjectSim** inherits from:

- COpenGLObject class, responsible for generating the 3D graphical models within the simulator environment;
- CNewtonGDObject class, where are implemented methods to create or interact with the physical representation of objects;
- CDataLocation2GraphicsAndPhysics class, in charge of managing the object location information and make it accessible to graphics and physics, while avoiding conflicts.

The constructor parameters for **CObjectSim** are, a pointer to the *NewtonWorld* being emulated, a *Matrix* defining the insertion location of the object in the

world, a *Vector* defining the size factor[3] relative to the input model, a *float* value for the mass and a *Vector* for the color[4]. The object creation flexibility, granted by the possibility to change all these parameters, is particularly helpful when the user works with basic shapes like boxes, spheres or cylinders. However, since we are creating a virtual environment comprising objects with custom size and color, such object classes only require as parameters a NewtonWorld pointer, a location matrix and a mass value.

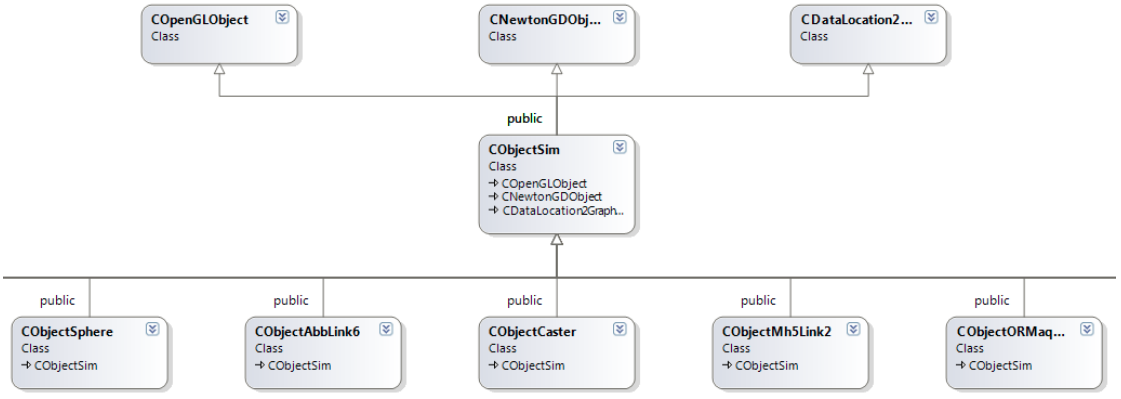


Figure 5.10: Simulator architecture on object classes and dependencies. Base classes, CObjectSim and some examples of derived object classes.

The **CObjectSim** class has virtual functions common to all the objects, such as Draw, Release and a pure virtual function Setup. Each object class can either use the Draw and Release functions defined at the base class, or implement their own. On the other hand, the Setup function is responsible for setting the object's singular graphics and physical representations, which are subjective to each one. Therefore it needs to be defined for every object. In figure 5.11 it is presented the surgical lights object, graphical and physical representation by black line contours that depict the NewtonGD collisions. The physics engine allow the use of boxes, cylinders and spheres as basic collisions, therefore to create complex physical bodies, the developer needs to aggregate several basic shapes to create a compound collision as represented in figure 5.11.

[3]**subsection 5.2.1** By X,Y and Z.
[4]**subsection 5.2.1** By RGB(Red Green Blue) standard.

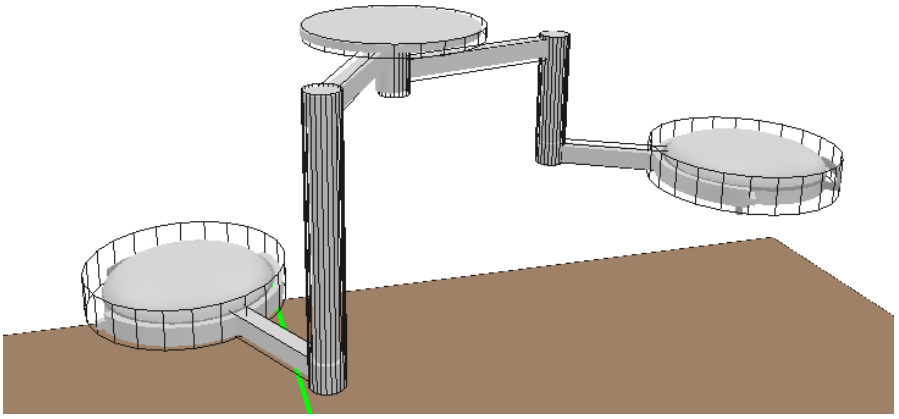


Figure 5.11: Surgical lights graphical and physical representations.

To create one single collision, the physics engine requires a position, orientation relative to the origin referential[5], dimensions and type of collision. Some of the created compound collisions have up to 10 basic shapes, which must be altogether specified for the 65 objects existing in the simulator, currently.

The NewtonGD provides the tools and methods to create physic shapes based in convex hull envelope approach. Using this functionality would greatly reduce the developer effort to emulate precise physic shapes, however it would also come with a large computational expense. Since we are simulating environments with rather complex objects, the simulator could become slow when running in less powerful machines. For this reason we did not follow the convex hull functionality.

In terms of visualization, the *CoopDynSim* UI window navigation tools were slightly improved. Both rotate and pan navigation were adjusted to be sensitive to the current value of the Z axis visualization plan, in order to allow a finer and precise navigation close to objects set in the world floor and to provide quicker navigation from an higher view to briefly adjust the visualization plan to a determined world coordinate. It was also added keyboard navigation functionalities with even greater visualization resolution than the previous implemented mouse events, so the user can combine both input devices to have an intuitive and further

[5]**subsection 5.2.1** This is one of the reasons why the referential should be correctly assigned to the object in *Solidworks* (design stage).

control over the visualization plan and angle.

5.2.2 Manipulator Robots

Being a simulator oriented to mobile robots, we still needed to implement robotic manipulators to be used in our project. The first task, similarly to the object creation task, was to find the graphical representations of each robot. Both *Abb*, *Motoman* and *Schunk* companies made the selected robotic systems 3D models available at their homepage.

The obtained 3D models, presented in section 4.5 (figures 4.7a, 4.9a and 4.11a), had every screw, nut, internal parts, wires and other details. Despite being visually more appealing, the truth is that each detail would hardly benefit the system simulated performance, while further adding to the computational resources required to render all details. The 3D robotic models were then modified to remove superfluous components, leaving only the body structure.

To simulate a robotic serial manipulator each link is an independent object, linked through simulated joint actuators created by the physics engine. The full 3D model was split in several links that were posteriorly introduced in the simulator for which it was assigned a physical body (compound collision). Since we are working with anthropomorphic serial manipulators, the cylindrical and revolute joints are emulated using hinge joints from NewtonGD libraries.

Each robot has a robot and a manipulator class. The robot class stands for the robotic system, so it includes both the manipulator as the robotic base and is responsible for setting all components and handling the communication. The manipulator class, includes all the link objects and is responsible to set the position of each link relative to the robot origin referential and also to define the hinge joints positions and orientations of rotation along the kinematic chain. When all the links are correctly positioned and connected through joints. The robots at their home position look like in figure 5.12.

At this moment the manipulator is no more than a chain of rigid bodies linked by free, non-actuated joints. In fact the robot's body could not hold their home

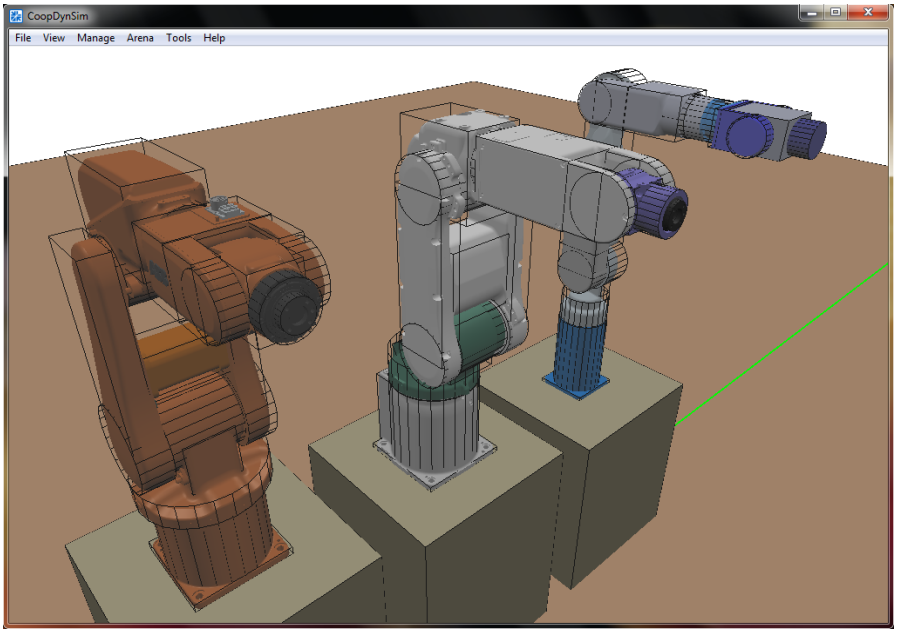


Figure 5.12: ABB IRB 120, Motoman MH5 and Schunk LWA II assembled in their home position.

position as in figure 5.12 and would fall right after their placement in the world. NewtonGD allows the developer to associate to each joint, an user defined callback that will exert control over the joint and will be updated every physics cycle. As in real manipulators, the simulated manipulators can be maneuvered through position or velocity. Each joint callback has a position and velocity control functions, that receive as parameters the current joint position or velocity[6] and the desired final position or velocity passed by the user, respectively.

Like mobile platforms, also robotic manipulators when inserted in the virtual world, are assigned to a new thread that runs independent from other threads. After initiating a thread, the robot is set on a run cycle where it updates its variables based on external communication. One of the upgrades made to the simulator was the implementation of signal/wait events, in order to update the robot variables only when a new communication message was received, instead of updating it every $50ms$. Upon receiving a new message, the robot will check the communication variables from each module and then update its own objects.

[6]**subsection 5.2.2** In our case, read directly from the physics engine. In real platforms such information is retrieved from sensors.

Joint controllers

Real robotic platforms have controllers with embedded control algorithms capable of moving each joint based in high level commands where the user only needs to explicit desired positions or velocities for the manipulator to follow. Being the goal of the robotic simulator to replicate as close as possible the real robot performance, the virtual joint controllers implemented are responsible to move the associated joints according to a target position and velocity, passed as parameters. By doing so, we guarantee that the same communication protocol implemented to control the simulated manipulators can be ported and used to actuate the real robots.

The hinge joints created using the NewtonGD libraries can only be controlled by acceleration. Thus, for both the position and velocity controllers, the difference between the current and desired values must be shortened by changing the acceleration. Additionally, each joint has a maximum and minimum friction value associated, which means that the acceleration input is not exactly reflected in the final joint acceleration.

In light of these system requirements, we chose to implement the position controllers using a PID algorithm, and the velocity controllers using a PI algorithm, based in Wescott, T. [105]. The control algorithm is the same for every joint, however each joint has a structure type object associated that keeps the proportional, integral and derivative gains, and also records previous errors. Starting with the position control function, it receives as parameters the target joint angle (value_desired), the current joint angle (value) and the joint to be actuated. It is presented below the pseudocode of the implemented controllers:

```
function Joint_Position_PIDController(value_desired, value, joint_index):
```

```
error = value_desired - value
P_term = Kp * error

integral = sum_previous_errors + error
I_term = Ki * integral

derivative = error - previous_error
D_term = Kd * derivative

previous_value = value_desired - previous_error
current_velocity = (value - previous_value) / time_step
output_velocity = P_term + I_term + D_term

previous_error = error

output_acceleration = (output_velocity - current_velocity) / time_step

return output_acceleration
```

Despite being omitted in the pseudocode, the joint_index variable was responsible for accessing the right joint controller structure with the correct gains and errors. To keep the algorithm simple to read, other details were also excluded in the example but still used in real code.

One of those details is the limits imposed to the integral term value. We noticed the sum of errors escalated too quickly as a result of disparity between the error decrement and the initial error value, which usually exceeded the decrement by several orders of magnitude. This situation lead to a large overshoot and posterior instability. Reducing the integral gain to nullify this behavior in all situations would render the integral term contribution too small, when the error was smaller or the joint angle was in the vicinity of the target value, regarding the output term. The solution found was to implement superior and inferior limits, so the integral term saturates when the initial error is too large, while keeping its contribution

significant for the final output when the error becomes smaller. We also limited the number of entries that were summed for the integral value, from all, to the previous 10. The initial differences between the initial joint positions to the desired position, were relatively large. The cumulative parameter was storing values since the first instant and those errors affected the arm performance throughout the movement disregarding how the manipulator was successfully approaching the final position.

Several conditions were also added to the control algorithm so that the joint, velocity and acceleration limits (tables 4.1, 4.4 and 4.7) of the selected manipulator would not be crossed.

The proportional, integral and derivative gains were manually tuned based in the methodology presented in [105]. Although real robots have different controller parameters for each joint, the information provided by the robot manufacturers fail to mention this details. In the absence of a better approach and facing the positive feedback of the adopted solution, the controller gains were set equally to every joint.

The velocity controllers are implemented similarly to the position controllers, the only differences reside in the use of desired and current velocities instead of joint angles, and the controller is a PI type. The derivative component was taken since the signal had some noise. While in the position control, it is rather easy to stop the joint from exceeding their angle limits by simply introducing a condition, in velocity controllers, this task gets more complicated since the control is based on velocities. The solution found was to pass the current joint angle as a parameter, and if the angle was within a certain range from the angle limits, the joint would respond normally to the controller whether if it was outside the range, it would stop. To assure that the joint stops before the limit, the condition was set for a slightly inferior range compared to the joint limits imposed. In some cases, the joint angle would be within the allowed limits, but violating the range imposed in this condition, which would stop the joint movement, even if the velocity was set in the opposite direction of the limit. To compensate for this exception, we added an alternative condition that would allow the joint to move if the signal of the current angle, was inverse to the signal of the velocity, figure 5.13.

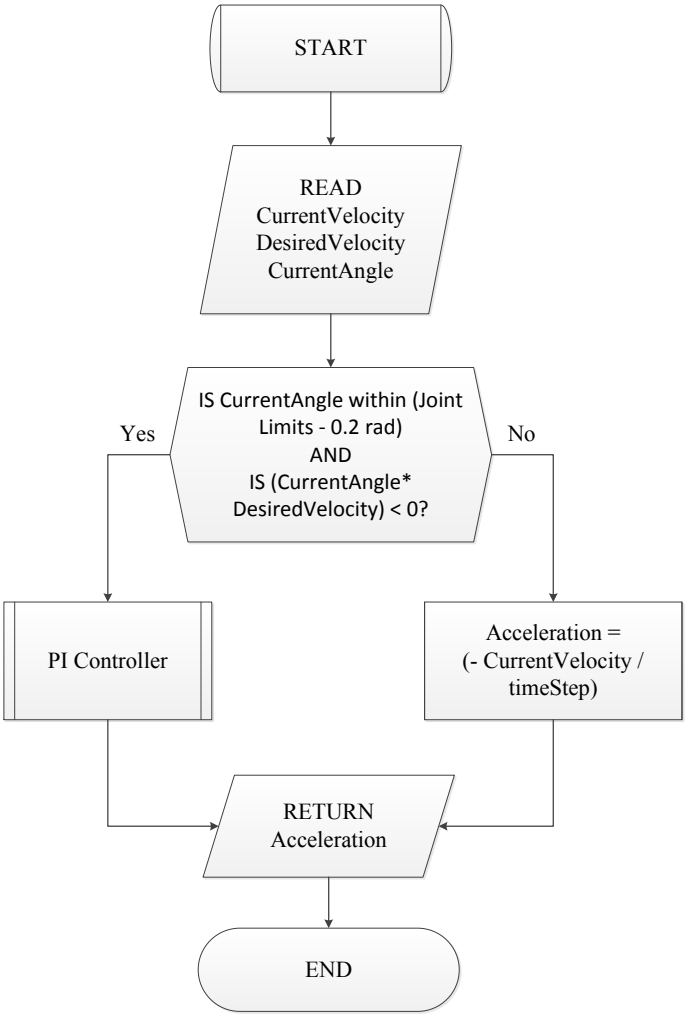


Figure 5.13: Flowchart of a velocity controller callback function.

The velocity proportional and integral gains, were also set using the manual tuning method [105]. We chose not to share the gains for each joint, since they are not the closest choice to the real system, nor totally reliable in terms of precision for the sake of our project. This is not considered to be a major problem, since the real robotic manipulators have optimized controllers with guaranteed precision and repeatability standards. One the other side, the implemented simulator controllers fulfill their duty to provide a reasonable precision, so the user can grasp how the robot will behave, how should it fit inside the operating room without conflicting with the intraoperative equipment and medical team and how should it be used in an assistive context in DBS surgery.

NewtonGD limited angle range

One of the problems that appeared along with the control code implementation had to do with the limited range ($[-\pi, \pi]$ radians) of how the physics engine deals with angles. Since the joint angles can go in some cases beyond $\pm 2\pi$ radians without overcoming the joint limits, when it went past the $[-\pi, \pi]$ radians range, the current joint angle would go from $-\pi$ to π radians and vice-versa. These discontinuities would result in a sudden peak on the difference between the desired and current angle, which would lead to an unstable behavior of the manipulators within the simulator. The easy solution to deal with this problem was to convert the desired joint angle into an angle within the ($[-\pi, \pi]$ radians) range. But, by doing so we would be reducing all the joint's range to this interval and consequently alter the robot's performance. We had to find a way to adapt the NewtonGD angle range to $[-2\pi, 2\pi]$ radians, at least[7].

Since this problem only occurs when the desired joint angle goes past $[-\pi, \pi]$ radians and since the angle value read from the physics engine is updated at each iteration, we set a condition to trigger when the joint desired value transposes $\pm\pi$ and when the absolute value of error overcomes 1.9π. Then, if the joint current angle is positive, on that physics cycle we subtract 2π to this value whereas if the it is negative, we add 2π. Considering the maximum velocity allowed for joint, the controller response time and the frequency of physics update, we know that when the joint overcomes the angle range set by NewtonGD, the error would be inferior but close to 2π. If we had chose an absolute error value of 2π instead of 1.9π the condition would only trigger when the current joint angle was coincident with the desired value, thus not giving enough time for the controller to respond, (figure 5.14).

[7]**subsection 5.2.2** Only the 6th joint limits from the ABB IRB 120, slightly overcome $[-2\pi, 2\pi]$ radians range. To brief the problem the limits of this joint were shortened to $[-2\pi, 2\pi]$.

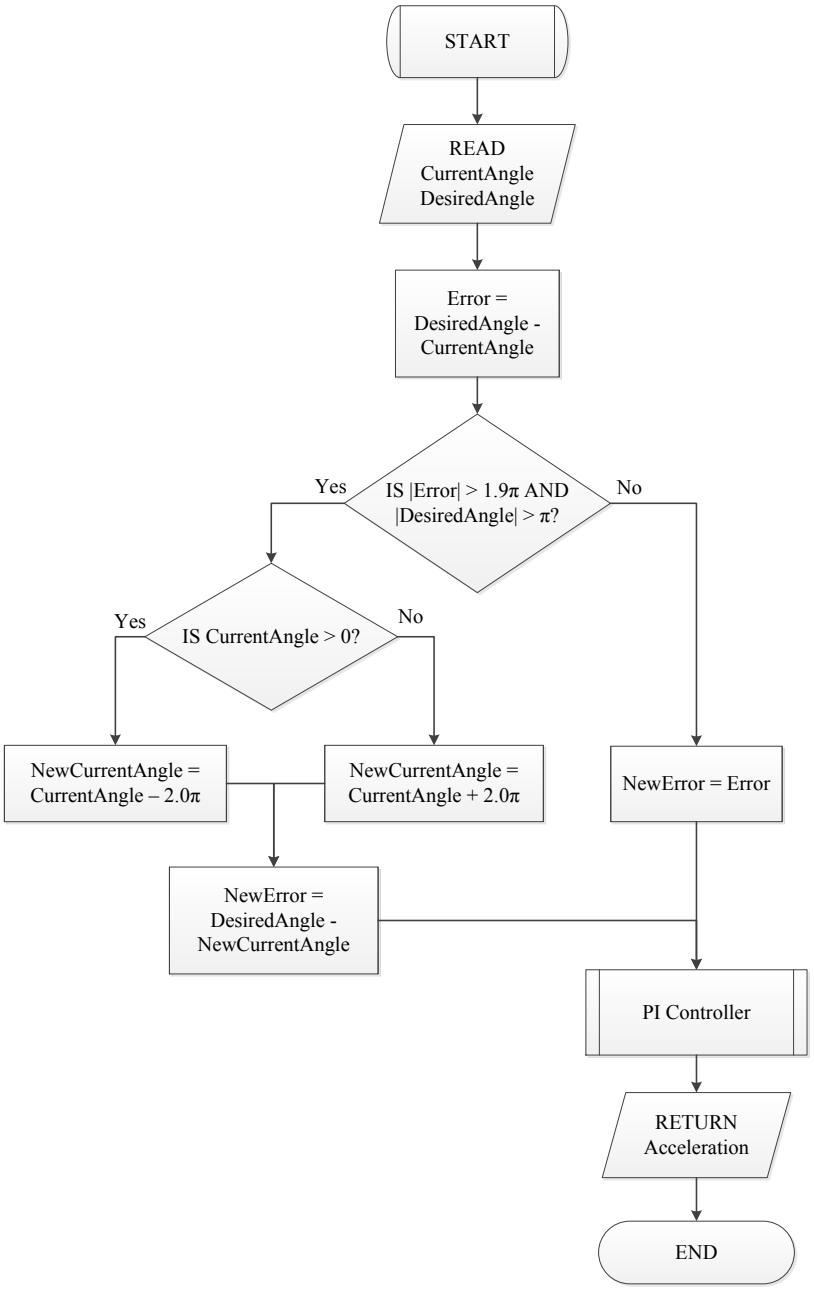


Figure 5.14: Flowchart of method to extend NewtonGD range from $[-\pi, \pi]$ to $[-2\pi, 2\pi]$ radians.

Communication

The robotic manipulator communication was devised using the same standards as mobile robots, which were already established in *CoopDynSim*. Based on YARP, we created a communication class for each manipulator module: actuated joints,

end-effectors and surgical plan.

The communication classes emulate the server that receives and stores or replies information from the client via a message protocol (figure 5.6). The information received can be handled by reply or process methods, depending on whether it is expected the server to provide feedback or not, respectively. On the server side, the client commands can either be an information request or simply a directive to exert control over the simulated elements. In both cases it is used the reply over the process method, because even outside the commands that request feedback, the simulator will return a response concerning the successful delivery of the instruction or not and why.

Regarding the joint actuator module, the server recognizes commands for:

- *ARM_GET_NJOINTS* to retrieve the number of joints of the manipulator;
- *ARM_GET_ANGLE* to set the desired joint angles for each actuator;
- *ARM_SET_ANGLE* to retrieve the current joint angles from each actuator of the manipulator;
- *ARM_GET_VELOCITY* to set the desired joint velocities for each actuator;
- *ARM_SET_VELOCITY* to retrieve the current joint velocities from each actuator of the manipulator.

Despite having the same structure, the message carried differs depending on the command. The *Error code*, *Text* and *Command* variables (figure 5.6) are always present in the message, the integer and float variables might or not be included and can vary in number. When the client sends *GET_NJOINTS*, the server returns the number of degrees of freedom as an integer variable. When the client sends *GET_ANGLE* or *GET_VELOCITY* commands, the server returns a vector of floats with the joint angle or velocity of each joint with the size corresponding to the number of joints. The *SET_ANGLE* and *SET_VELOCITY* message is defined so the user can selectively actuate in different joints. The message integer variables carry a vector with the indexes of the joints to be actuated; the float variables carry the desired joint angles or velocities of the correspondent joint indexes. Example following the protocol established (figure 5.6):

```
Message = (0, "Send Desired Joint Angles", 7604, 3, [1,3,5], 3, ...
[1.047, 0.524, 0.785]);
```

If the example message was sent, the robotic manipulator 1st, 3rd and 5th joints would be moved to 1.047 *rad*, 0.524 *rad* and 0.785 *rad* respectively. When a communication event is signaled, the message updates the server desired values (joint angles or velocities) based on the client message, while the server current values (joint angles or velocities) are refreshed based on the robot classes. The robot class is then responsible to pass the desired and current values to their actuator's callback functions, so the physics engine can move the selected joints.

The other modules will be addressed in the *End-effectors* and in the *Surgical plan* subsections below.

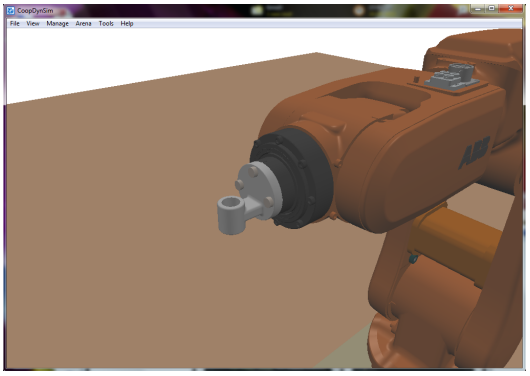
End-effectors added

One of the most desired features in *CoopDynSim*, was the possibility to add and use robot operative oriented end-effectors[8] in the manipulators. The robotic system as idealized for neurosurgeon assistance is expected to be able to change its end-effector instrumentation during the procedure, without the need to recalculate the robot's position. In light of this feature we also developed the simulator towards the capability to change its end-effector on robot instantiation and during runtime.

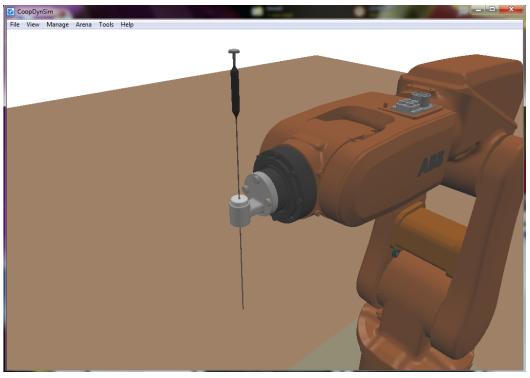
The robot can either be used to lower the medical instrumentation itself using *Differential kinematics* or be used to simply orient the instrumentation for the neurosurgeon to move it along the defined trajectories. However, in simulation there is no human to move the surgical tools, so we implemented the end-effectors as rigid bodies and as 1 DOF slider actuators. It was created a passive end-effector for instrument orientation (figure 5.15a), an actuated end-effector to simulate the guidance of a probe (figure 5.15b) and another active end-effector to simulate a trepan instrument (figure 5.15c).

Like any other operating room objects inserted in the virtual world, we started by creating the 3D models and including them in the simulator as standard objects.

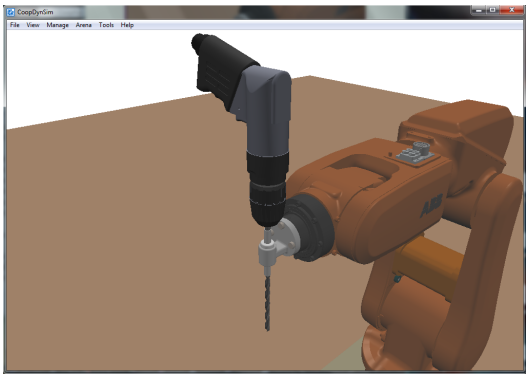
[8] **subsection 5.2.2** which we will henceforth, simply refer as end-effectors



(a)



(b)



(c)

Figure 5.15: Robot end-effectors.

The guiding end-effector (figure 5.15a) was the easiest to implement, since it is a static single object that should only be positioned and attached to the last robot link. The actuated end-effectors implementation involve NewtonGD slider joints, to displace the surgical tools relative to the end-effector body attached on the robotic arm. Similarly to the robotic arm joints, a callback function was assigned to the slider, which in its turn calls a controller function based in a PID control. The controller receives the current and the desired position of the slider, and iteratively manages the acceleration of the joint until both positions are coincident. The control algorithm is the same used for the manipulator joints.

Updating the end-effectors during run-time was a challenging task since it involves several program cores: the **Dialog** where the user selects the end-effector, **Data** that links the dialog interface to the robot's class, **Robot** where the end-effector is actually attached to the robotic arm and the **Communication** classes

responsible for handling the client commands towards end-effector maneuvering. If the end-effector was attached when the robot is instantiated, the end-effector position relative to the robot would be always the same, since the robots are inserted in their home position. However, the position to attach an end-effector on the manipulator's last link during run-time is dependent on the current robot body configuration.

When the end-effector is instantiated, it is possible to specify a transformation matrix to define the position and orientation of the object insertion. To calculate this matrix, we started by obtaining the location of both the last ($\mathbf{p}_n$) and the second last link ($\mathbf{p}_{n-1}$) from the manipulator class. With both positions we computed the coordinates of the vector that passes through both points and normalized it ($\hat{\mathbf{v}}$) .

$$\hat{\mathbf{v}} = \frac{\mathbf{p}_n - \mathbf{p}_{n-1}}{\|\mathbf{p_n} - \mathbf{p_{n-1}}\|} \tag{5.1}$$

The coordinates to place the end-effector ($\mathbf{p}_e$) are also dependent of the distance from the last link referential to its outer limit where the end-effector is attached. This distance is represented by L_n,

$$\mathbf{p_e} = (\hat{\mathbf{v}} * L_n) + \mathbf{p}_n \tag{5.2}$$

The orientation to attach the end-effector is the same as the last link one.

On the communication side, the client only has 3 commands to interact with the simulator on the end-effector module:

- *ARM_GET_ACT_TYPE* to retrieve the type of the end-effector currently attached to the manipulator;
- *ARM_GET_ACT_DIST* to retrieve the current end-effector slider position;
- *ARM_SET_ACT_DIST* to set the desired end-effector slider position;

The end-effector type is sent in the message as code integer value: 0(none), 1 (empty), 2 (probe) and 3 (trepan). The current and desired distance for the end-effector slider actuator is a continuous number and thus is passed as a float value.

Surgical plan

It was developed a new feature that allows the user to visualize the preoperative directives inside the simulated room, namely the target to be stimulated and the trajectory through which the instrumentation must descend. The robotic simulated system at this point was capable of maneuvering based on position or velocity directives, hold different end-effectors, follow trajectories and reach target points. However, the visual sense of the robot moving in the virtual world was not enough to evaluate how the manipulator was behaving relative to the preoperative plan.

The idea was to create basic shapes without a physical body in the virtual world to represent the preoperative coordinates, to see how was the instrumentation befitting the planned trajectories and also to facilitate the debug process of the control algorithms. The target to be stimulated was marked as a red sphere while the trajectory to be followed was depict as a green thin cylinder of fixed dimensions having one of its tips coincident with the target and the other 45 centimeters away from the target along the trajectory vector, (figure 5.16). Both entities were design with transparency, so the user could visualize the instrumentation passing through the trajectory or reaching the target.

In terms of implementation, these targets were designed in *Solidworks* and the resulting 3D models introduced in the simulator. The target and trajectory classes derive from the **CObjectSim**, however since these shapes do not have a physical representation, the Draw, Release and Setup methods had to be reimplemented instead of using the ones from the base class. Neither the Setup nor Release functions have any reference to the physical engine, so they only generate/destroy the graphical representation. The Draw function is very similar to the **CObjectSim** one, however it does not have the possibility to draw the physics contours.

Starting with a single preoperative directive of a target position ($\mathbf{p}_{tar}$) and an entry position ($\mathbf{p}_{ent}$) from the imaging system, the simulator was expected to place both surgical plan entities (target and trajectory) according to these coordinates. We started by positioning the center of the target sphere at the target location specified. Since the trajectory cylinder had its origin frame at the center of one

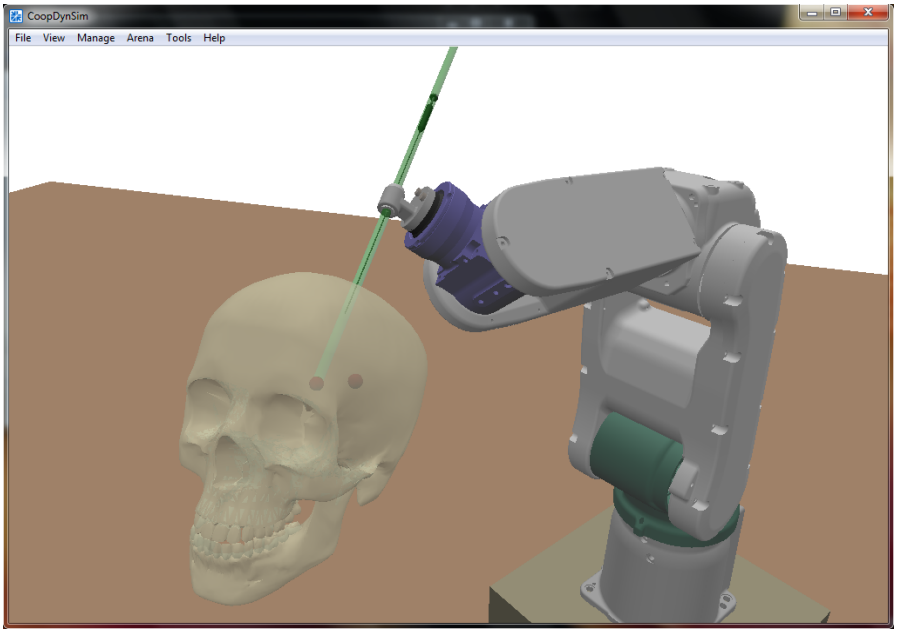


Figure 5.16: Probe end-effector guided by the MH5 robotic arm towards a surgical target.

of its bases, we placed the cylinder base coincident to the target position. The trajectory orientation is defined by a vector ($\mathbf{u}$) created from the target to the entry point and not on the contrary, because one base of the trajectory cylinder must be coincident with the target instead of the entry point[9].

$$\mathbf{u} = \mathbf{p}_{ent} - \mathbf{p}_{tar} \tag{5.3}$$

The final trajectory orientation is the rotation matrix that places the trajectory z-axis collinear to $\mathbf{u}$, being z-axis the axis along the cylinder's height, (refer to the left and center subfigures of figure 5.19). We calculate this rotation matrix based in 2 consecutive rotations, firstly around the z-axis and the secondly around the x-axis. The first rotation around z-axis, places the trajectory x-axis collinear to a vector ($\mathbf{w}$) that is normal to both the world z-axis and to $\mathbf{u}$. This step will guarantee that the posterior rotation around the x-axis can place the cylinder z-axis straight with $\mathbf{u}$. Being $\hat{\mathbf{z}}$ the unit vector $[0, 0, -1]$, the vector $\mathbf{w}$ is given by

[9]**subsection 5.2.2** The entry position can have difference euclidean distances from the target, but the trajectory graphical representation has a fixed length of 45 cm

the cross product:

$$\mathbf{w} = \hat{\mathbf{z}} \times \mathbf{u} \tag{5.4}$$

With the resulting vector, the desired angle for the initial z-axis rotation is given by the arc tangent of the y and x components of the $\mathbf{w}$:

$$\theta = -\arctan_2(\mathbf{w}_y, \mathbf{w}_x) \tag{5.5}$$

The second rotation angle, around the new x-axis, should be:

$$\phi = \arctan_2\left(\sqrt{(\mathbf{u}_x)^2 + \mathbf{u}_y)^2}, \mathbf{u}_z\right) \tag{5.6}$$

Since the trajectory is a cylinder, the rotation can be made in 2 steps and thus the ψ angle is kept 0. With the rotations along each axis, the rotation matrix for the trajectory is provided by simply pre-multiplying the Roll-Pitch-Yaw matrix. Hence, when the user inputs the target and entry coordinates, the surgical plan class will generate the correspondent target and trajectory marks in the virtual world, figure 5.16.

The preoperative information can contain coordinates for several trajectories. The end-user will be interacting with the virtual world through the client control application. The client program has a manageable list that contains all the surgical entities represented in the simulator. Each surgical entity is handled in the client and by the communication as a vector containing the coordinates of target and entry points. Through the client, the user can add, remove single entities or the entire list, additionally the user can select one surgical entity from the list.

Since each entity is represented by a target sphere and a trajectory cylinder, when the user is working with multiple entities at the same time it can get a little confusing. With the select function, only the selected entity trajectory is shown while all the others are omitted, except for the target marks which are always visible (see example with 2 entities and only one selected in figure 5.16). It is also provided the option for the user to reveal all the entities again.

The surgical plan together with the joint and end-effector modules, form the communication supported for manipulators, towards a surgical assistive role. Like the joint and end-effector modules, also the surgical plan, is associated to a robot instance whose variables are updated when a communication event is received/triggered, during the robot running cycle.

In the simulator code, the surgical plan is represented by a structure that stores a vector with information relative to each entity, namely: i) the entry and ii) target coordinates, iii) a flag that indicates whether the entity is selected or not and iv) an unique ID that identifies each added entity.

The communication commands that establish the link between the client and server applications are as follows:

- *ARM_SET_SURGICAL_PLAN* to set the entire surgical plan entities coordinates;
- *ARM_SELECT_SURGICAL_PLAN_ENTITY* to select one entity from the surgical plan, and hide the others;
- *ARM_ADD_SURGICAL_PLAN_ENTITY* to add a surgical plan entity to the existing ones;
- *ARM_REMOVE_SURGICAL_PLAN_ENTITY* to remove a surgical plan entity from the list;
- *ARM_REMOVE_ALL_SURGICAL_PLAN* to remove the whole surgical plan;
- *ARM_SHOW_ALL_SURGICAL_PLAN* to show all the surgical plan entities trajectories;

Starting with the *SET_SURGICAL_PLAN*, the message from client has 1 integer value that counts the number of entities sent and a continuous vector of floats for all the entities, which concatenates groups of 6 values that stand for the entry and the target coordinates. The simulator goes through each set of 6 values, adds the first 3 to the entry coordinates variable and the other 3 to the target coordinates, sets the current ID to the read entity and increments the global ID for the next entity, and sets the trajectory flag as selected. In the *SELECT_ENTITY*

command the client sends the index of the entity to be selected as one integer value. The simulator module upon receiving the message will go through the vector of entities and set the select flag value to 0 in all elements except for the index indicated in the message. When drawing each surgical plan entity, the simulator will check if that entity is selected or not and draw the trajectory when the condition is met. In a *ADD_ENTITY* command message, the float parameters include 6 values that represent the entry and target coordinates, which are added to the surgical plan vector in a similar fashion as the command to set the whole surgical plan. In the *REMOVE_ENTITY* command it is specified the index of the entity to be removed while in *REMOVE_ALL* command, all the entities are removed. When the command *SHOW_ALL* is received the selected flags of each entity are toggled to 1.

5.3 Controller Interface

Having built the simulator basis and upon establishing the communication protocol, we headed to develop the control application. This application is expected to establish a communication link with the simulator or an external device, to control virtual or real robots via different kinematic approaches, introduce and set the surgical plan, manipulate actuated end-effectors and to provide online information back to the user about the robots.

The need to quickly develop an application where it was possible to promptly implement, test algorithms and scrutinize data, with further abstraction from computer domain drove us to pick a high-level language. Aside from the need to easily create customized UIs, the other programming language's pre-requisite is the ability to support and call YARP library, which is responsible for setting up the client-server communication. Being YARP a C/C++ based project and due to the high portability and wide reach of the language it can be used in several programming languages through the interface compiler SWIG[10], among others:

- Java

[10] **section 5.3** `http://www.swig.org/`

- PERL
- Python
- C#
- MatLab (via Java)

SWIG is responsible for generating wrapper code that higher-level/interpreted languages require to access and use underlying C/C++ code.

Upon evaluating the possibilities and regarding the acquired knowledge and previous experiences, we selected MatLab as the implementation language. It rounds up all the sought characteristics, which makes it a perfect platform to developed at least the initial approach towards a final control application solution. Furthermore, MatLab was the language suggested to develop the control application by the medical team involved. The process of porting YARP to MatLab followed, can be referred at:

1. `http://eris.liralab.it/wiki/Calling_yarp_from_Matlab`
2. `http://eris.liralab.it/viki/images/6/63/Yarp4Matlab.pdf`

After importing the shared YARP library and the generated classes to MatLab, the following step was to implement a class capable of managing the communication events on the client application side, named **YarpClient**. This class constructor loads the YARP library, initializes the Network class responsible for manipulating the YARP network including initializing and shutdown, sets the client port, initializes the carriers of messages ("bottle" objects) and defines several flags that indicate whether the client port is opened and connected. The **YarpClient** class has methods for connecting the client device to another device through the network, several methods to send information (with, without and ignoring the reply), a method to close communication, a network check function and accessors to change the devices names.

With the client-communication interface implemented, we started developing the client application. Since the application revolves around an event driven UI, its core ends up being the *function* and *figure* files of the UI. The program structure is composed by various functions and classes, that can be summarily grouped

in central cores depicted in figure 5.17, which will be described in the coming subsections.

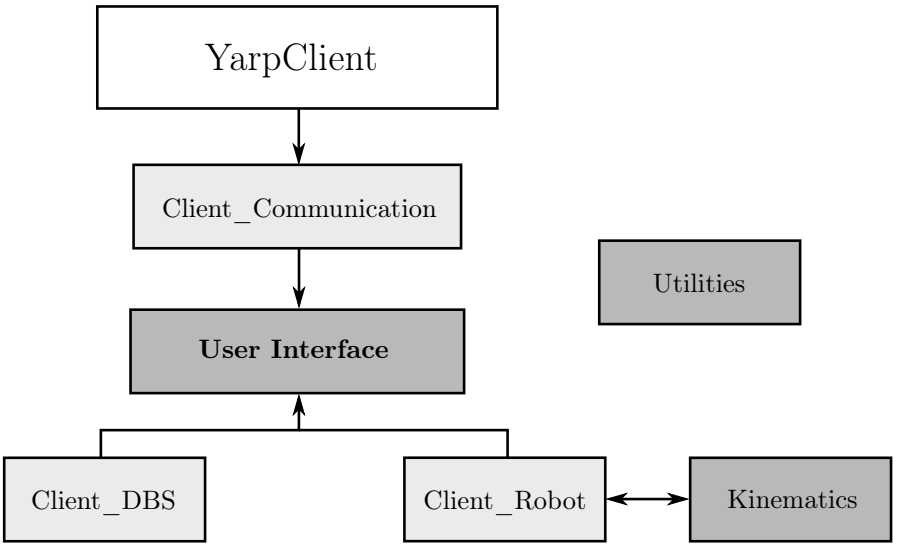


Figure 5.17: Diagram of Control Application basic architecture.

5.3.1 Utilities

The core entitled "Utilities" (see figure 5.17) is independent from all the others and most of its implemented functions are recurrently used across the application. It gathers a set of isolated tools that fulfill general and basic duties, to make the code simpler and easier to read.

Among the files we can find methods to convert from the Roll-Pitch-Yaw values to the rotation matrix equivalent and vice-versa, to compute the transformation matrices from Denavit-Hartenberg parameters, to calculate the transformation matrices from the surgical referential to the robotic base referential and also from the tip of the manipulator last link, to the tip of the end-effector. If the end-effector is an actuated device, the transformation is updated at each iteration according to the displacement of the instrumentation.

One fundamental issue with surgical manipulator robots on mobile platforms is the ability to firmly attach its base and grasp its position relative to the surgical referential, relative to which all intraoperative process is planned. In spite of not

having developed any procedure to retrieve the surgical referential relative coordinates, we still implemented the function to calculate the resulting transformation if the coordinates were known. With this transformation the user can coordinate the robot actions, and understand its position relative to the surgical referential instead of its base referential. The same can be said for the end-effector instrumentation, the transformation matrices calculated allow the user to understand at each moment the position and orientation of the tools instead of the arm. With these, we can isolate the robotic control from the operating room positioning and end-effector used.

5.3.2 Kinematics

The Kinematics core rounds up a set of functions distributed by several files that include the geometric and differential kinematic functions, the full Jacobian computation method and also some cost function algorithms for optimal arm posture for each manipulator included. The kinematic equations presented along chapter 4 are implemented in these methods, always taking into account each robot's specific features also described in chapter 4. Before looking with detail at the developed algorithms, it is important to refer that the kinematic methods implemented are rather similar across the different robots. In order to avoid an extensive and repetitive descriptive process for all the robot types, we chose to present the basic algorithm and highlight only the differences.

Implemented Geometric Direct Kinematics

The geometric direct kinematics function expects as parameters *Client_Robot* object containing specific characteristics of each robot, the current joint angles, the surgical reference frame[11] to the robotic base reference frame[12] and the manipulator's last link to the end-effector tip transformation matrices. It returns a vector with the position and orientation of the current end-effector relative to the surgical

[11] **subsection 5.3.2** henceforth, referred as **surgical frame**
[12] **subsection 5.3.2** henceforth, referred as **robot base**

referential. The Denavit-Hartenberg values are accessed through the *Client_ Robot* object and the joint displacement variables $(\theta_1, ..., \theta_n)$ associated to the current joint angles, see table 4.3, 4.6 and 4.9.

According to equation 4.26, one can express the transformation from the manipulator base referential to its last link $({}^0\mathbf{T}_n)$ as the product of transformations from link to link taking into account the robot's singular dimensions.

Each link to link transformation is computed using an *Utilities* function that receives the Denavit-Hartenberg and returns the correspondent transformation. After multiplying the matrices, the resulting manipulator position and orientation alone can be calculated using equation 4.3. To obtain the final end-effector transformation relative to the surgical frame, the surgical frame to manipulator base $({}^b\mathbf{T}_0)$ and manipulator's last link to end-effector's $({}^n\mathbf{T}_e)$ transformation are needed:

$$ {}^b\mathbf{T}_e = {}^b\mathbf{T}_0 \; {}^0\mathbf{T}_n \; {}^n\mathbf{T}_e \tag{5.7} $$

From the resulting matrix the user can retrieve the end-effector position coordinates (4.24) and a rotation matrix to be converted to Roll-Pitch-Yaw parameters using the *Utilities* tools.

Implemented Geometric Inverse Kinematics

The geometric inverse kinematic function receives as input a *Client_ Robot* object, the desired final end-effector position and orientation $({}^b\mathbf{T}_e)$, and the transformation matrices: from the end-effector to the manipulator's last link $({}^e\mathbf{T}_n)$ and from the robot base to the surgical frame referential $({}^0\mathbf{T}_b)$. It returns a set of possible arm configurations as vectors of joint angles, that enable the end-effector to reach its target.

The first step of this function is to transform the desired position and orientation into a matrix shape. Following the same philosophy as for geometric direct kinematics, it is expected that provided position and orientation would be relative to the surgical target and to be achieved by the end-effector. These coordi-

nates passed as parameter need yet to be be translated to the robotic manipulator space, which is accomplished by multiplying the (${}^0\mathbf{T}_n$) with the transformations also passed as parameters:

$$ {}^0\mathbf{T}_n = {}^0\ \mathbf{T}_b\ {}^b\mathbf{T}_e\ {}^e\mathbf{T}_n \tag{5.8} $$

With the manipulator desired position and orientation (${}^0\mathbf{T}_n$), the geometric inverse kinematics inverse algorithm is presented in figure 5.18. Following the algorithm presented in chapter 4, we start by calculating the wrist position for a given orientation, equation 4.29. Then the program does an initial check, to assess whether the wrist position can be reached by the lower arm or not. If the condition fails, the algorithm stops and instead of possible joint angle solutions it retrieves an error code to notify the user for the fact that the target position falls outside the robotic manipulator workspace. If the position is reachable, the algorithm tests every combination of the 4 possible configurations, (see equations 4.35 and 4.40) which will be latter subjected to a selection criteria to decide which robot configuration to chose. The rest of the algorithm presented in chapter 4, is strictly followed and implemented using standard geometric and algebraic tools, since the computation of the inverse kinematics for the arm to solving the spherical wrist coordinates. In the end, the joint angles vector solution is tested to check whether any joint limits are violated, and only viable solutions are returned.

The only significant different between the implemented algorithm and the theory equations resides in the fact that similarly to real robots, the simulated systems do not have infinite precision. In conditions where a variable was checked to have a specific value like 0 or π (equations 4.31 or 4.41), we had to set the condition to trigger when the variable was within a short range ($\pm 0.01 rad$) of the limit value.

The cost function sets the criteria to distinguish the best arm configuration for the desired target among all the possible solutions. As stated previously, the selection condition is biased by the premises of avoiding collision to intraoperative elements and having the smallest joint displacement relative to their home position, situation in which the joints are considered to achieve their maximum

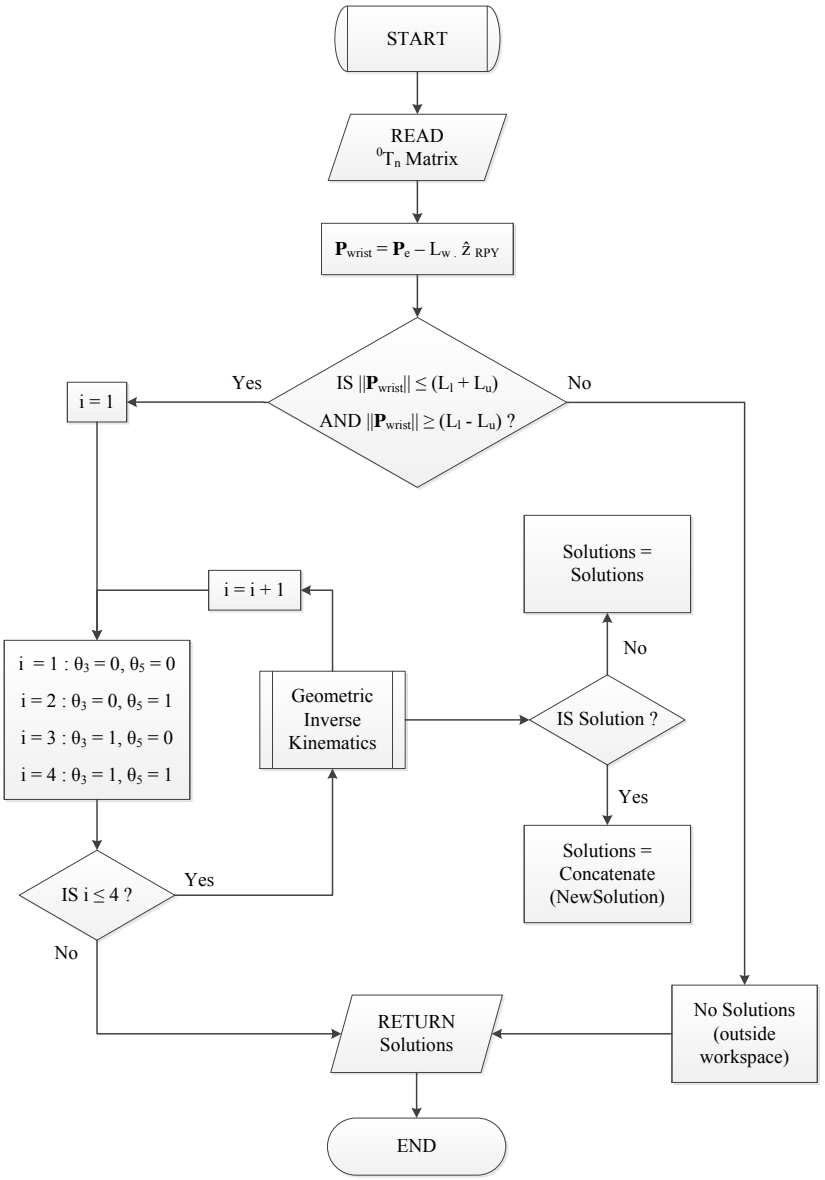


Figure 5.18: Geometric Inverse Kinematics generic algorithm.

precision. We started devising the collision detection algorithm, but the complexity of the problem regarding a dynamic environment and time limitations rendered the solution unfeasible in this dissertation. Nevertheless, we will describe an initial approach and the main ideas towards a solution in subsection 5.3.7. To avoid hazardous interactions between the robot and the surgical team or other surgical equipment, we will initially rely in a weighted spatial placement of the manip-

ulator. Therefore the cost function implemented only focus the principle of the smallest joint displacement relative to their home position. Being ($\mathbf{q}$) a vector of joint angles from the k number of solutions, the initial cost function approach was,

$$C = \min_{i=1,2,\ldots,k} \sum_{n=1}^{DOF} \mathbf{q}_i(\theta_n) \tag{5.9}$$

The inverse kinematics function for the 7 DOF arm, is slightly different since besides the end-effector expected position and orientation, the function also expects the arm plane angle (α), (refer to chapter 4 section 4.3.2). The geometric inverse kinematics function starts by converting the desired coordinates for the end-effector relative to the surgical referential to the robot space coordinates (from robotic base to the tip of its last link), (see equation 5.8). The algorithm presented in chapter 4 was thorough followed to retrieve the final joint angle displacements, (see figure 5.18).

However, the algorithm presented was developed considering the arm to be placed along the negative y-axis, whereas our solution is drafted accounting that the manipulator is placed with its shoulder along the positive z-axis. To bypass this problem, in addition to the initial transformations to the parameter coordinates, we added on further transformation to account for the base referential rotation. One might wonder why not to develop a solution according to the expected robot placement. The fact is that the actual Schunk LWA II arm in our laboratory, used for testing, is mounted in the position depict in figure 4.5, and the kinematic functions for this configuration were already implemented by [91]. By simple adapting the base referential we can reuse the developed code to our solution, and easily manage/remove this transformation to test our solution with the actual mounted robotic manipulator without further coding.

In the end, the inverse kinematics returns the possible joint angle solutions (at most 4) for a specified end-effector position, orientation and arm plane angle (α). The cost function besides selecting the lowest sum of joint angles among the solutions, also simulates the geometric inverse kinematics for all the possible arm plane angles. By iterating the arm plane angle value, one can explore other

manipulator configurations and perhaps find possible or better solutions. For each arm plane angle value, the geometric inverse kinematic function needs to run again, so there is a tradeoff between the arm plane angle iteration step and the computational demand/time. After some empirical experiments, we converged to an iteration step of 5 degrees and therefore 72 cycles.

In 6 DOF manipulators, the k number of solutions is at most 4, while in 7 DOF the number of solutions can ascend to 288 (5.9). In the end, only the possible solutions that fit within the joint limits are considered as viable, and from the selected group the one with the lowest joint angle displacement sum will be chosen, equation 5.9. If there is no solution, the control UI notifies the user.

Implemented Differential Kinematics

Since our differential kinematics method is based in the *Analytical Jacobian*, the first step is to compute the Jacobian matrix that establishes the relation between the joint velocities and the end-effector linear and angular velocity. To build the full jacobian matrix given by 4.56, we first need to calculate the transformation from the base of the manipulator to its end-effector, and each partial premultiplication matrix from the base to each manipulator joint[13].

The Jacobian function receives as parameters a *Client_ Robot* object, the current joint angles, the transformation from the robot to the end-effector (or vice-versa if we are dealing with direct kinematics) and a flag vector variable to enable or disable control over specific linear or also angular velocities components. By accessing the *Client_ Robot* object and adding the current angles to the Denavit-Hartenberg parameters, we stored each transformation matrix from link to link. With each separate matrix it was possible to briefly compute the transformation from the base to the end-effector, by premultiplying all, but also to compute each partial cumulative premultiplication matrix used in the *Analytical Jacobian*. With both these matrices we can compute the Jacobian following the expression 4.56 in chapter 4 section 4.4.1.

[13] **subsection 5.3.2** (Example) The partial premultiplication matrix from the base to the 3rd joint would be: ${}^{0}\mathbf{T}_{3} = {}^{0}\mathbf{T}_{1}\ {}^{1}\mathbf{T}_{2}\ {}^{2}\mathbf{T}_{3}$

The differential direct kinematics can be computed by multiplying the resulting Jacobian matrix with the current joint velocities (equation 4.54), to obtain the cartesian and angular velocities. On the other hand for the differential inverse kinematics, the required joint velocities can be computed by multiplying the inverse Jacobian matrix with the desired velocities vector (equation 4.58). The differential inverse kinematics function, returns an error code and a vector of each joint velocity if the error code returns null (success). The error code identifies whether the result is approved, if the manipulator has reached a singularity or if the output joint velocities cross any manipulator velocity limits, for which cases the returned velocity is ignored.

There is however a slight difference in the algorithm for the 7 DOF manipulator in the differential inverse kinematic function only, because its Jacobian matrix is not square shaped. Therefore, we resorted to the solution described in chapter 4, where we used the right pseudo-inverse calculated as in equation 4.60, in this case the each joint velocity is computed following the formula 4.61.

5.3.3 Client Robot

The UI file function called immediately after the pre-generated initialization code, also known as the dialog create function is in charge of initializing all the classes and global variables. Since the control application is expected to control several types of robots with different features and dimensions, this starting function calls an initialization script file created to set all the specific characteristics of each simulated manipulator. This file is called whenever a new UI window is opened and instantiates the 3 types of *Client_Robot* objects used throughout the program (Abb IRB 120, Motoman MH5 and Schunk LWA II), and all the associated information required for the developed control algorithms.

In the initialization script file the robot **name** and **type** are specified at the constructor stage. Afterwards the **number of joints**, manipulator **physical distances** between links, **joint**, **velocity** and **acceleration limits** and **Denavit-Hartenberg** parameters are also assigned. After fitting all the properties to the

associated robot types, a vector of robot objects is dispatched to the UI function handles to be accessed by callbacks or other classes.

The *Client_ Robot* class properties have private "set" privileges and public "get" access privileges, so the script file instantiation had to resort to accessor methods. Aside from the properties previously referred, the class also includes the **offset**, variable that saves the offset from the origin joint position to the used home configuration joint position; the transformation matrices from the manipulator's last link to the end-effector tip, and vice-versa; and the transformation matrices from the surgical referential to the robot base and vice-versa.

The class has a method to check the joint limits that receives as parameters the target joint values and checks whether they overcome or not the specific limits for the current robot type. The class has the geometric direct kinematics, inverse kinematics and differential kinematic functions that in their turn call the function associated to each type of robot. So when the instantiated robot is selected in the control application the same function can be used to control any of the included manipulators.

5.3.4 Client Communication

The communication class works as an additional layer to smooth data transition from robot oriented commands to abstract communication, **YarpClient**. It has several properties of private "set" and public "get" access. The *Client_ Communication* class stores information about the **server port** and **client port**, has a variable to account for the state of connection and communication (providing the information to the user by an UI field). Regarding the communication details, it keeps track of the desired and current joint angles and joint velocities, end-effector type and displacement.

The class has implemented methods for standard connect and disconnect events, and functions to send the desired joint angles, joint positions and end-effector displacement. All functions receive as parameters the desired values, which are verified to check whether the values fit in an acceptable range and if the number

of parameters is correct for the selected robot, before sending them to the server. Also the commands to retrieve information from the server, such as the current joint angles, joint velocities, end-effector displacement and type have protection conditions responsible for doing a quick check on the variables received. The communication class is also responsible for managing and sending to the virtual world, the surgical plan directives. It has implemented methods for sending, removing and showing the entire surgical plan, and singular commands for inserting, removing and selecting a surgical plan entity (target and trajectory).

In the UI function, a communication object is associated to each device (joint, end-effector control and surgical plan management), thus making the code more perceptible and easier to restructure. This communication classes are instantiated at the dialog create function, immediately after initializing the robot classes.

5.3.5 Client DBS

This class has variables and methods associated to intraoperative information and intends to be an initial approach to some basic robot functionalities towards an assistive intraoperative role. Once this class is instantiated, a key **procedure number** variable is associated so that each surgery is an unique event. The *Client_DBS* properties include a **target** and **entry position** variables used for trajectory addition to the surgical plan, the current *selected trajectory* along which the manipulator should move, the desired **initial distance** from the tip of the end-effector to the entry point. Additionally, the class counts with an **increment distance** variable that controls the distance to be covered by the end-effector at each descending or ascending movement within the selected trajectory. Another class property is the **control type** which allows the user to either move the end-effector by commanding the robot arm through velocity commands or by moving the end-effector instrumentation alone. Lastly, the class stores the surgical plan, which is no more than a list containing the surgical entities that can be manipulated through the UI and have direct impact on the surgical plan generated at the virtual world.

Among the implemented methods we can find, a trajectory generation that from entry ($\mathbf{p}_{net}$) and target ($\mathbf{p}_{tar}$) coordinates is able to generate an unit vector to define the tridimensional direction of the trajectory ($\hat{v}$).

$$\hat{\mathbf{v}} = \frac{\mathbf{p}_{tar} - \mathbf{p}_{ent}}{||\mathbf{p}_{tar} - \mathbf{p}_{ent}||} \tag{5.10}$$

It has some surgical plan manipulation functions and a method to generate the manipulator target position and orientation when provided the selected trajectory and the starting distance to the entry point. The desired position ($\mathbf{p}_d$) is easy to calculate, by subtracting the starting distance (L_{sd}) from the entry point along the selected trajectory.

$$\mathbf{p}_d = \mathbf{p}_{ent} - L_{sd} * \hat{\mathbf{v}} \tag{5.11}$$

The orientation however is a bit more complex to define, since the trajectory and instrumentation is cylindrical it only requires 2 orientation coordinates to be defined. Thus the robot end-effector can positioned concentrical to the trajectory at the desired cartesian coordinates with its last link revolving 360 degrees perpendicular to the trajectory.

To find the orientation, we followed the method one described in the Surgical plan subsection 5.2.2. However, the algorithm described only computes the initial θ and ϕ, thus positioning the end-effector z-axis concentric to the trajectory, left and center subfigures of figure 5.19. The end-effector can rotate along the new z-axis while still being collinear to the desired trajectory, therefore exploring new approaches for the manipulator using the Euler Angles Z-X'-Z" notation, figure 5.19 (right subfigure).

To explore all the possible orientations and after calculating the initial angles, we proceed to multiply the initial rotation matrix with a posterior rotation matrix around the new z-axis. The angle of rotation of this latter matrix should cover the $[0, 360^o]$ with an iteration step of 5 degrees[15]. We started by gathering all the

[14]**subsection 5.3.5** Free licensed media from Wikimedia Commons, with the authorship of DF Malan and edited by C. Faria `http://en.wikipedia.org/wiki/File:EulerG.png`

[15]**subsection 5.3.5**A lower iteration angle means that the best result can be closer to the

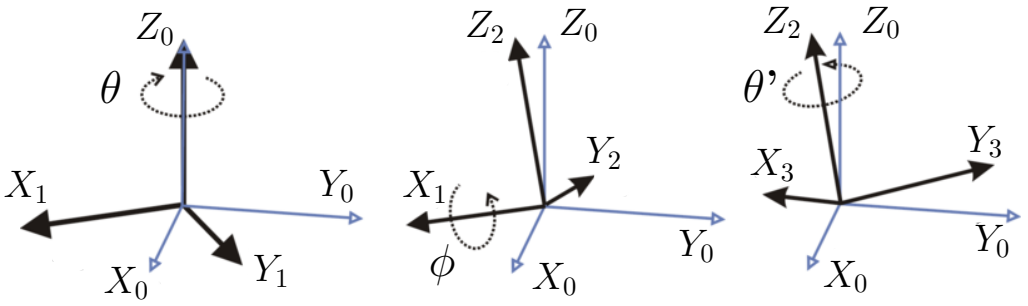


Figure 5.19: Euler Angles Z-X'-Z" convention[14].

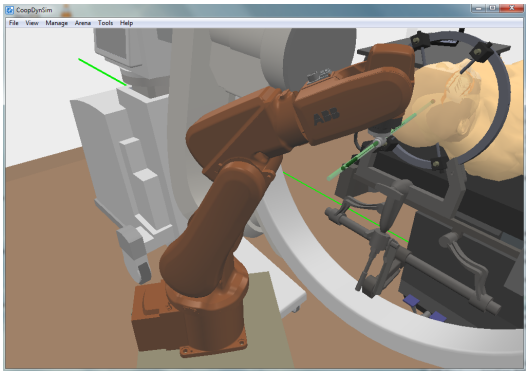
possible end-effector position and orientations. The geometric inverse kinematics is run for each solution to identify the feasible ones, which are later added to a new vector.

This new vector of only possible solutions is then submitted to a selection criteria, which was originally defined to achieve the least joint displacement from their home position (5.9). After some trials where the robot was expected to position along a desired direction, we realized that in several cases, the arm posture either obstructed or hindered the access to the instrumentation. Most of the time this was caused by a eccentric position of the wrist revolute joint (see figure 4.1f) commonly the penultimate joint. Based on the visual feedback, we changed the cost function, so the penultimate joint had the least displacement, eq 5.12 (where i is the index of each of the k solutions, and $n-1$ is the penultimate index of number of joints).

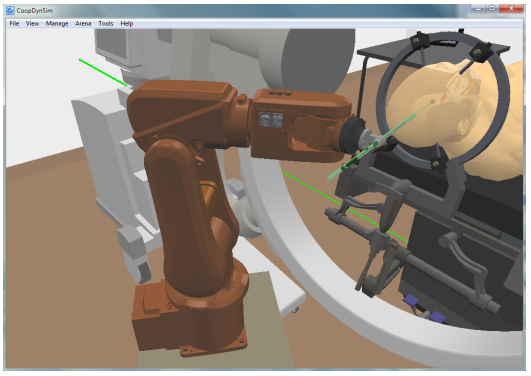
$$C = \min_{i=1,2,\ldots,k} \mathbf{q}_i(\theta_{n-1}) \tag{5.12}$$

By doing so, we are forcing solutions where the wrist revolute joint displacement is minimum, which also generates more natural postures and facilitates the access to the end-effector instrumentation, see example in figure 5.20.

optimal solution, however having a smaller step means more cycles and thus a greater computational demand. For such reason and after some tryouts, we decided that the iteration angle should be 5 degrees.



(a) Standard cost function, based in the minimum sum of all joint displacements.



(b) New cost function, based in the least displacement of the penultimate joint.

Figure 5.20: Different cost functions to decide the best approach orientation for the manipulator, considering the specified trajectory.

5.3.6 User Interface

In this last subsection we intend to provide some insight regarding the control application UI, since it is one of the best ways to explain what was developed and what can the user count on. Looking at the user interface figures 5.21 and 5.22, we can partition the application in 3 blocks: *i)* **Connections** interface, *ii)* **Developer** interface and *iii)* **User** interface. Before going into further details involving these blocks, we want to emphasize some global aspects.

Firstly, to help the user become acquainted to the application we left the fields where the user can input information with a white background color, while the fields that strictly display information are colored in a light yellow background. Also the toggle buttons that trigger a continuous **Update** of Geometric Direct Kinematics or Distance to target, are easily identified by a green color when toggled and with a red color when they are set to "off". The variables presented are either in millimeters for cartesian coordinates/distances or in degrees if they measure angular displacements. On the code behind the application, the angular values are treated as radians however to facilitate the user perception, the angles are presented as degrees at the UI.

Starting with the **Connections** interface, upon launching the YARP server and instantiating the world, robots, end-effectors and other equipment in the *CoopDyn-*

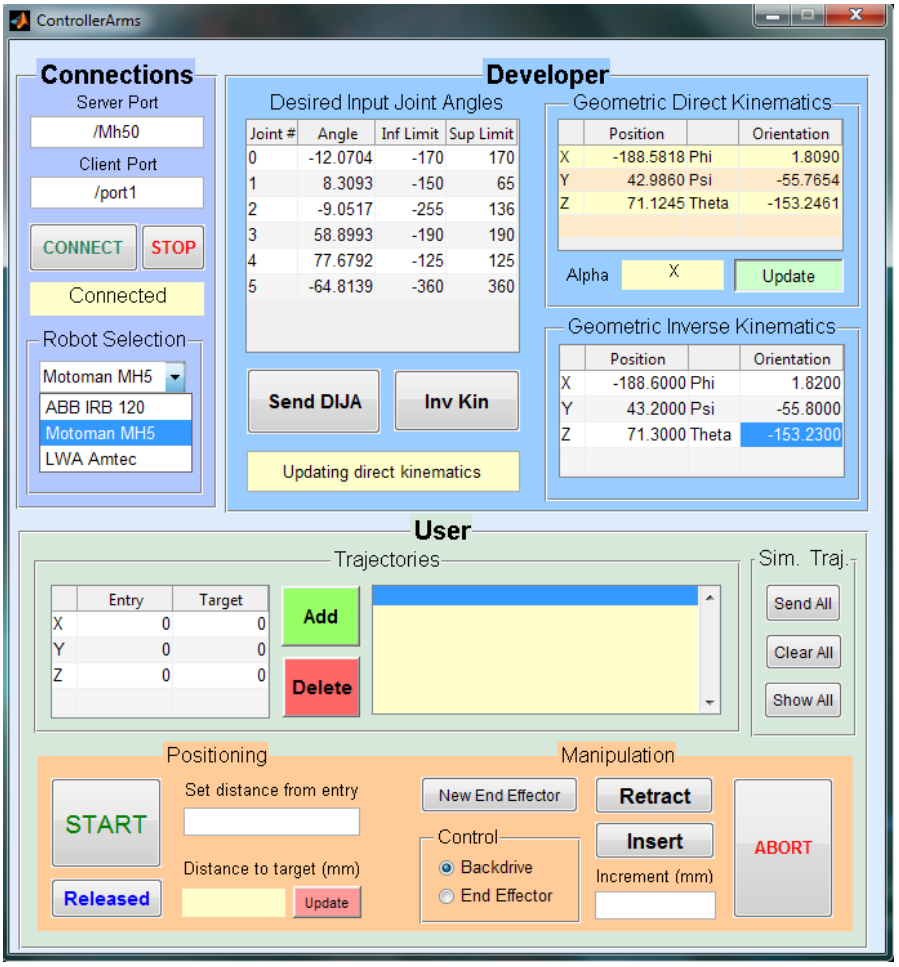


Figure 5.21: Control Application UI, Developer panel.

Sim robotics simulator, the server ports for each device are created and ready to be connected. In the client application, the user must insert the specific robot port to be controlled. The example presented in figure 5.21, the robot instantiated at the simulator was an *Motoman MH5* with the identification number 0, so the port name is "/Mh50". Since the *CoopDynSim* simulator associates 3 devices to each robot, it also creates 3 server ports for the inserted manipulator. Following a convention protocol, the server ports created are named after the manipulator reference name plus the name of each device:

```
/"Robot_name"+"Robot_number"/"Device_name"
```

so for the example presented in figure 5.21, the server ports are: "/Mh50/arm" to control the joints, "/Mh50/ee" to control the end-effector and "/Mh50/sp" to

manage the surgical plan. To simplify the interface for the user, the client only requires the robot name and number from where, it generates the 3 server port names and the correspondent 3 client ports. Also for the client port it only requires an unique port name, to which it assigns 3 client ports, like

```
/"Client_port_name"/"Device_name"
```

which in light of the presented example turns out to be: "/port1/arm", "/port1/ee" and "/port1/sp".

Upon indicating the server and client ports, the user can now connect the server to the client. The field in light yellow immediately below the **Connect** and **Disconnect** button, feeds the user with the current state of connection (either *Connected* or *Disconnected*). Below this connection state field, the user is expected to select 1 from the list of 3 available robot types. By selecting one robotic manipulator, the control application will automatically load all specific features for that robot and work only with the respective kinematic equations.

The **Developer** interface was initially designed to debug and test the control application in terms of developed features and implemented kinematic relations. It allows the user to specify the joint angles individually, to verify online the manipulator's end-effector position and orientation at the *Geometric Direct Kinematics* sub-panel, as well as the arm plane angle denoted as Alpha, for the 7 DOF Schunk LWA II robotic arm. For the 6 DOF robotic manipulators, it does not make sense to use the arm plane angle variable so it is omitted (displayed as "X" in figure 5.21). To test the geometric inverse kinematics, the user needs to specify a target position and orientation to be reached by the robot. After introducing both at the *Geometric Inverse Kinematics* sub-panel, the inverse kinematics functions are called by pressing the **Inv Kin** button and depending on the result returned the following report messages are shown in the status field below:

1. *Success*, if at least one viable solution was found;
2. *Ouside workspace*, if the specified target coordinates fall outside the robot workspace;

3. *All solutions exceed joint limits*, if the specified target coordinates fall outside the dexterous workspace, and despite the position being reachable by the manipulator, it can not be grasped from the desired orientation;
4. *Incorrect elbow angle*, if the elbow angle violates the joint limits for the 7 DOF robotic arm in all solutions (no solution available);

If the kinematic equations return *Success*, the application displays the best solution set of joint angles in the *Desired Input Joint Angles* sub-panel. From here and likewise the process to send directly desired joint angles, the user must press the **Send DIJA** button to send the set of angles through communication to the simulator, or eventually the real robot.

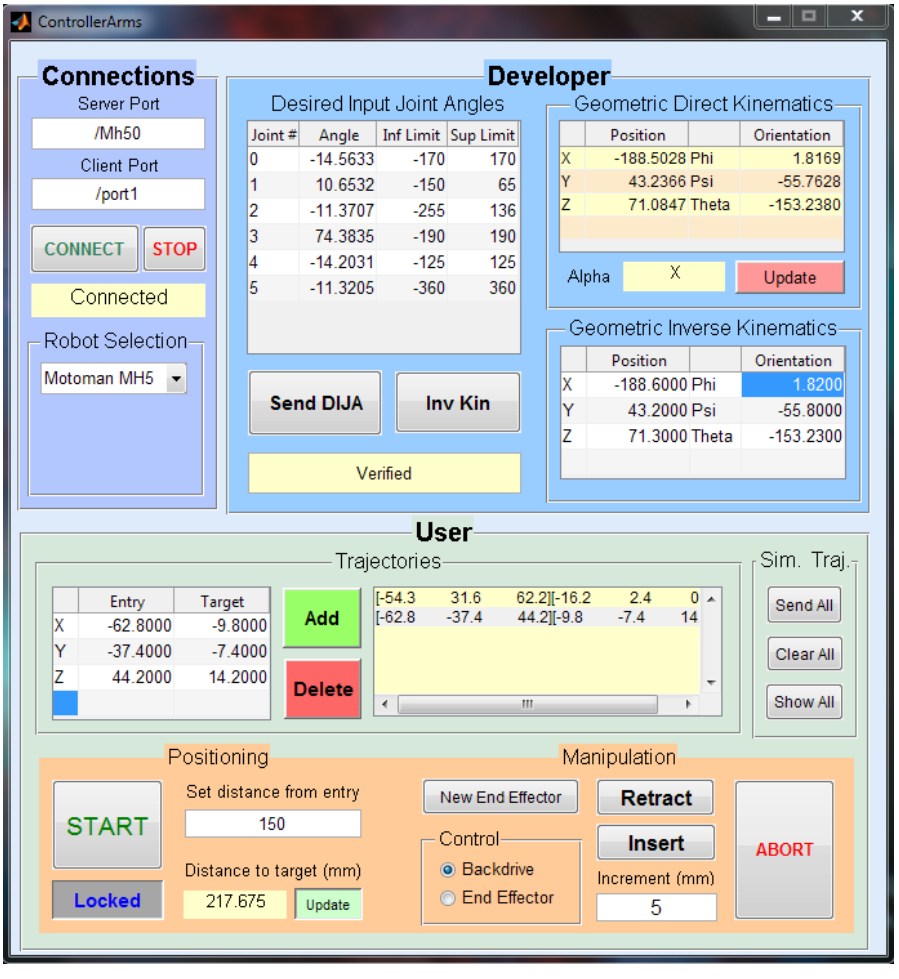


Figure 5.22: Control Application UI, User panel.

The **User** interface is an initial attempt to devise a control application specif-

ically oriented towards robotic assisted DBS. Although some of the implemented functionalities are exclusive of the simulated environment, most of them can be ported to a real robot. Starting by the *Trajectories* panel, at the top left there is a table where the user can introduce the entry and target coordinates from the preoperative plan. By pressing the **Add** button, the coordinates previously inserted are added to the surgical plan list (also depict in figure 5.22 to the right of the **Add** button) and sent to the simulator where it draws the surgical plan trajectory and target. The **Delete** button allows the user to remove a highlighted item from the surgical plan list. By selecting (highlighting) an item in the surgical plan list it tells the manipulator to follow the chosen trajectory and communicates to simulator to hide the remaining trajectories, drawing only the selected.

The *Sim. Traj.* sub-panel gathers a set of functions exclusive to the interaction with the simulator. The **Send All** button lets the client user, communicate the entire list of surgical plan at a time for the client to draw. On the other hand, the **Clear All** button removes all surgical plan entities from both the simulator and the client application. The **Show All** button as the name says, reveals all hidden trajectories on the simulated world. This set of functionalities, come really handy since both softwares are independent so theoretically one can set all the preoperative plan on the client application and only later on, connect to the server (simulator or real platform) and send the information.

Having selected the desired trajectory, the next step is to define the distance to the entry position. The user can set this distance at the *Electrode Insertion* sub-panel, which in figure 5.22 is set to 150 mm. Below the distance input field, there is a toggle button to continuously check the distance from the end-effector to the target. By pressing the **Start** button, the control application computes the geometric inverse kinematics to place the end-effector at the desired distance from entry along the selected trajectory. Once again, the manipulator may not be able to reach the desired coordinates so it returns the same messages as enumerated for the geometric inverse kinematics. When it is successful it inputs the desired joint angles in the *Desired Input Joint Angles* table. The **Released**/**Locked** toogle button is part of a safety routine. When locked, the robot is not allowed to move

away from the currently selected trajectory. It can only move its end-effector collinear to the defined direction and even the path points must be coincident to the trajectory. More details regarding this functionality are discussed in subsection 5.3.7.

In the *Manipulation* sub-panel we will start by talking about the **New End Effector** button. It basically checks the communication to grasp what end-effector is currently attached to the robot. Different end-effectors have different sizes and may or not have movable parts, information that is vital for the control application to understand how to handle the manipulator control and kinematics. The **Retract** and **Insert** buttons withdraw or approach the target, respectively, by an increment distance defined in the field below. The movement can be executed using a **Backdrive** control to simulate the floating mode throughout velocity control (Differential Kinematics) or by **End-effector** control. In this last control option, only the end-effector actuator is controlled (this option is exclusively available for actuated end-effectors).

Finally, the **Abort** button stops the current motion sequence and stops the robotic manipulator actions. This event was created to be launched in emergency situations or to prevent any damages, when the robotic manipulator is not behaving as expected.

5.3.7 Safety

Safety should always be one of the pillars of any system towards surgical activity. As stated by Lavellée et al. [84], safety was and still currently is the fundamental reason why robotic systems are so difficult to include in an operating room and be accepted/trusted by a medical team. As referred in chapter 2, the safety concerns should be addressed at all stages of the system development. We listed some of the desired safety features that meet the interest of this project like movement restriction, precision control collision detection and system log. Most of these were implemented in the simulated environment while others like collision detection were impossible to achieve in time, however the algorithm we envisioned to develop them

will be described nonetheless.

Movement restriction We split the whole motion planning of robot actions in 2 stages: the positioning and the manipulation phases. At the positioning phase, the robot moves to a desired position along a planned direction (trajectory) to hold the instrumentation. In this step, it should only avoid collisions with the surrounding environment (people and equipment) and precisely achieve the target position and orientation. The movement restriction problem is thus limited to collision avoidance.

In the manipulation stage, the robot moves the instrumentation strictly along the defined trajectory either by arm joint action or through tool manual actuation. Also at this stage, the robot is expected to adapt its functioning based on collision avoidance routines. But the implemented feature we wanted to emphasize is the motion restriction when the computed solutions are not coincident with the selected trajectory. Since the motion restriction should be a behavior exclusive of the manipulation stage we created a toggle event button **Released**/**Locked**. When released the robot is able to freely maneuver, while at the Locked state, it actively restricts any motion outside the selected and locked trajectory.

It was implemented by controlling the joint angles passed by communication. Before sending the desired joint angles, the control application simulates the final end-effector position and orientation through Geometric Direct Kinematics. If either the expected position or orientation fall outside a safety range around the selected trajectory, the communication no longer sends the angles to the robot and notifies the user of the safety issue. To move outside the selected trajectory, the user needs to toggle the motion restriction to Release the trajectory.

Another movement restriction policy implemented was the imposed limits on the joint velocity actuation. The joint controller should be aggressive ,i.e, higher gains to assure that the target is reached with utmost precision. On the other hand, higher gains imply higher inertia and velocity so we introduced a limit velocity of 5 deg/s to every joint.

The safety measures implemented also include the expected robot behavior

with the different end-effectors. Specifically, the trepan instrument should strictly be used to open a burr hole in the patient's skull. Therefore we established a condition to avoid the trepan to go beyond the entry point, thus avoiding any injuries even if the user erroneously commands the robotic arm to move further. This control is implemented through a simple euclidean distance check[16],

$$trepan\_check \ = \parallel \mathbf{p}_{EEPosition} - \mathbf{p}_{Target} \parallel \geq \parallel \mathbf{p}_{Entry} - \mathbf{p}_{Target} \parallel \qquad (5.13)$$

A simple condition like this suffices, because previously the controller checked if the instrumentation position and orientation are collinear to the desired trajectory. The robot only moves if both conditions are met.

Finally, the robot posture is invariant during the whole manipulation stage (as suggested by Lavallée et al. [84]). During positioning, the motion plan cost functions, choose one optimal posture over the others possible (at most 4). As the robot passes to the instrumentation manipulation stage (trepan drilling, probe inserting), the robot posture is fixed and next position and orientation points can only be reached with that posture.

Precision control The precision control is closely related to the motion restriction, and its implementation follows the same methodology. In fact the precision control is also a two step process: i) the evaluation of the generated solution precision, specifically position and orientation; ii) assessment of the result robot joint angles distance to the expected values[17]. To better explain how the algorithm was implemented we present some key variables in figure 5.23.

The broken line represents the desired trajectory to be followed, where the almost parallel segment represents the probe position (or any other instrumentation used). Since the instrumentation can assume several positions along the selected trajectory, our first concern was to measure how collinear to this trajectory were the computed solutions.

[16]**subsection 5.3.7** Returns 1 if the movement is safe and 0 otherwise.

[17]**subsection 5.3.7** We are considering that the robotic joint sensors are able to return trustworthy feedback.

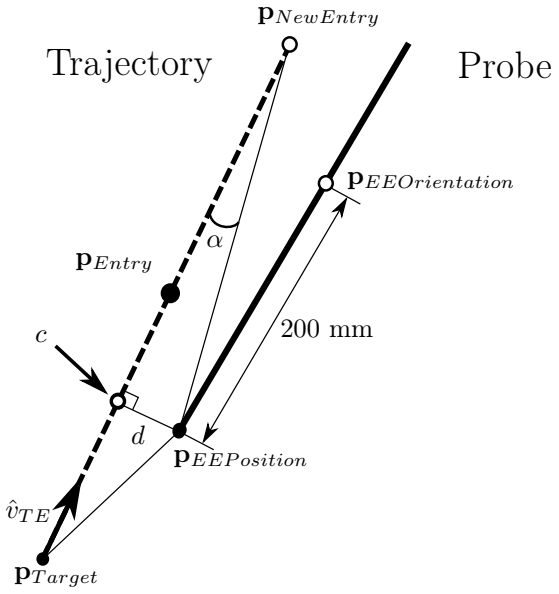


Figure 5.23: Representation of the precision control algorithm variables.

The generated joint angles solution or sequence of joint angles[18], is firstly converted to end-effector/instrumentation final position and orientation variables through Geometric Direct Kinematics. To see how precise are the generated solutions we will measure the distance of the end-effector tip position relative to the closest point of the desired trajectory (d).

We started by using the trajectory unit vector $\hat{v}_{TE}$ from the target coordinates ($\mathbf{p}_{Target}$) to the entry coordinates ($\mathbf{p}_{Entry}$), to obtain a new point ($\mathbf{p}_{NewEntry}$) whose distance (L) is fixed to the target coordinates along the trajectory vector,

$$\mathbf{p}_{NewEntry} = \mathbf{p}_{Target} + (\hat{v}_{TE} * L) \tag{5.14}$$

We want to reduce a much as possible the problem variables, and this was the thought method to avoid dealing with a variable distance from the target to the entry. With the $\mathbf{p}_{NewEntry}$, $\mathbf{p}_{Target}$ and end-effector position point ($\mathbf{p}_{EEPosition}$) known, we form a triangle in 3D space. The shortest distance between the $\mathbf{p}_{EEPosition}$ and the trajectory chosen (d), forms a normal line to the trajectory whose coincident point will be henceforth called "c".

[18]**subsection 5.3.7** If the movement of the robotic manipulator is defined after a set of path points.

Considering,

$$\begin{cases} \overrightarrow{v}_{ne\_c} = c - \mathbf{p}_{NewEntry} \\ \overrightarrow{v}_{ne\_eep} = \mathbf{p}_{EEPosition} - \mathbf{p}_{NewEntry} \\ \overrightarrow{v}_{ne\_t} = \mathbf{p}_{Target} - \mathbf{p}_{NewEntry} \end{cases}$$

from basic geometry we know that,

$$\cos\alpha = \frac{||v_{ne\_c}||}{||v_{ne\_eep}||} \tag{5.15}$$

and based on the dot product geometric interpretation,

$$v_{ne\_c} \cdot v_{ne\_eep} = ||v_{ne\_c}||\ ||v_{ne\_eep}||\ \cos\alpha \tag{5.16}$$

By simple algebraic manipulation we get that,

$$||v_{ne\_c}|| = \frac{v_{ne\_c} \cdot v_{ne\_eep}}{||v_{ne\_t}||} \tag{5.17}$$

Using the Pythagorean theorem, we can compute (d) by,

$$d = \sqrt{||v_{ne\_eep}||^2 - ||v_{ne\_c}||^2} \tag{5.18}$$

and thus get the closest distance between the tip of the probe and the selected trajectory. Then a safety threshold can be defined to mark the safety region around the desired trajectory. If the "d" is inferior to the threshold, the end-effector tip position is considered safe.

However simply having the probe tip close to the trajectory does not mean the instrumentation is correctly placed, since the orientation can be inaccurate. Unlike position, the orientation coordinates are harder to compare because there is several sets of coordinates that grant a safe solution. Additionally the orientation coordinates change along the trajectory as the end-effector gets closer to the target.

Confronting this problem we made use of the instrumentation linear profile and instead of comparing the orientation coordinates, we extrapolated the instrumentation position 200 mm above the tip along the end-effector orientation (depict in figure 5.23 as $\mathbf{p}_{EEOrientation}$) and measured its distance to the selected trajectory.

In order to find the $\mathbf{p}_{EEOrientation}$, we started by computing the Roll-Pitch-Yaw rotation matrix from the orientation coordinates, $\mathbf{R}_e$. The orientation coordinates ($\mathbf{R}_e$) set the end-effector z-axis along the desired trajectory. Therefore, with the 3^{rd} column of the rotation matrix ($\mathbf{R}_{e,z}$) we can compute $\mathbf{p}_{EEOrientation}$ by,

$$\mathbf{p}_{EEOrientation} = \mathbf{p}_{Target} + (\mathbf{R}_{e,z} * -200) \tag{5.19}$$

With the point coordinates, we can follow the same procedure described for $\mathbf{p}_{EEPosition}$ to check the distance to the desired trajectory. The precision condition is only verified if both the position and orientation are fit within the safe limits.

Actions history log One of the steps of Fei's [58] systematic method to evaluate/improve safety conditions referred the use of a log system to record all the robotic equipment actions and controls sent. It is a great tool to evaluate not only the robot performance, but also to study how the medical team is using the equipment, potentially correcting some misuses.

The control application was developed so that each surgery or procedure carried out, would be labeled as a single event independent and distinct from all others. Following the same philosophy, a new system log file is generated for each procedure. The generated log files have a universal *.txt format. At the heading, the procedure key number is registered, along with the current date and each event is saved together with the current system time.

The log keeps track of the connection success between client and server, the selected robot type, the manipulation control mode and end-effector tool installed. It saves the joint angles sent during single or sequential motions and for both of them, it registers if the movement is executed on a safe or free maneuvering routine. Finally, the log file also contains surgical plan managing actions like adding, deleting or selecting the pre-planned trajectories.

This initial solution can not be considered final however, because a *.txt file can be edited which compromises the registration reliability.

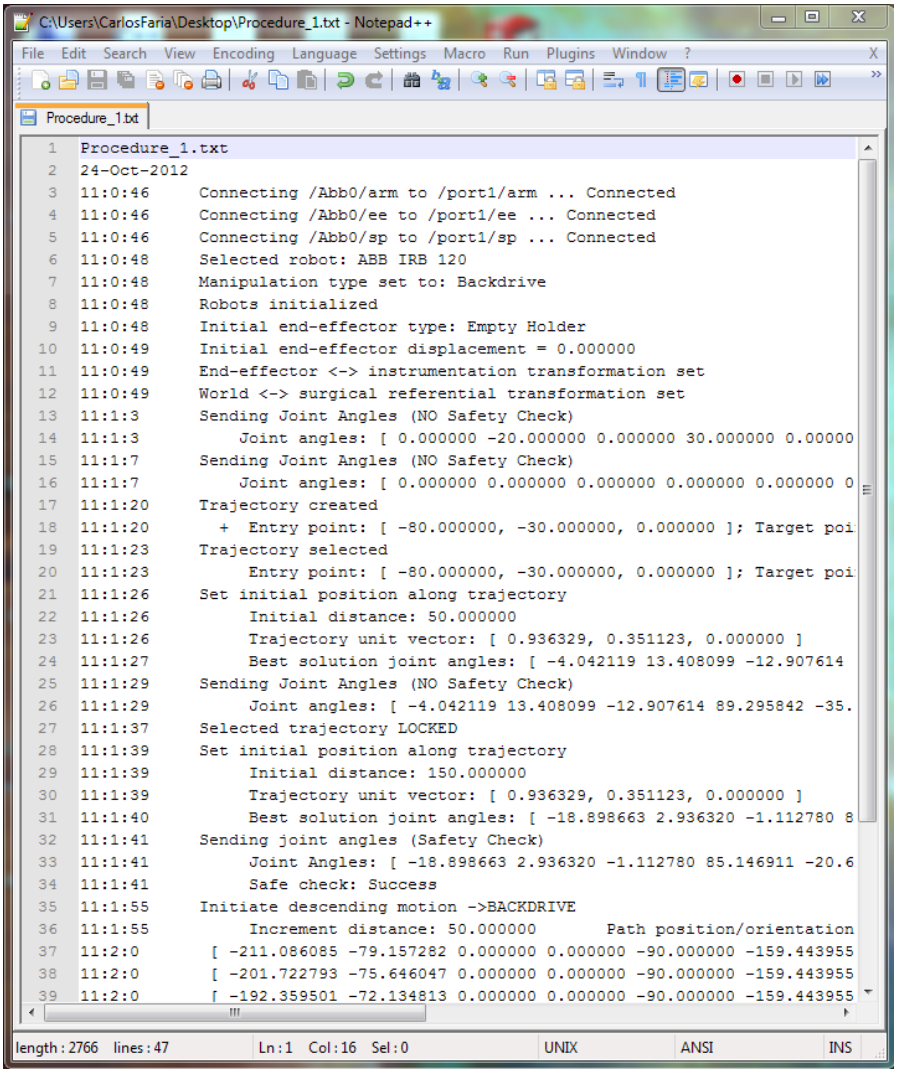


Figure 5.24: Example of a typical log file.

Collision Avoidance Despite not having a final solution implemented for collision avoidance we thought of a simple and realistic method whose main idea is described below. We split collision detection into 2 sub-problems: i) relative to a static environment where the robot manipulator is placed and ii) the other associated to moving elements within the robot workspace. To implement collision detection routines we need a stereocamera to record tridimensional images of the environment.

For the first part of the problem and considering that the equipment surrounding the robot is static during the surgery, we only need a tridimensional representation of the environment previous to the operation. The vision system

identifies the objects in the robot workspace and assigns a cloud of points or lines to represent their limits.

To check if the robotic arm collides with the surrounding environment we need to assign reference points along its arm and to each point indicate a radius of collision. Then we check if the object limits are within the collision radius of each point and if so, launch a collision warning. Since reference points are assigned to the robot links, they move along with the manipulator. To brief this problem, we considered saving each point information as a transformation matrix from the preceding joint (${}^{m}\mathbf{T}_{ref}$). By doing this we can calculate each reference point transformation from the robot base frame (${}^{0}\mathbf{T}_{ref}$) by,

$$ {}^{0}\mathbf{T}_{ref} = {}^{0}\,\mathbf{T}_{m}\ {}^{m}\mathbf{T}_{ref} \tag{5.20} $$

being ${}^{0}\mathbf{T}_{m}$ the transformation from the robot base frame to the m joint. One potential problems with this approach is the tradeoff between a having an object representation closer to reality with more points but at the expense of a higher computational demand, since the number of collisions increases exponentially. To avoid checking all the potential collisions, we can divide the robot workspace into smaller subspaces, and only test collisions with points that share the same subspace.

The task of obstacle avoidance gets more complex with moving objects within the robot's workspace. The computational resources and the optimization of algorithms are even more critical in this problem, because the robotic controller is expected to answer as quick as possible to an obstacle. This implies that the vision system object vertices computation and the collision detection algorithms must be quick enough to search for collisions with an acceptable refresh rate. Here is another reason for using a low velocity profile on the robot joint actions because during the time interval when the controller checks for collisions and actually launches a warning the manipulator movement is smaller.

Granting that the detection algorithm is quick enough, how should the robotic manipulator react to a potential collision event. Once again we chose to split

the manipulator motion planning in positioning and manipulation. During positioning stage, the only requirement is for the robot to achieve a final precise position/orientation and so if the current posture leads to a collision the robot may opt by an alternative collision-free posture to reach its goal. If no alternative is viable or is collision free, the robot should stop its motion and promptly notify the user.

On the manipulation stage however, the robot is committed to a specific trajectory, from which it must not deviate even to change its posture. In this phase, the robot should not adapt its trajectory to avoid collisions. Even before executing the motion, it checks for collision with the static environment, during the movement if any obstacle crosses the robot path, it should simply stop, and notify the user for the collision possibility. This way, the robot motion is consistent and always predictable during the manipulation phase.

Part III

Conclusions

Chapter 6

Results

With the acquired knowledge about the DBS surgery and using the simulation tools developed, we devised a method to test how each robotic manipulator suited the operating room and how aptly it accomplished the assigned tasks. In chapter 3, we analyzed the different robotic features provided by the manufacturers and labeled the key characteristics sought for the "perfect" neurosurgery robot. However and as said previously the consequences of features like workspace, degrees of freedom, joint rigidity and horizontal reach, on an assistive role can not be reviewed exclusively on numbers.

Assessing these variables and the robotic control algorithms was our main focus on gathering results, since task precision and repeatability can not be evaluated due to the lack of information about the real robots controllers and functionalities. The virtual robots evaluation was carried based on visual feedback of how each system carried out different tasks. Most of the findings became apparent after numerous experiments which makes it hard to represent in paper format. Therefore we chose to comment and discuss along with the results to emphasize the conclusions and point out other, not so evident, aspects.

6.1 Fitting in the Operating Room

Where and how to position the robot regarding the operating room arrangement and equipment/personnel positions? This was the question that we tried to answer

along this section. The formulated method evaluates how successfully the robot reaches a designated position and orientation, while measuring if it can descend the end-effector instrumentation collinear to the selected trajectory. We tested a wide range of trajectories (figure 6.1) that cover the area where the robot is expected to hold the instrumentation as an assistive agent during a DBS surgery, and even further.

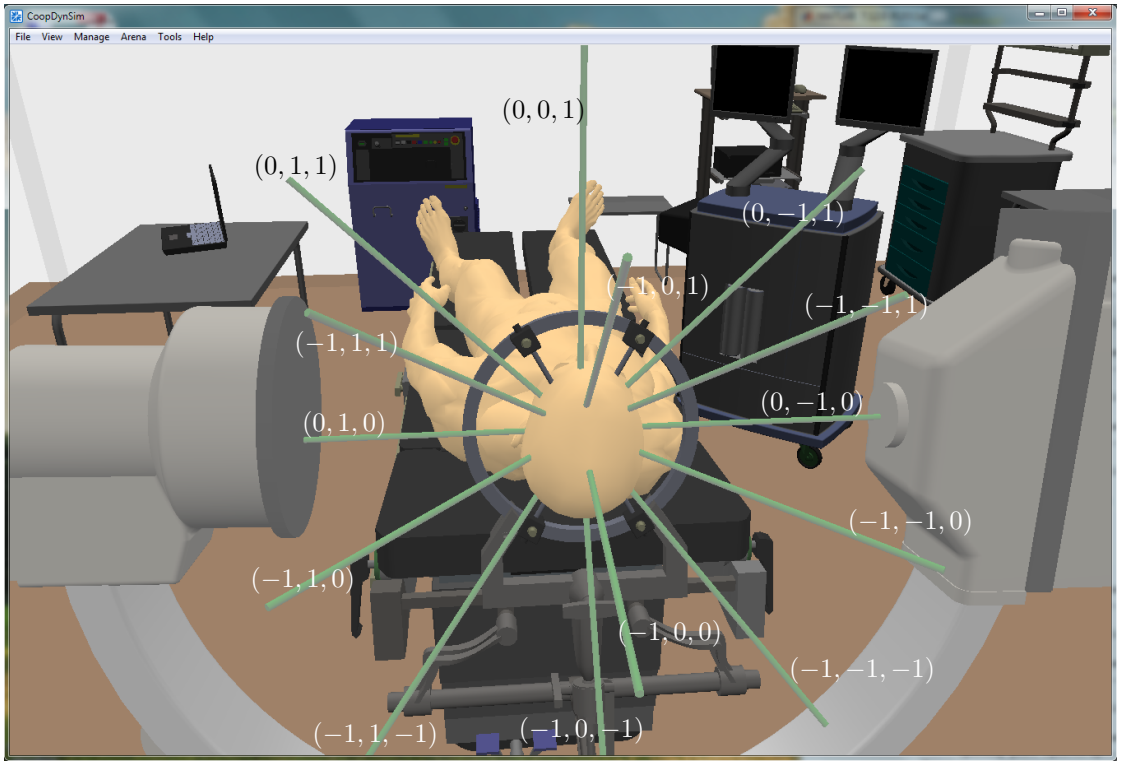


Figure 6.1: Generic trajectories to be reached by the manipulator.

Figure 6.1 depicts each trajectory tested, along with the correspondent vector coordinates that originated them on the operating room environment[1]. Each trajectory vector starts from the same point within the patient's model head, placed to simulate deep brain targets. In a real DBS surgery, there rarely is only a single target and their position changes from patient to patient. However in the robotic control domain, the consequence of reaching different targets distanced a few millimeters from each other (figure 1.5) while maintaining the same entry position,

[1]**section 6.1** The simulation environment also includeds the robotic manipulator, but it was omitted in this case to avoid obstructing the view of the trajectories.

has a minimal impact on the trajectories generated. Therefore we chose to keep the target coordinates constant and vary the entry point position.

To simplify the notation of the trajectories they will be represented by different cells as in the odd-shaped table, figure (6.2).

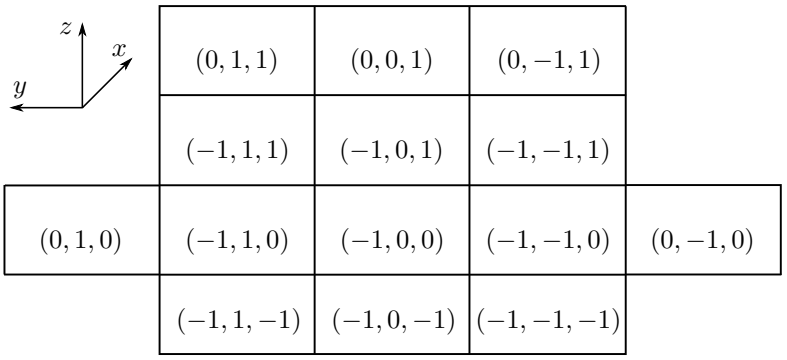


Figure 6.2: Generic trajectories representation.

Due to the large number of tests and to make the results easier to read, we will be using a color code to represent whether the robot could reach and move along each trajectory, table 6.1.

Table 6.1: Simulation results color code.

Color	Description
	Positioning the end-effector at 150 mm from the entry point, and advancing the end-effector collinear to the trajectory until reaching the target.
X	Positioning the end-effector at X mm from the entry point, and advancing the end-effector collinear to the trajectory until reaching the target.
	The manipulator posture, obstructs the access to the end-effector and/or prevents instrumentation handling.
	The robotic manipulator can not reach the defined trajectory.

For easier understanding of what the Color code (table 6.1) descriptions stand for, in figure 6.3 we depict the initial position of the manipulator at 150 mm (green color) from the entry point and the consequent 100 mm and 50 mm distance positions (yellow color), for an example trajectory.

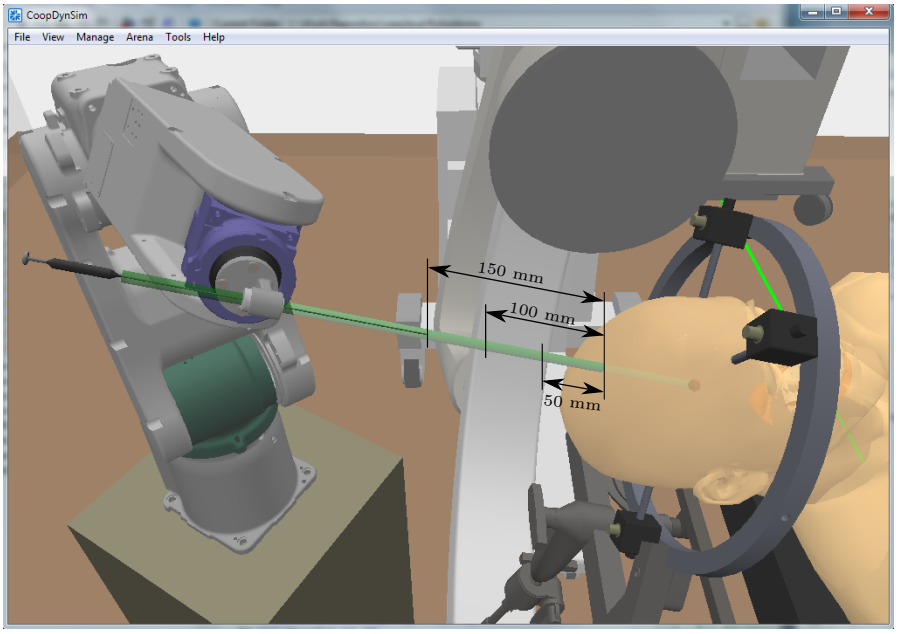


Figure 6.3: End-effector positioned at X mm from the entry point collinear to the desired trajectory.

To find the adequate mobile platform height and the optimal position for the robot within the operating room we tested several combinations. Starting with the platform where the robotic manipulator is mounted, its shape was not addressed in this project, even so we can anticipate its dimensions. Relative to the dimensions parallel to the ground, it is desired to occupy as less space as possible but it still needs to be large enough to provide a stable base for the manipulator. These proportions do not seem to affect the robotic manipulator performance, however the same can not be said for the height. The effectiveness of the manipulator approach to the selected trajectories was one of the main criteria for the decision of the base platform height. We started by testing 4 different heights for the robotic manipulator: i) -100 mm, ii) 0 mm, iii) +100 mm, iv) +200 mm. These values stand for the difference between the end-effector tip and the target position along the Z-axis (height). At the end of the tests and due to the success of the +200 mm height, we decided to assess how the robot behaved for +300 mm and add this option to the results.

To study the best manipulator position we started by positioning the robot

in front of the patient with its arm along the world X-axis and the end-effector around 100 mm away from the patient's head as in figure 6.4.

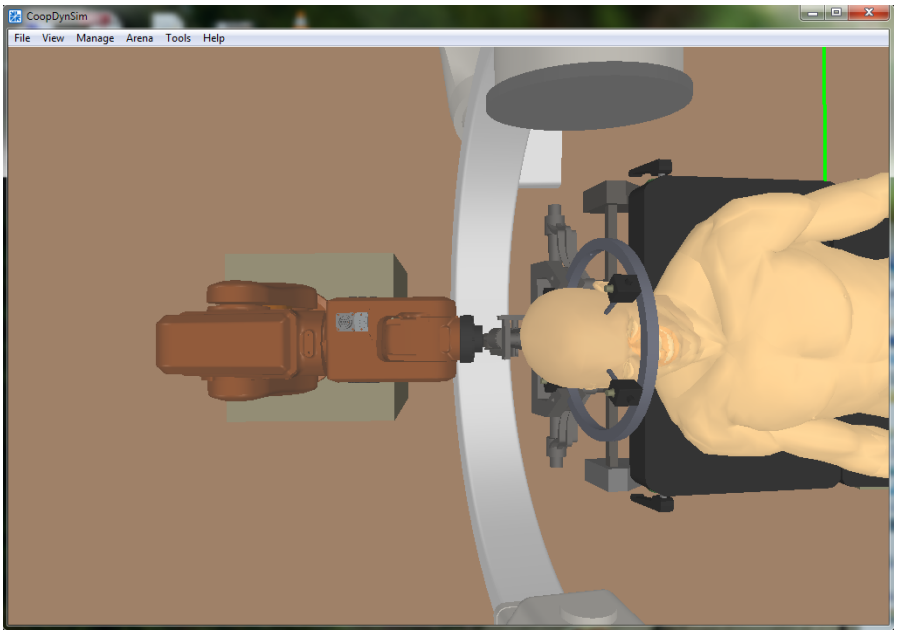


Figure 6.4: Initial position for testing the robotic manipulator placement.

We distanced the robotic manipulator from the patient's head so the arm movements would not collide and potentially hurt the patient. Besides this configuration which shall be denoted as 0^o, we tested the manipulator performance at $+15^o$, $+30^o$ and $+45^o$ angles. Each angle corresponds to the manipulator orientation around a central point fixed on the patient's head while always keeping a distance of about 100 mm from the same. We only test positive angles because the manipulator structure is symmetric, so the results would be exactly mirrored when the angle was negative. The choice of the $[0, 45]$ degrees range has to do with the workspace limitation caused by the mobile X-ray imaging system also depict at the figure 6.4. The last detail we want to point out is that in terms of the trajectory representation (figure 6.2), as the robot position angle increases its end-effector gets nearer the trajectories on the left side of the odd-table.

6.1.1 ABB IRB 120 results

Starting with the ABB IRB 120 manipulator, we expected this robot to display the worst results since its horizontal reach is the shortest among the selected robots. The following results were gather:

Figure 6.5: ABB IRB 120 trajectory reaching results.

Our initial prediction pointed to the 0 mm height, 0^o orientation as the best configuration since it was supposed to be the closest to all trajectories. We soon realized that the ABB manipulator at this height was not capable of reaching most of the pre-defined trajectories. Not only were the top trajectories unreachable, also the trajectories immediately below are achieved at the limit of the robot range, or in other words when the arm was fully stretched. This concerns us regarding the stability and movement resolution of the robot when driven to these limits, since a superior precision is required. At 0 degrees, the trajectories reached are

as expected symmetrical for the left and right sides of the representation. Aside from the limited reach of the outer trajectories, at the 0^o and $+45^o$ the robot configuration for the $(-1, 0, 0)$ and $(-1, 1, 0)$ trajectories respectively restrained the possibility to manipulate instrumentation. These results marked as orange, mean that while the manipulator is in fact able to reach and move along the target trajectories the end-effector instrumentation collides with the preceding link for all the configurations possible.

As we increased the robot position angle, the robot becomes unable to reach the outer trajectories (when the vector y-axis value is negative), begins reaching the most proximal trajectories (when the vector y-axis value is positive) and the central trajectories remain unchanged. When we lowered the arm height by 100 mm, the ABB manipulator was incapable of reaching most of the trajectories even the central ones[2] considered as the most likely to be chosen in a DBS surgery. Therefore, a robotic system that is unable to match the flexibility of the currently used mechanical stereotactic frame is held as a step backwards and would largely discourage medical teams to acquire and accept it. We aim to provide a better solution in terms of flexibility and precision, so the manipulator should at least be able to flawlessly reach and move along the central trajectories. This condition immediately excludes all the 0 mm and −100 mm height ABB possibilities for any robot position angle.

Following the same criteria, the possibility of using a height of +100 mm with a $+45^o$ position angle is also excluded. The +100, +200 and +300 mm present quite similar and the best results. There is a clear tendency relating the increase of height with an increase of the number of reachable trajectories, which eventually lead us to test the +300 mm possibility not programmed originally. With an height of +100mm, the ABB robotic manipulator can successfully reach the central trajectories and even has some margin to work on outer coordinates. However for the upper central trajectories, the arm is almost completely stretched which raises the stability problem referred above. At the +300 mm, we noticed that the

[2]**subsection 6.1.1** The $(-1, 1, 1), (-1, 0, 1), (-1, -1, 1), (-1, 1-0), (-1, 0, 0)$ and $(-1, -1, 0)$, refer to figure 6.2.

manipulator despite reaching almost all trajectories had some problems positioning for the central ones due to the proximity between the target position and the inner limits of the manipulator workspace. That is noticed in the results for the $+45^o$ and $+30^o$ angles, marked in orange. Not only that, but also the arm configuration at this height required the neurosurgeon to work from below the robot links, which may cause some discomfort and unable a clear view of the operative field.

For the reasons cited above, we decided that the ABB IRB 120 robot platform should take into account position of the patient before the surgery and optimally place the robot end-effector around 200 mm above the patient's head. In terms of the position angle, the somewhat unexpected results revealed that the robot body configuration only varied slightly to reach the same trajectories from different angles. Acknowledging that, we purpose to position the robot to the side, around $+45^o$ to $+30^o$ so the neurosurgeons can have more workspace available.

During the experiments, we noticed that the ABB manipulator configuration and motion path rarely conflicted with the intraoperative equipment or the patient. This conclusion was unexpected since the manipulator flexibility on reaching the desired trajectories depends on the proximity of the robot to the patient, thus sharing a small space with other equipment.

6.1.2 Motoman MH5 results

Considering the horizontal reach, link dimensions and the overall characteristics of Motoman MH5, we expected it to be more flexible and able to reach further from the same position, when compared to ABB. However the link length also raised concerns regarding the possibility of collisions with the equipment or the patient. The results are displayed below:

Starting at the 0 mm height, the robot was able to reach the central trajectories, when its position angle was between $+0^o$ to $+15^o$. When the position angle increases, the $(-1, -1, 1)$ and $(-1, 0, 1)$ trajectories become unattainable for a starting distance of 150 mm along the trajectory. In this cases the manipulator needs to extend to its limits which drives us to a potential instability

Figure 6.6: Motoman MH5 trajectory reaching results.

problem. Even for the lower position angles at this height, where the manipulator comfortably reaches the most central trajectories, the arm configuration restrains instrumentation handling for trajectories around the vector $(-1, 0, 0)$. Based on these reasons, we discourage placing the robot at a 0 mm relative height. When the manipulator end-effector is lowered by -100 mm, no orientation allows the robot to reach all the central trajectories which directly sets this possibility as non-viable.

The remaining solutions for +100, +200 and +300 mm relative heights, the manipulator is generally capable of comfortably reaching all the most common trajectories. At +100 mm height and for lower position angles (0^o to $+15^o$), the robot can easily reach the most common trajectories. Increasing the position angle made the MH5 manipulator extend the arm to its outer limits to reach the right side trajectories, which can be problematic. At +200 mm the manipulator is in general able to reach all the central and even some outer trajectories successfully.

On the +300 mm mark the results in terms of trajectory reaching are similar to the +200 mm height, however at this point the manipulator configuration hinders the instrumentation substitution and handling from the part of the neurosurgeon.

As a result we concluded that the optimal manipulator height should be around +100 and +200 mm. If the robot can occupy any position (angle) within the available workspace, both heights are viable. However if the space needs to be shared by the robot and the neurosurgeons team, we suggest to place the manipulator at a relative height of +200 mm, at an angle of around $+30^o$ to $+45^o$.

The MH5 manipulator, despite having larger links it does not reflect on collisions with intraoperative equipment. In fact, the only issue with the robot, had to do with the robotic links movement path when moving between trajectories on opposite sides of the of the patient. This problem can be abbreviated by making the robot follow strategic safety points along its trajectory to guarantee that it avoids collisions.

6.1.3 Schunk LWA II results

From the 7 DOF structure of the manipulator and the length of its links, we foresaw that this robot would be able to reach more trajectories than its 6 DOF alternatives. The actual results however, overcame our expectations, and the manipulator can effectively achieve the central coordinates for almost any combination of height and position angle.

It only has problems reaching the trajectories when the manipulator is set to -100 mm height. The extra DOF grants it the flexibility to avoid situations where the instrumentation intercepts the robotic body or avoid poses where the instrumentation handling by the neurosurgeon would be hindered. The Schunk LWA robot is however more problematic because its body easily collides with the equipment or the patient. However, since the robot can reach the same trajectory from several configurations, we can make use of that property to set its pose based on collision avoidance parameters. Implementing this feature requires the robot to have a constant perception of the surrounding elements position, perhaps resorting

Figure 6.7: Schunk LWA II trajectory reaching results.

to visual feedback of the environment.

Regardless of achieving the best results for this test mainly due to the extra DOF, this same characteristic is also a setback as the end-effector precision and repeatability is dictated by the cumulative error on the displacement of each joint in a serial manipulator. Despite not being accounted on this test the precision still plays a major role on the final system and so if the superior flexibility of a 7 DOF manipulator is desired, it needs to be accompanied by a greater investment on precision actuators and control algorithms.

6.1.4 Comparative analysis

Before going into detail on comparing all three robotic systems, we want to address some potential critics to the test and the results attained. Firstly, why so

few trajectories, heights and position angles tested? This decision is based on 2 reasons: i) the manipulator joint displacements and the arm configuration change little and predictably from trajectories near one another, considering the several combinations; ii) there needs to be a limited number of trajectories since we are testing so much parameters combined (height, position, trajectories and different manipulators). Therefore we can consider that each trajectory represents a larger area where the manipulator reaching configuration is similar. Adding further combinations largely increases the number of tests, that even now still ascended to 840.

Despite testing 14 trajectories, the viability of the height and position angle combination is decided as whether the manipulator can effectively reach the 6 central trajectories. These trajectories cover the area where the end-effector should most commonly be placed to hold the instrumentation throughout a DBS surgery. Nonetheless we still tested for the outer trajectories to anticipate how the robot would behave in less common surgery plans. Moreover if we want to broad our horizons and apply this robotic technology in other types of neurosurgeries, we can understand how the manipulators would perform on more eccentric trajectories. For instance, we noticed that literally in every combination of height, position angle and robot model the 3 lower trajectories were always successfully reached. This conclusion tells us that the robotic models selected are specially apt in assisting, manipulating and holding instrumentation for skull base surgeries.

Comparing the results of this test, the Schunk LWA II achieved the best results followed by Motoman MH5 and the last one was the ABB IRB 120. By best results we mean that the robot is more flexible in terms of reaching trajectories and can reach further from the same starting position. At the same time we verified that all the selected systems can successfully fulfill an assistive role, since they all have height and position angle combinations where the manipulator can reach the central and even more extreme trajectories.

The Schunk LWA manipulator was built towards a low weight, controller embedded 7 DOF robotic body with precision standards being left to a secondary objective. It was included in this project to measure the performance and influence

of an additional DOF. Also the fact that such robot is available at our laboratory opens the door to future tests with a real robotic system. The low precision standards of the LWA manipulator (around 1 mm) does not fit the surgery demands, and so it was regarded as a test equipment only.

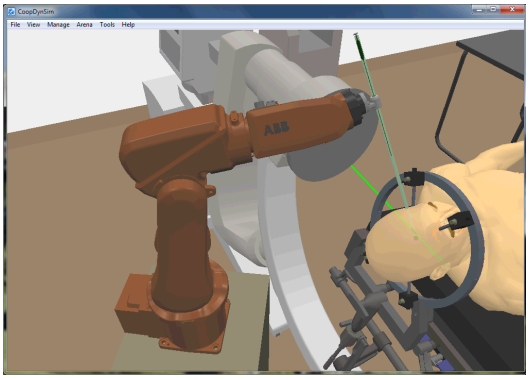
Between the 6 DOF viable robots, the MH5 slightly surpassed the ABB and since we can not evaluate their precision we dare to point MH5 as the most adequate robotic system for our purpose.

6.2 Trajectory execution

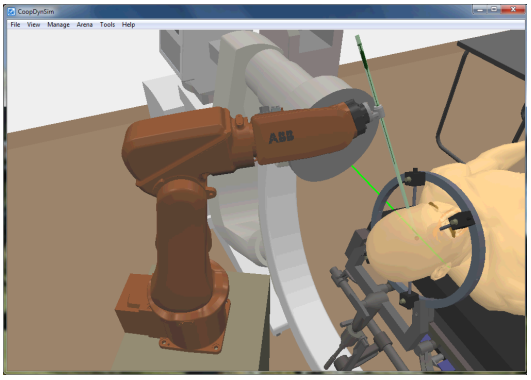
Another topic we wanted to address is how each manipulator was able to move along a defined trajectory. The objective of this section is to depict how each manipulator changes its configuration to move its end-effector always collinear to the defined path passing the entry point towards the target location. For each robot, only one motion sequence will be portrayed since the remaining are conceptually similar with only the manipulator configuration varying. Therefore they do not add any further conclusions and thus will be omitted.

In figure 6.8, we can see the ABB IRB 120 model executing a descending trajectory of 200 mm. In the context of the DBS surgery the robot will rarely perform descending motions with such extent. The arm is required to move to a starting position distanced of X mm along the selected trajectory, and from here the movements along the trajectory should be really small and controlled steps, even until reaching the entry point. From here and as referred in section 1.3, the electrodes should be lowered carefully until reaching 15 millimeters from the target and then proceed by millimeter or half-millimeter steps.

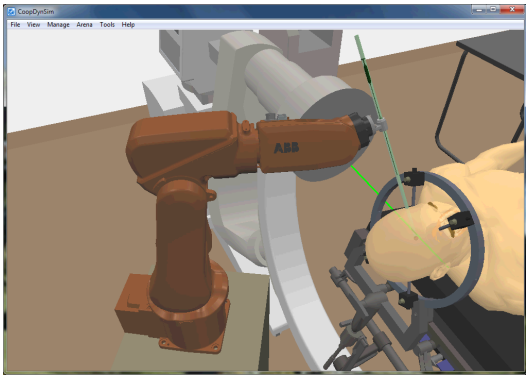
We chose to depict a sufficiently long trajectory to see the manipulator changing its configuration across the several images presented, otherwise the manipulator posture change would be undetectable. Figure 6.9, shows 8 instants of a slightly lower trajectory (200 mm) executed by the Motoman MH5. Figure 6.10 depicts the LightWeightArm II robotic manipulator, executing a 200 mm linear in a sequence of 8 stages.



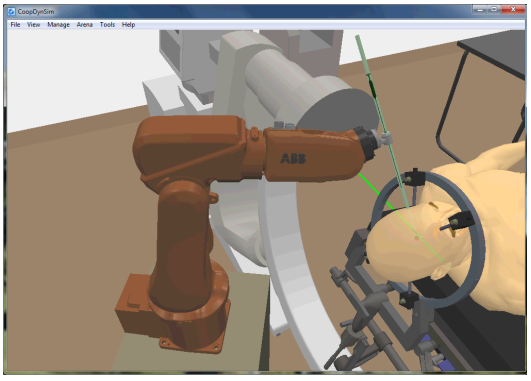
(a) ABB-0



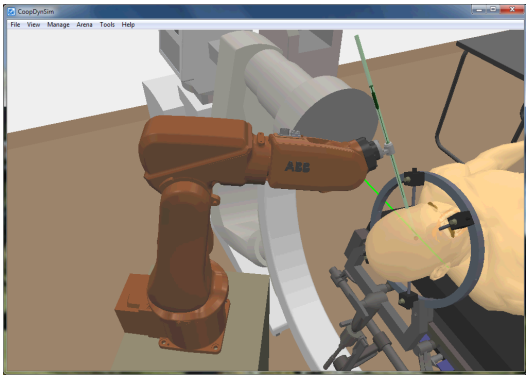
(b) ABB-1



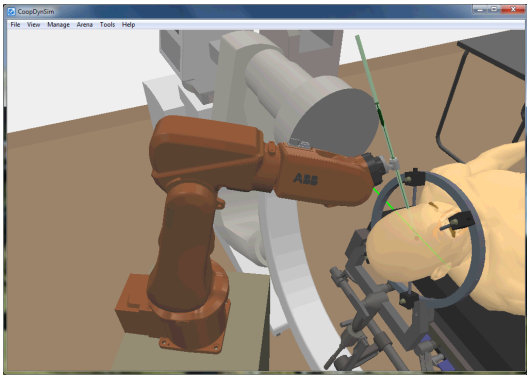
(c) ABB-2



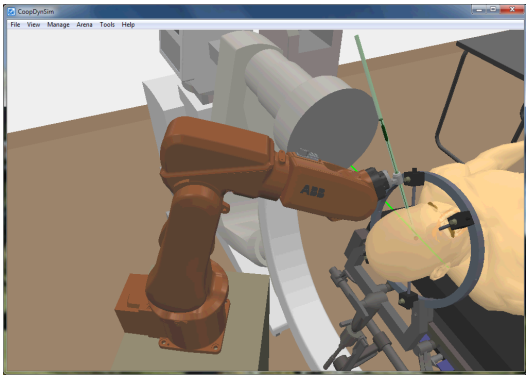
(d) ABB-3



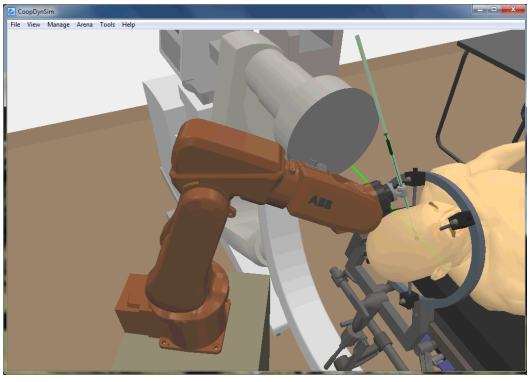
(e) ABB-4



(f) ABB-5

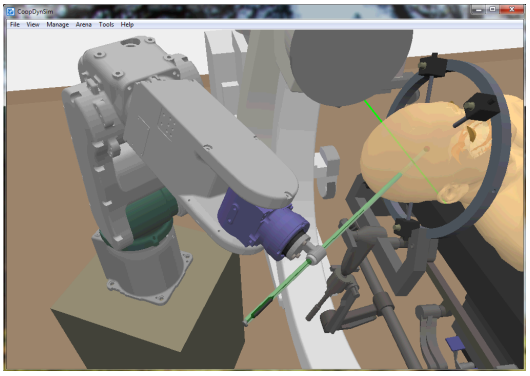


(g) ABB-6

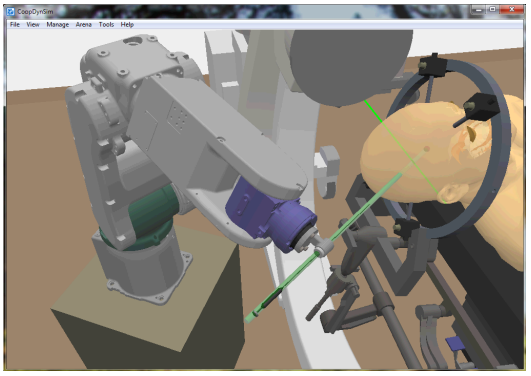


(h) ABB-7

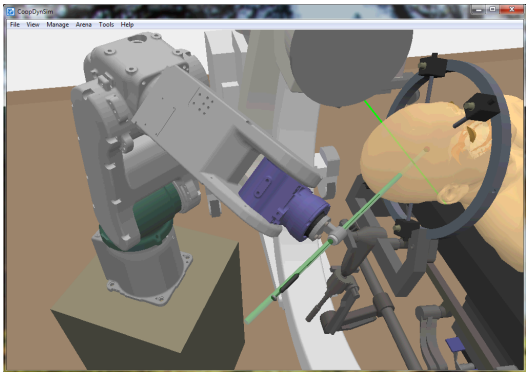
Figure 6.8: ABB IRB 120 executing a trajectory.



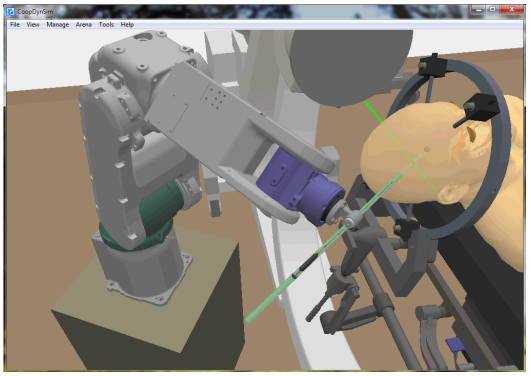
(a) MH5-0



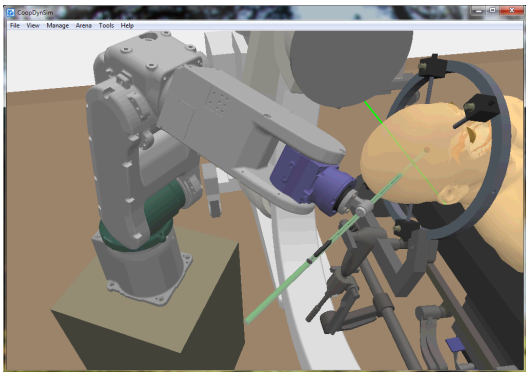
(b) MH5-1



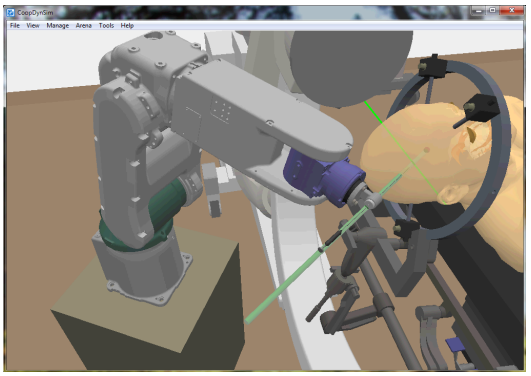
(c) MH5-2



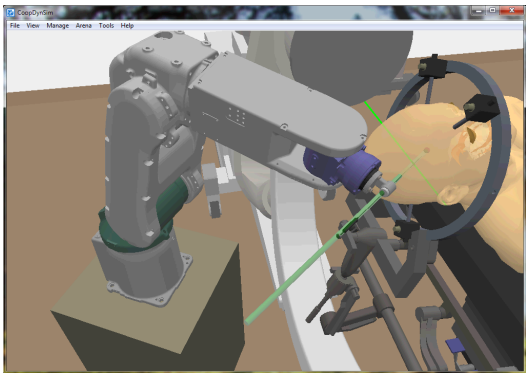
(d) MH5-3



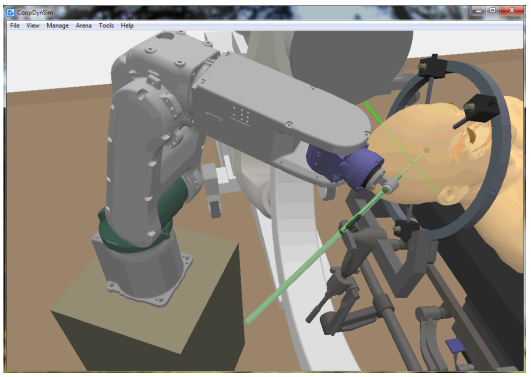
(e) MH5-4



(f) MH5-5

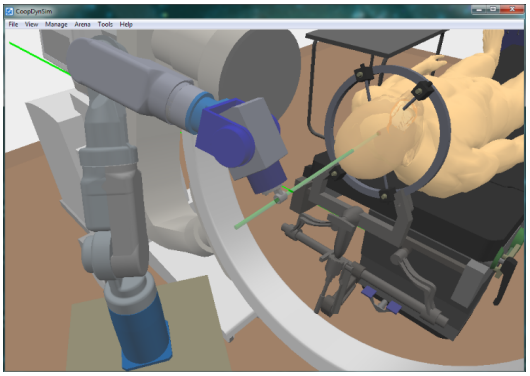


(g) MH5-6

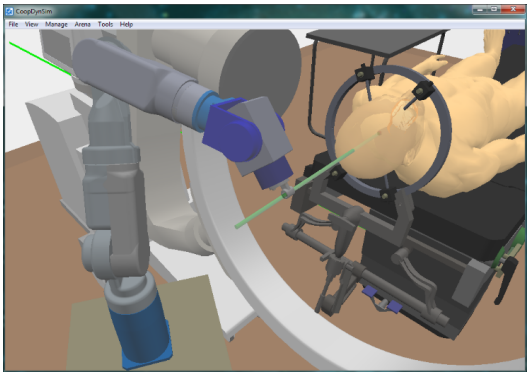


(h) MH5-7

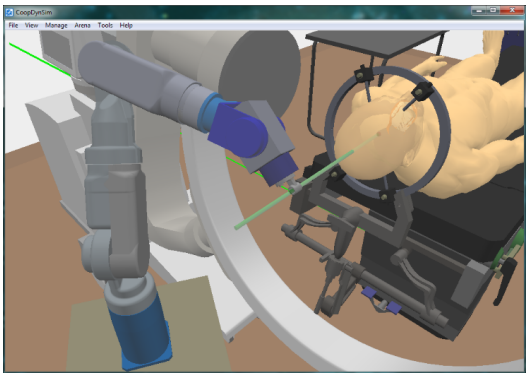
Figure 6.9: Motoman MH5 executing a trajectory.



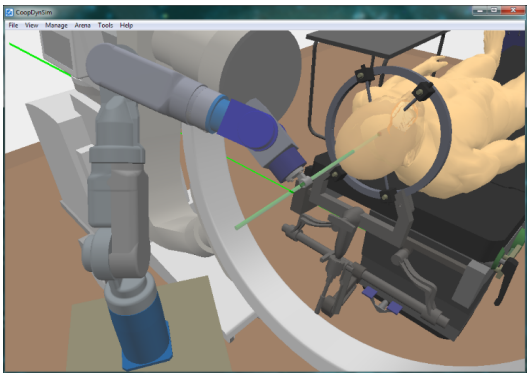
(a) LWA-0



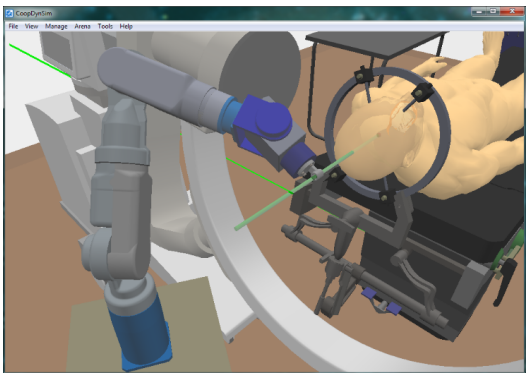
(b) LWA-1



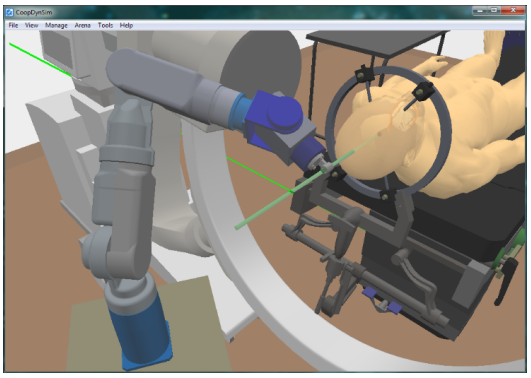
(c) LWA-2



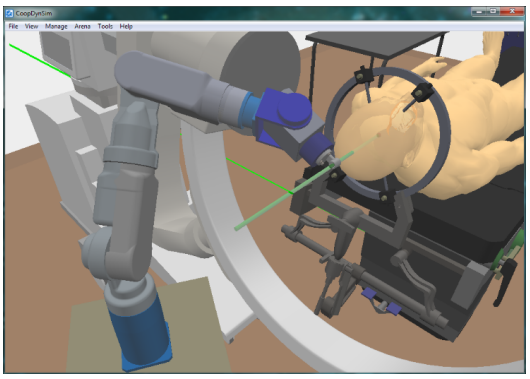
(d) LWA-3



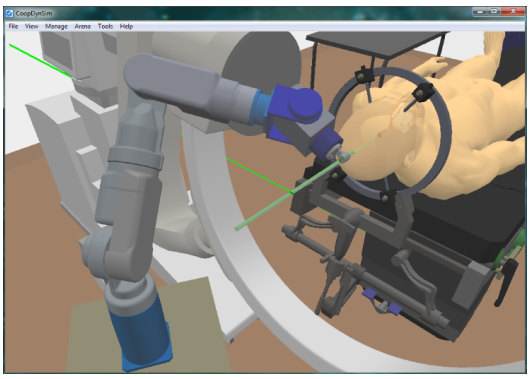
(e) LWA-4



(f) LWA-5



(g) LWA-6



(h) LWA-7

Figure 6.10: LightWeightArm II executing a trajectory.

Chapter 7

First Conclusions

7.1 Summary and Discussion

The aim of this master's thesis was to contribute for the development of a robotic system to assist the medical team in Deep Brain Stimulation (DBS) neurosurgeries.

We started by collecting information and understanding the paradigm of the treatment in the panorama of the treatment sensitive neurological disorders. Several studies (e.g. [10], [13], [15], [11]) and broader essays (e.g. [5], [106]) about these neurological disorders place them among the most debilitating and stigmatizing disorders with higher incidence and prevalence rates. Moreover, there were clear indicators to the increasing numbers of these disorders and to the significant relation between low income population and the disease incidence/prevalence rates.

The growing success and acceptance of the DBS treatment [23], is not supported by the numbers of experienced neurosurgeons and healthcare institutions that perform this treatment [46]. Additionally, DBS neurosurgeries tend to last for several hours during which is required utmost steadiness and precision. The combination of both factors, have lead neurosurgery services that actually perform DBS surgeries to their limits [26]. After attending to a Parkinson's disease DBS surgery we were able to summarily point out some aspects that could be upgraded by using an assistant robotic manipulator (cf. subsection 1.3.3).

Upon acquiring insight about DBS treatment/surgery and the epidemiology of the associated disorders, we did a thorough search on neurosurgical robotic systems that could be used in stereotactic procedures. In table 2.2 in section 2.3, we compiled several systems either oriented towards stereotactic minimal invasive surgery or potentially adaptable. Despite having unique features like MRI-compatible [81], modular multi-body actuators [71], hexapod parallel structures [76] that set them apart from the others, none rounded up the optimal set of characteristics. In conclusion, the desired features for a DBS assistive robotic manipulator include:

1. handling and precisely guiding instrumentation without a stereotactic frame;
2. having a mobile platform easy to attach to the skull clamp;
3. having a vision system able to register the robot's position relative to the surgical reference frame;
4. having a controller able to interface/adapt to the imaging softwares being used in the neurosurgery service;
5. having a low budget acquisition/maintenance costs, be simple and intuitive to use;

With this we went to search the industrial robots market for a system to fulfill our needs. Initially we gathered a set of desired features to narrow our scope, and converged to serial, at least 6 DOF, anthropomorphic, with preferably rigid and precision oriented joint actuators. The search covered the most renowned industrial robot companies and compared the provided data sheet information like manipulator and controller weight, payload capacity, horizontal reach and repeatability. It was briefly discussed the implication of each of this characteristics in an assistive neurorobot. In the end, we selected the *ABB IRB 120*, *Motoman MH5* and *Schunk LightWeightArm* as the most fit robotic systems.

The next step was to develop the kinematic equations for each robotic system. It included the geometric and differential direct and inverse relations taking into account the singular body structures of each robot. It is described the kinematic equations for both 6 DOF and 7 DOF manipulators. In chapter 4, we introduce the singular characteristics of each manipulator needed to generate the kinematic

equations.

Having selected the robotic systems and created the kinematic equations, the next step was to test both with a robotic simulator. After examining several paid and free solutions, facing the required flexibility to add custom components and considering the expected modularity between the robotic system and the control station we decided on an ongoing robotic simulator program know as *CoopDynSim*, currently being developed in our Anthropomorphic and Mobile Robotic Laboratory. The shared knowledge and all the simulator characteristics like the OpenGL graphical representations, the NewtonGD precise physics engine and modular communication also contributed to the final decision.

Although promising, the simulator was not ready to include robotic manipulators, and so part of this dissertation's work had to do with implementing robotic manipulators and the surgical room environment in *CoopDynSim*. We began by designing several intraoperative equipment, adding the physical counterparts and position them creating an environment as close as possible to the operating room in Coimbra University Hospitals neurosurgery service. All the selected serial manipulator systems were created and it was also added several new features like ability to change between several end-effectors and to represent the pre established surgical plan containing the trajectories followed to introduce the electrodes.

The controller application developed in MatLab was created separately from the Simulator and communicates with it using YARP (cf. section 5.3). The controller software provides basic functionalities for testing the developed kinematics and the control algorithms. Moreover, it presents an initial approach towards positioning and manipulating tasks specifically oriented for a DBS surgery assistive role. Special attention was given to safety and thus we implemented mechanisms like: i) movement restriction, to prevent the robot from moving outside a safe area; ii) precision control, to check if the generated kinematic solution precision was acceptable; and iii) actions history log to keep track of all the events taking of the controller application. Also some key thoughts were presented regarding the collision avoidance problem.

Since the robot manufacturers choose to omit some of their product's speci-

fications, mostly relative to their motion execution, the *CoopDynSim* simulator can only partially reproduce their behavior. Thus, it is unrealistic to assess the robot's precision in simulation since the results would most likely differ from the real robots. In spite of this limitation, we still wanted to address the question of where and how to position the robotic manipulator within the operating room to have the least interference with the equipment and yet the optimal reach to fulfill its function. We defined a successful reach the ability to position the tip of the end-effector at least 150 mm from the entry position and moving the instrumentation along the selected trajectory towards the target location. Among the selected manipulators, the 7 DOF *Schunk LightWeightArm* had the larger reaching workspace followed by the *Motoman MH5*. Positioning the manipulator laterally to the patient's head allowed the robot to successfully reach the majority of the tested trajectories without restricting the neurosurgeon workspace and did not cause harmful interactions with the surrounding equipment. We also concluded that to increase the robot's flexibility to reach more trajectories, in its home position, its end-effector should be around 200 mm above the patient's head / entry point.

Despite the initial aim of the robotic system being towards stereotactic neurosurgery, we became recently aware for other potential applications of the same technology. Upon contacting some of the future collaborators they noticed us of their endeavor to study the electrical cortical activity in mice brains, which demanded a precise placement of microelectrodes. Outside the operating room environment, a precise manipulator for instrumentation holding and guidance can still prove to be useful, for instance in research laboratories.

Furthermore, the robotic system in development can be easily adapted to perform other stereotactic functional neurosurgeries. For instance, in StereoElectroEncephaloGraphy the neurosurgeons in mean insert around 12 stimulating electrodes per patient [107]. For each singular electrode the target position needs to be tested in the phantom, and the stereotactic frame placed both in the phantom device and for the patient. In this case the robotic flexibility and swiftness would

be an even more remarkable advantage.

The existence of robotic resources suitable for our purpose, the knowledge about robotic control paradigm along with the overall insight about the DBS proceedings, lead us to believe that the development of a robotic manipulator towards a neurosurgery assistive role is definitely viable. Moreover, if the system can round up the sought safety features, be simple, pragmatic and have low-cost acquisition and maintenance prices it will be very welcome by the medical society. There is already a good feedback on behalf of the contacted neurosurgery services and there is an evident determination to bring this project to fruition, which encourages us to carry on with our work.

7.2 Future Work

Developing a robotic system from practically nothing to be brought and used at an operating room is a challenging task and was realistically an unattainable objective in light of a master's thesis. Notwithstanding this fact, we still embraced the project and during this dissertation we have been continuously building up evidence that point towards the success of our outlined solution.

Foreseeing the potential of the final product and given the novelty and practical sense of this investigation, we sought to continue our work into a PhD program where we aim to develop a final robotic system to assist in neurologic stereotactic procedures. In future work we want to answer some unattended issues like:

- Implement the developed control algorithms into a real robot platform;
- Create a stable mobile platform with an attach mechanism to link the robot to the skull clamp;
- Implement the collision avoidance algorithms;
- Introduce vision system so the robot controller can have a perception of the surrounding environment;
- Develop a reliable method to compute the transformation from the robot reference frame to the surgical reference frame.

Bibliography

[1] T. Haidegger, L. Kovacs, G. Fordos, Z. Benyo, P. Kazanzides, A. Katashev, Y. Dekhtyar, J. Spigulis, and R. Magjarevic, "Future Trends in Robotic Neurosurgery," in *14th Nordic-Baltic Conference on Biomedical Engineering and Medical Physics* (A. Katashev, Y. Dekhtyar, and J. Spigulis, eds.), vol. 20 of *IFMBE Proceedings*, (Riga), pp. 229–233, Springer Berlin Heidelberg, 2008.

[2] W. J. Marks Jr., "Introduction: The expanding role of deep brain stimulation," in *Deep Brain Stimulation Management* (W. J. Marks Jr., ed.), ch. 1, pp. 1–3, Cambridge University Press, 1st ed., 2010.

[3] D. E. Sakas and I. G. Panourias, "Cortical Stimulation versus Deep Brain Stimulation in Neurological and Psychiatric Disorders: Current State and Future Prospects," in *Deep Brain Stimulation: Applications, Complications and Side-effects* (M. H. Rogers and P. B. Anderson, eds.), ch. 3, pp. 49–82, Athens: Nova Science Publishers, first ed., 2009.

[4] J. M. Bronstein, M. Tagliati, R. L. Alterman, A. M. Lozano, J. Volkmann, A. Stefani, F. B. Horak, M. S. Okun, K. D. Foote, P. Krack, R. Pahwa, J. M. Henderson, M. I. Hariz, R. A. Bakay, A. R. Rezai, W. J. Marks, E. Moro, J. L. Vitek, F. M. Weaver, R. E. Gross, and M. R. DeLong, "Deep brain stimulation for Parkinson disease: an expert consensus and review of key issues.," *Archives of Neurology*, vol. 68, no. 2, p. 165, 2011.

[5] World Health Organization, *Neurological Disorders: public health challenges*. Geneva: WHO Press, 2006.

[6] A. Gollhofer, K. M. Newell, and E. G. James, *Routledge Handbook of Biomechanics and Human Movement Science*. London: Taylor & Francis e-Library, 1st ed., 2008.

[7] B. B. Graham, *Using an Accelerometer Sensor to Measure Human Hand Motion*. PhD thesis, Massachusetts Institute of Technology, 2000.

[8] H. B. Stewart, "Surgical placement of deep brain stimulation leads for the treatment of movement disorders: intraoperative aspects," in *Deep Brain Stimulation Management* (W. J. Marks Jr., ed.), ch. 3, pp. 20–31, Cambridge University Press, 1st ed., 2010.

[9] F. Seijo, M. Alvarez-Vega, B. Lozano, F. Fernández-González, E. Santamarta, and A. Saíz, "Common Questions and Answers to Deep Brain Stimulation Surgery," in *Deep Brain Stimulation: Applications, Complications And Side Effects* (M. H. Rogers and P. B. Anderson, eds.), ch. 1, pp. 1–29, New York: Nova Science Publishers, first ed., 2009.

[10] L. Lau and M. Breteler, "Epidemiology: Parkinson disease," *The Lancet Neurology*, vol. 5, pp. 525–35, 2006.

[11] A. Neligan and J. W. Sander, "The incidence and prevalence of epilepsy," in *Epilepsy 2011*, vol. 76, ch. 1, pp. 1–7, London: International League Against Epilepsy's 13th, 2011.

[12] E. D. Louis, "Essential tremor," *The Lancet Neurology*, no. 4, pp. 100–110, 2005.

[13] E. D. Louis, "Environmental epidemiology of essential tremor," *Neuroepidemiology*, no. 31, pp. 139–149, 2008.

[14] J. Benito-Leóon and E. D. Louis, "Essential tremor: emerging views of a common disorder," *Nature Clinical Practice Neurology*, vol. 2, no. 12, pp. 666–678, 2006.

[15] G. Defazio, "The epidemiology of primary dystonia: current evidence and perspectives.," *European journal of neurology: the official journal of the European Federation of Neurological Societies*, vol. 17 Suppl 1, pp. 9–14, July 2010.

[16] R. J. Schwartzman, J. Grothusen, T. R. Kiefer, and P. Rohr, "Neuropathic Central Pain," *Archives of Neurology*, vol. 58, pp. 1547–1550, 2001.

[17] D. Guehl, A. Benazzouz, and B. Bioulac, "Psychosurgery of Obsessive-Compulsive Disorder a New Indication for Deep Brain Stimulation?," *Recent breakthroughs in basal ganglia research*, p. 383, 2006.

[18] M. M. Robertson, "The prevalence and epidemiology of Gilles de la Tourette syndrome. Part 1: the epidemiological and prevalence studies.," *Journal of psychosomatic research*, vol. 65, pp. 461–72, Nov. 2008.

[19] B. J. Coffey, J. Biederman, D. A. Geller, T. J. Spencer, G. S. Kim, C. A. Bellordre, J. A. Frazier, K. Cradock, and M. Magovcevic, "Distinguishing illness severity from tic severity in children and adolescents with Tourette's disorder.," *Journal of the American Academy of Child and Adolescent Psychiatry*, vol. 39, pp. 556–61, May 2000.

[20] E. J. Pappert, C. G. Goetz, E. D. Louis, L. Blasucci, and S. Leurgans, "Objective assessments of longitudinal outcome in Gilles de la Tourette's syndrome," *Neurology*, vol. 61, pp. 936–940, Oct. 2003.

[21] A. L. Benabid, "Deep brain stimulation for Parkinson's disease," *Current Opinion in Neurobiology*, vol. 13, no. 6, pp. 696–706, 2003.

[22] A. L. Benabid, P. Pollak, and D. Hoffmann, "Long-term suppression of tremor by chronic stimulation of the ventral intermediate thalamic nucleus," *The Lancet*, vol. 337, no. 8738, pp. 403–406, 1991.

[23] E. B. Montgomery Jr., *Deep Brain Stimulation Programming: Principles and Practice*. Oxford: Oxford University Press, Inc., 1st ed., 2010.

[24] J. S. Perlmutter and J. W. Mink, "Deep brain stimulation.," *Annual review of neuroscience*, vol. 29, pp. 229–57, Jan. 2006.

[25] A. Machado, A. R. Rezai, B. H. Kopell, R. E. Gross, A. D. Sharan, and A. L. Benabid, "Deep brain stimulation for Parkinson's disease: surgical technique and perioperative management.," *Movement disorders official journal of the Movement Disorder Society*, vol. 21 Suppl 1, no. S14, pp. S247–S258, 2006.

[26] A. E. Lang and H. Widner, "Deep brain stimulation for Parkinson's disease: patient selection and evaluation," *Movement disorders*, vol. 17, no. 3 Suppl, pp. S94–S101, 2002.

[27] J. L. Ostrem, "Patient Selection," in *Deep Brain Stimulation Management* (W. J. Marks Jr., ed.), ch. 2, pp. 4–19, Cambridge University Press, 1st ed., 2010.

[28] W. H. Theodore and R. S. Fisher, "Brain stimulation for epilepsy," *The Lancet Neurology*, vol. 3, pp. 111–118, Feb. 2004.

[29] J. Volkmann and R. Benecke, "Deep brain stimulation for dystonia: patient selection and evaluation," *Movement disorders*, vol. 17, no. 3, pp. S112–S115, 2002.

[30] J. W. Mink, J. Walkup, K. A. Frey, P. Como, D. Cath, M. R. DeLong, G. Erenberg, J. Jankovic, J. Juncos, J. F. Leckman, N. Swerdlow, V. Visser-Vandewalle, and J. L. Vitek, "Patient selection and assessment recommendations for deep brain stimulation in Tourette syndrome.," *Movement disorders : official journal of the Movement Disorder Society*, vol. 21, pp. 1831–8, Nov. 2006.

[31] Huntington's Outreach Project for Education at Stanford., "Basal Ganglia," 2003.

[32] J. Yelnik, "Functional anatomy of the basal ganglia.," *Movement disorders official journal of the Movement Disorder Society*, vol. 32, no. 5, pp. S15–S21, 2002.

[33] J. G. L. Simon, M. A. Caldwell, and R. A. Barker, "Basal ganglia circuitry in normal conditions," 2003.

[34] A. M. Kuncel and W. M. Grill, "Selection of stimulus parameters for deep brain stimulation.," *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, vol. 115, pp. 2431–41, Nov. 2004.

[35] J. O. Dostrovsky and A. M. Lozano, "Mechanisms of deep brain stimulation.," *Movement disorders : official journal of the Movement Disorder Society*, vol. 17 Suppl 3, pp. S63–8, Jan. 2002.

[36] M. E. Anderson, N. Postupna, and M. Ruffo, "Effects of high-frequency stimulation in the internal globus pallidus on the activity of thalamic neurons in the awake monkey.," *Journal of neurophysiology*, vol. 89, pp. 1150–60, Feb. 2003.

[37] E. B. Montgomery Jr. and J. T. Gale, "Mechanisms of action of deep brain stimulation (DBS)," *Neuroscience Biobehavioral Reviews*, vol. 32, no. 3, pp. 388–407, 2008.

[38] J. L. Vitek, "Mechanisms of deep brain stimulation: excitation or inhibition.," *Movement disorders : official journal of the Movement Disorder Society*, vol. 17 Suppl 3, pp. S69–72, Jan. 2002.

[39] E. B. Montgomery Jr., "Principles of Neurostimulation," in *Deep Brain Stimulation Management* (W. J. Marks Jr., ed.), ch. 4, pp. 32–42, Cambridge University Press, 2010.

[40] P. A. Starr, A. J. Martin, J. L. Ostrem, P. Talke, N. Levesque, and P. S. Larson, "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*, vol. 112, no. 3, pp. 479–490, 2010.

[41] M. I. Hariz, "Complications of deep brain stimulation surgery.," *Movement disorders : official journal of the Movement Disorder Society*, vol. 17 Suppl 3, pp. S162–6, Jan. 2002.

[42] T. Foltynie, L. Zrinzo, I. Martinez-Torres, E. Tripoliti, E. Petersen, E. Holl, I. Aviles-Olmos, M. Jahanshahi, M. I. Hariz, and P. Limousin, "MRI-guided STN DBS in Parkinson's disease without microelectrode recording: efficacy and safety.," *Journal of neurology, neurosurgery, and psychiatry*, vol. 82, pp. 358–63, Apr. 2011.

[43] P. B. McBeth, D. F. Louw, P. R. Rizun, and G. R. Sutherland, "Robotics and neurosurgery.," *The Surgical clinics of North America*, vol. 188, no. 6, pp. 1339–1350, 2003.

[44] A. Morita, S. Sora, M. Mitsuishi, and S. Warisawa, "Microsurgical robotic system for the deep surgical field: development of a prototype and feasibility studies in animal and cadaveric models," *Journal Of Neurosurgery*, vol. 103, pp. 320–327, 2005.

[45] R. M. Satava, "Future trends in the design and application of surgical robots.," *Seminars in laparoscopic surgery*, vol. 11, pp. 129–35, June 2004.

[46] T. Wichmann and M. R. DeLong, "Deep brain stimulation for neurologic and neuropsychiatric disorders.," *Neuron*, vol. 52, pp. 197–204, Oct. 2006.

[47] G. Cole, J. Pilitsis, and G. Fischer, "Design of a robotic system for MRI-guided deep brain stimulation electrode placement," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4450–4456, IEEE, May 2009.

[48] S. Pandya, J. W. Motkoski, C. Serrano-Almeida, A. D. Greer, I. Latour, and G. R. Sutherland, "Advancing neurosurgery with image-guided robotics.," *Journal of neurosurgery*, vol. 111, pp. 1141–9, Dec. 2009.

[49] F. Cardinale and R. Mai, "Robotic Implantation of Intracerebral Electrodes in Epilepsy Surgery," *Congress of Neurosurgical Surgeons Spring*, vol. 12, no. 2, pp. 24–26, 2011.

[50] J. Angeles, *Fundamentals of Robotic Mechanical Systems*. Mechanical Engineering Series, Boston, MA: Springer US, 2nd ed., 2007.

[51] K. Cleary and C. Nguyen, "State of the art in surgical robotics: clinical applications and technology challenges.," *Computer aided surgery : official journal of the International Society for Computer Aided Surgery*, vol. 6, pp. 312–28, Jan. 2001.

[52] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*. Advanced Textbooks in Control and Signal Processing, London: Springer London, 2009.

[53] D. B. Camarillo, T. M. Krummel, and J. K. Salisbury, "Robotic technology in surgery: past, present, and future.," *American journal of surgery*, vol. 188, pp. 2S–15S, Oct. 2004.

[54] N. Nathoo, M. C. Cavuşoğlu, M. A. Vogelbaum, and G. H. Barnett, "In touch with robotics: neurosurgery for the future.," *Neurosurgery*, vol. 56, pp. 421–33; discussion 421–33, Mar. 2005.

[55] P. Gomes, "Surgical robotics: Reviewing the past, analysing the present, imagining the future," *Robotics and Computer-Integrated Manufacturing*, vol. 27, pp. 261–266, Apr. 2011.

[56] R. H. Taylor and D. Stoianovici, "Medical robotics in computer-integrated surgery," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 765–781, Oct. 2003.

[57] K. Masamune, L. Ji, M. Suzuki, T. Dohi, H. Iseki, and K. Takakura, "A newly developed stereotactic robot with detachable drive for neurosurgery," in *Medical Image Computing and Computer Assisted Intervention Society* (W. M. Wells, A. Colchester, and S. Delp, eds.), vol. 1496, pp. 215–222, Springer Berlin / Heidelberg, 1998.

[58] B. Fei, W. S. Ng, S. Chauhan, and C. K. Kwoh, "The safety issues of medical robotics," *Reliability Engineering & System Safety*, vol. 73, pp. 183–192, Aug. 2001.

[59] M. A. Talamini, S. Chapman, S. Horgan, and W. S. Melvin, "A prospective analysis of 211 robotic-assisted surgical procedures.," *Surgical endoscopy*, vol. 17, pp. 1521–4, Oct. 2003.

[60] Y. S. Kwoh, J. Hou, E. A. Jonckheere, and S. Hayati, "A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery.," *IEEE Transactions on Biomedical Engineering*, vol. 35, no. 2, pp. 153–160, 1988.

[61] K. Hongo, T. Goto, T. Miyahara, Y. Kakizawa, J. Koyama, and Y. Tanaka, "Telecontrolled micromanipulator system (NeuRobot) for minimally invasive neurosurgery.," *Acta neurochirurgica. Supplement*, vol. 98, pp. 63–6, Jan. 2006.

[62] K. Hongo, T. Goto, Y. Kakizawa, J. Koyama, T. Kawai, K. Kan, Y. Tanaka, and S. Kobayashi, "Micromanipulator system (NeuRobot): clinical application in neurosurgery," *International Congress Series*, vol. 1256, pp. 509–513, June 2003.

[63] D. Handini, "System integration of NeuroBot: a skull-base surgical robotic system," in *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, vol. 1, (Singapore), pp. 43–48, IEEE, 2004.

[64] B. Davies, S. Starkie, S. J. Harris, E. Agterhuis, V. Paul, and L. M. Auer, "Neurobot: a special-purpose robot for neurosurgery," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, pp. 4103–4108, IEEE, 2000.

[65] L. M. Auer, S. Starkie, D. P. Auer, and B. Davies, "Simulating minimally invasive neurosurgical interventions using an active manipulator," *International Congress Series*, vol. 1247, pp. 199–210, Dec. 2002.

[66] Q. H. Li, L. Zamorano, A. Pandya, R. Perez, J. Gong, and F. Diaz, "The application accuracy of the NeuroMate robot–A quantitative comparison with frameless and frame-based surgical localization systems.," *Computer aided surgery : official journal of the International Society for Computer Aided Surgery*, vol. 7, pp. 90–8, Jan. 2002.

[67] T. R. K. Varma and P. Eldridge, "Use of the NeuroMate stereotactic robot in a frameless mode for functional neurosurgery.," *The international journal of medical robotics and computer assisted surgery : MRCAS*, vol. 2, pp. 107–13, June 2006.

[68] M. S. Eljamel, "Validation of the PathFinder neurosurgical robot using a phantom.," *The international journal of medical robotics + computer assisted surgery : MRCAS*, vol. 3, pp. 372–7, Dec. 2007.

[69] P. S. Morgan, T. Carter, S. Davis, A. Sepehri, J. Punt, P. Byrne, A. Moody, and P. Finlay, "The application accuracy of the Pathfinder neurosurgical robot," *International Congress Series*, vol. 1256, pp. 561–567, June 2003.

[70] G. Sivakumar, S. Smith, and P. S. Morgan, "The bionic surgeon: Initial Experience with the PathFinder Robot," 2003.

[71] M. D. Comparetti and E. D. Momi, "Optically tracked multi-robot system for keyhole neurosurgery," *Engineering*, 2011.

[72] E. D. Momi, P. Cerveri, and G. Ferrigno, "Robot and sensors integration for Computer Assisted Surgery and Therapy," in *International Conference on Advanced Robotics, ICAR 2009*, (Munich, Germany), NearLab, Bioengineering Department, Politecnico di Milano, 2009.

[73] D. De Lorenzo, E. De Momi, I. Dyagilev, R. Manganelli, A. Formaglio, D. Prattichizzo, M. Shoham, and G. Ferrigno, "Force feedback in a piezoelectric linear actuator for neurosurgery.," *The international journal of medical robotics + computer assisted surgery : MRCAS*, vol. 7, pp. 268–275, Apr. 2011.

[74] Medtech, "Press-Publication, Available at: http://www.medtechsurgical.com/Press-room/Peer-review," 2012.

[75] Medtech S.A, "Premarket Notification 510(K) Submission Rosa Surgical Device," 2010.

[76] C. Nimsky, J. Rachinger, H. Iro, and R. Fahlbusch, "Adaptation of a hexapod-based robotic system for extended endoscope-assisted transsphenoidal skull base surgery.," *Minimally invasive neurosurgery : MIN*, vol. 47, pp. 41–6, Feb. 2004.

[77] M. Zimmermann, R. Krishnan, A. Raabe, and V. Seifert, "Robot-assisted navigated endoscopic ventriculostomy: implementation of a new technology and first clinical results.," *Acta neurochirurgica*, vol. 146, pp. 697–704, July 2004.

[78] N. Villotte, D. Glauser, P. Flury, and C. W. Burckhardt, "Conception Of Stereotactic Instruments For The Neurosurgical Robot Minerva," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 3, pp. 1089–1090, IEEE, 1992.

[79] C. W. Burckhardt and P. Flury, "Integrated MINERVA system meets demanding robotic requirements," *IEEE Eng Med Biol*, 1995.

[80] G. R. Sutherland, I. Latour, and A. D. Greer, "Integrating an image-guided robot with intraoperative MRI: a review of the design and construction of neuroArm.," *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society*, vol. 27, no. 3, pp. 59–65, 2008.

[81] G. R. Sutherland, P. B. McBeth, and D. F. Louw, "NeuroArm: an MR compatible robot for microsurgery," *International Congress Series*, vol. 1256, pp. 504–508, June 2003.

[82] J. Arata, Y. Tada, H. Kozuka, T. Wada, Y. Saito, N. Ikedo, Y. Hayashi, M. Fujii, Y. Kajita, M. Mizuno, T. Wakabayashi, J. Yoshida, and H. Fujimoto, "Neurosurgical robotic system for brain tumor removal.," *International journal of computer assisted radiology and surgery*, vol. 6, pp. 375–85, May 2011.

[83] "Report of Brain Tumor Registry of Japan," tech. rep., The Committee of Brain Tumor Registry of Japan, Tokyo, 2003.

[84] S. Lavallee, J. Troccaz, L. Gaborit, P. Cinquin, A. L. Benabid, and D. Hoffmann, "Image guided operating robot: a clinical application in stereotactic neurosurgery," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, (Nice), pp. 618–624, IEEE Comput. Soc. Press, 1992.

[85] A. M. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery.," *Current opinion in urology*, vol. 19, pp. 102–7, Jan. 2009.

[86] N. R. B. Martins, W. Erlhagen, and R. A. Freitas, "Non-destructive whole-brain monituring using nanorobots: neural electrical data rate requirements," *International Journal of Machine Consciousness*, vol. 4, pp. 109–140, June 2012.

[87] B. Molaee-Ardekani, J. Márquez-Ruiz, I. Merlet, R. Leal-Campanario, A. Gruart, R. Sánchez-Campusano, G. Birot, G. Ruffini, J. Delgado-García, and F. Wendling, "Effects of transcranial Direct Current Stimulation (tDCS) on cortical activity: A computational modeling study.," *Brain stimulation*, Feb. 2012.

[88] F. Merat, "Introduction to robotics: Mechanics and control," *IEEE Journal on Robotics and Automation*, vol. 3, pp. 166–166, Apr. 1987.

[89] M. W. Spong, *Robot Dynamics and Control Rapid Review of Kinematics*. PhD thesis, University of Illinois, 2007.

[90] R. Pio, "Euler angle transformations [Corrections]," *IEEE Transactions on Automatic Control*, vol. 12, pp. 476–476, Aug. 1967.

[91] E. Costa e Silva, *Reaching, grasping and manipulation in anthropormorphic robot systems.* Phd thesis, University of Minho, 2011.

[92] H. M. Vu, *Control of an anthropormorphic manipulator involved in physical Human-Robot Interaction.* Msc thesis, University of Minho, Guimaraes, 2012.

[93] I. Y. H. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots," in *2009 IEEE International Conference on Robotics and Automation*, pp. 232–237, IEEE, May 2009.

[94] C. Pepper, S. Balakirsky, and C. Scrapper, "Robot simulation physics validation," in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems - PerMIS '07*, (New York, New York, USA), pp. 97–104, ACM Press, 2007.

[95] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *2011 Proceedings of IEEE Southeastcon*, vol. 1, pp. 243–249, IEEE, Mar. 2011.

[96] T. Machado, M. Sousa, S. Monteiro, and E. Bicho, "CoopDynSim: a 3D robotics simulator," in *Robotica*, (Braga), pp. 45–50, University of Minho, 2012.

[97] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," *Southeastcon, 2011 Proceedings of IEEE*, pp. 243–249, 2011.

[98] S. Kucuk and Z. Bingul, "An off-line robot simulation toolbox," *Computer Applications in Engineering Education*, pp. n/a–n/a, 2009.

[99] "OpenGL [Available at: http://www.opengl.org/ (accessed at 28 August 2012)]."

[100] J. Jerez and A. Suero, "Newton Game Dynamics," 2012.

[101] P. Fitzpatrick, L. Natale, and G. Metta, "YARP," 2012.

[102] "3D Content Central [Available at: http://www.3dcontentcentral.com/default.aspx (accessed at 30 August 2012)]."

[103] "GrabCAD [Available at: http://grabcad.com/ (accessed at 30 August 2012)]."

[104] "RevitCity [Available at: http://www.revitcity.com/index.php (accessed at 30 August 2012)]."

[105] T. Wescott, "PID without a PhD," *Embedded Systems Programming*, 2000.

[106] D. Hirtz, D. J. Thurman, K. Gwinn-Hardy, M. Mohamed, A. R. Chaudhuri, and R. Zalutsky, "How common are the "common" neurologic disorders?," *Neurology*, vol. 68, pp. 326–37, Jan. 2007.

[107] M. Cossu, F. Cardinale, L. Castana, A. Citterio, S. Francione, L. Tassi, A. L. Benabid, and G. Lo Russo, "Stereoelectroencephalography in the presurgical evaluation of focal epilepsy: a retrospective analysis of 215 procedures.," *Neurosurgery*, vol. 57, pp. 706–18; discussion 706–18, Oct. 2005.