Contributions to utilize a Cobot as intermittent contact haptic interfaces in virtual reality

Vamsi Krishna Guda

Thèse de doctorat de

L'ÉCOLE CENTRALE DE NANTES

École Doctorale N° 602
Sciences pour l'Ingénieur
Spécialité : *Robotique-Mécanique*

Par
Vamsi Krishna GUDA

Contributions à l'utilisation de cobots comme interfaces haptiques à contact intermittent en réalité virtuelle.

Thèse présentée et soutenue à l'École Centrale de Nantes, le 16 juin 2022
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Gérard POISSONI	Professeur, Université de Orléans, Orléans
Charles PONTONNIER	Maître de Conférences, HDR, Ecole Normale Supérieure de Rennes, Rennes

Composition du Jury :

Président :	Anatole LECUYER	Directeur de Recherche INRIA, Inria Rennes
Examinateur :	Fouad BENNIS	Professeur, École Central de Nantes
	Gérard POISSONI	Professeur, Université de Orléans, Orléans
	Charles PONTONNIER	Maître de Conférences, HDR, Ecole Normale Supérieure de Rennes, Rennes
Dir. de thèse :	Damien CHABLAT	Directeur de Recherche CNRS, LS2N, Nantes
Co-dir. de thèse :	Christine CHEVALLEREAU	Directrice de Recherche CNRS, LS2N, Nantes

Dedicated to my parents, Venugopal Rao GUDA and Subbaravamma KOKA.

Acknowledgments

I would like to thank my thesis directors, Christine Chevallereau and Damien Chablat for offering me this opportunity, giving me the freedom to work on my own, and for being there whenever I needed help in spite of their super-tight schedule. Thanks to there collaborations across the world, I have had the honor to meet and interact with many clever people and learn from them. My co-supervisor, Christine Chevallereau for wonderful insights and ideas, especially for having patience during out long meetings. They have been a great mentor throughout this journey.

Special thanks to Lionel Dominjon (Scientific Manager, CLARTE), for helping me in establishing the virtual environment setup for the experiments. The mechanic of our lab, Stéphane Jolivet for his ultimate patience to print and design the required mechanisms. All the administrative staff on the third floor of our lab, staff of École doctorale and scolarité who are incredibly nice and friendly, and make it a cakewalk to deal with the otherwise annoying administrative stuff.

My parents have always encouraged me to pursue what I like, despite many hardships they had to face. I am eternally grateful to them for their love, education and support for I would not be here without them.

PhD life would not have been so much fun without my friends. Thanks to my friends Jahnavi Marthala , Valentin Le Mesle and Saketh Panuganti for sharing my happy moments and being there through tough times along the journey. I am also grateful to my friends and colleagues Guillaume, Julian, Daravuth, Saman, Keerthi, Kashyap, Sashi, Manohar, Franco, Quynh, Bin Bin, Lauren, Florence, Vivi and Margarita, my office-mates Samuel and Zhen, for interesting discussions and beautiful memories that I cherish.

Thanks to everyone and to all the personnel who kindly volunteered their valuable time to complete all the proposed user tests.

The research work for this doctoral thesis was conducted at Laboratoire des Sciences du Numérique de Nantes (LS2N, Ecole Central de Nantes), and in collaboration with Institut national de recherche en informatique et en automatique (INRIA, Rennes).The funding for this project was made possible by the Agence Nationale de la Recherche (ANR).

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

The following terminologies are repeatedly used in this doctoral thesis:

Definition 1. *A **joint** is a connection between two or more links. It is also known as a **kinematic pair**.*

Definition 2. ***Degree of freedom** (dof) or **mobility** is the number of independent parameters required to define the position of a rigid body in space.*

Definition 3. *A **Workspace** of a six-axis robot is the set of all poses attainable by a particular end-effector mounted on the robot*

Definition 4. *The configurations of a manipulator where it loses or gains one or more* dof *are called singular configurations.*

Definition 5. *A **virtual world** is an imaginary space often manifested through a medium.*

Definition 6. ***Virtual reality** is a medium composed of interactive computer simulations that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a virtual world).*

Definition 7. ***Immersion** is the sensation of being in an environment. It can be a purely mental state or can be accomplished through physical means. Physical immersion is a defining characteristic of VR.*

Definition 8. ***Position tracking** is the computerized sensing of the position (location and/or orientation) of an object in the physical world—usually including one or more parts of the participant's body.*

Definition 9. *In **collaborative environment**, multiple users interacting within a virtual world that enables interaction among participants; not necessarily manifested in virtual reality; a collaborative VR environment can be referred to as multipresence or multiparticipant.*

Definition 10. *An **avatar** is a virtual object used to represent a participant or physical object in a virtual world. The typically visual representation may take any form.*

Definition 11. *A **base** is the link in a mechanism where the first joint(is) is(are) connected and is usually fixed.*

Definition 12. *An **end-effector** is a link in a mechanism where the last joint(s) is(are) connected.*

There are other terms used in this doctoral thesis, which are defined directly in the text for the sake of continuity.

Nomenclature

dof	degree of freedom
M	Transformation matrix
R	Rotation matrix
d	Displacement vector
VR	Virtual Reality
DoF	Degrees of Freedom
ICI	Intermitent Contact Interfaces
SE	Special Euclidean
SO	Special Orthogonal
APF	Artificial Potential Fields
PRM	Probabilistic Roadmaps
ECD	Exact Cell Decomposition
ACD	Approximated Cell Decomposition
RRT/RRTs	Rapidly Exploring Random Trees
BRRT	Bi-directional RRT
RT-RRT	Real-time RRT
BiTRRT	Bi-directional transition-based RRT
ROS	Robot Operating System
API	Application Programming Interface
URDF	Unified Robot Description Format
SRDF	Semantic Robot Description Format
OMPL	Open Motion Planning Library
EST	Expansive Spacial Trees
BiEST	Bi-directional Expansive Spacial Trees
FCL	Flexible Collision Library
ACM	Allowed Collision Matrix
KDL	Kinematics and Dynamics Library
FOV	Field of View
HVS	Human Visual System
VE	Virtual Environment
HMD	Head Mounted Display
VOs	Virtual Objects
CHIs	Classical-Contact Haptic Interfaces
FPS	Frames Per Second
GUI	Graphical User Interface
HCI	Human Computer Interaction
LCD	Liquid Crystal Display

VPL	Visual Programming Lab
$\times$	Cross product
$\cdot$	Scalar product
$\forall$	For all
$\in$	Belongs to
$\subset$	Subset
$\cup$	Union
$\cap$	Intersection

Introduction

Virtual reality technologies allow a user to get immersed in virtual worlds. Haptic technologies born from robotics have increased the immersion in these virtual worlds by providing the sensation of touch.

In a Virtual Reality (VR) simulation, haptic interfaces allow a tangible and physical interaction with the virtual environment, but they must generally be constantly held in hand by the user and therefore do not allow objects to be touched in a natural way. At the same time, many applications require hands-on interaction without intermediaries. This is particularly the case for simulations that require tactile exploration of the physical properties of virtual objects.

Classical force feedback interfaces, also called here classic contact haptic interfaces (CHIs), are robotic systems allowing natural motion interactions with virtual or remote environments. They are used in several domains such as design, manufacturing, assembly, scientific visualization, entertainment, education, medicine, space, rehabilitation, micro manipulation and molecular biology. In all cases, they should provide adequate kinesthetic (force) feedback, contributing to enhance the sense of presence in the virtual environment.

With CHIs, the user is usually mechanically linked to the device's end-effector, typically a handle, whose movements, measured by the robot, are used to know the configuration (position and orientation) of his/her hand. This information is necessary to provide force feedback which is consistent with the virtual scene and the mechanical properties of the virtual object (VO) being touched. The mechanical link that is established when the user manipulates the haptic device has however a non-negligible influence since he/she experiences the friction, inertia and vibrations of the mechanical structure, even in free space where he/she is expected to feel nothing. Such unwanted sensations decrease the realism of the interaction since the user feels all the time the presence of the robot. In addition, the difference between free space and contact is less distinctively felt than in the real world.

In order to cope with these issues, several efforts can be made in terms of mechanical design, e.g. use of very lightweight and very stiff structures (even if an optimal trade off is difficult to attain) and more efficient transmission systems. Another approach consists in installing a force sensor at the level of the robot's end-effector in order to measure and compensate any resisting force in the direction of displacement. However, resisting forces can never be totally canceled and none of these approaches completely eliminates the feeling of the presence of the robot in free space.

Years of research on haptics, robotics, and interaction in VR have led to the development of a new generation of haptic devices called intermittent contact interfaces (ICI) [73],[87],[7]. Intermittent-contact interfaces (ICIs) represent an original and promising approach aiming to cope with the aforementioned issues. Its principle consists in removing the mechanical link between the human operator and the force feedback interface during manipulations in free space and come at his/her contact only when force feedback is required. This solution implies the need

to track and closely follow the user's movements in free space and to prevent him/her to move in the constraint direction when a VO is being touched. This way, the user doesn't feel any force in free space (perfect transparency) and the transitions from free space to contact are deemed to be felt more naturally as the robot really touches the user at the simulated contact moment. This approach aims to improve the realism of the interactions, however it suffers from several shortcomings. First, its efficiency has not yet been proven in terms of user safety. Second, even if IC interfaces are experimentally proven to be stable at low speeds, they tend to become oscillating at higher speeds. Finally, despite the fact that a lot of tasks are performed by the mean of tools in the real world, most of existing ICIs focus on bare finger interactions and are therefore not optimal for simulating tool-mediated tasks.

User safety is central to the implementation of an ICI. To satisfy this constraint, we propose to use a cobot, for which we will develop specific path planning algorithms taking into account its low performance in terms of travel speed in the intrinsically safe mode, as well as all segments of the robot and the user's entire body in interference management. These algorithms will be based on a user activity prediction model that will allow both specifying a final desired location and the constraints to be respected when defining the path to reach this desired location.

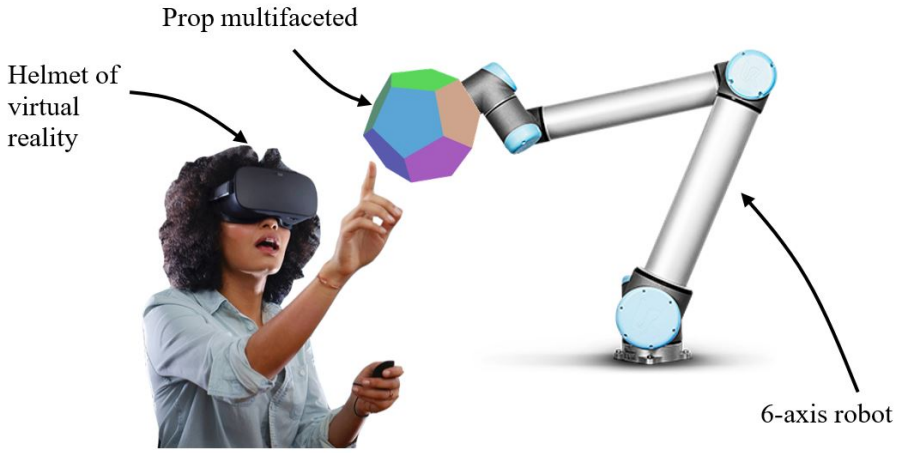


Figure 1 – Conceptual scheme of the experimental platform

This thesis, entitled "Contributions to utilize a cobot as intermittent contact haptic interfaces in virtual reality" presents a series of research works aiming to leverage the usability of such a system. This research work is created to address the current limitations in such a haptic device system.

Description of context

This thesis was developed under the frame of the French National Research Agency (ANR) LobbyBot Project. The LobbyBot project consisted in developing a system that could be integrated into an industrial application for automotive interior prototyping. The system is intended to be used to recreate an automotive cockpit for faster prototyping in VR for the Renault Group, a French automotive company. In this fast prototyping process, designers had to explore and evaluate the perceived quality of a virtual car interior with their sense of touch of different materials, shapes, and objects that could be arranged in a virtual automotive interior that could be easily configured in VR. This new paradigm was conceived as a means to save the Renault Group

costs in budget and time for fabricating actual automotive cockpits that are exclusively used for prototyping purposes. The object of the thesis will be the control of the robot to implement this ICI.

Figure 2 – The LobbyBot project setup.

The project assembled a consortium consisting of four partners: The Renault Group which was in charge of providing the use-case scenario and problematic. The Laboratory of Digital Sciences of Nantes (LS2N) was in charge of the path-planning algorithm for avoiding collisions with users. The National Institute for Research in Computer Science and Automation (INRIA) was in charge of conceiving new 3D interaction techniques to compensate for the inherent limitations of ICI. CLARTE was in charge of integrating the aforementioned contributions from all the other partners to implement the system.

Interaction techniques will allow managing any delays by the ICI in relation to the user. To increase the extent of sense of the form and materials can be restored by the use of an adapted pseudo-haptic return, and to follow a surface with the finger through appropriate sensory feedback. The integration of all the results into an ICI prototype will make it possible to validate the interest of the solution on an industrial case study. With current technologies, this industrial application (evaluation of the perceived quality of a virtual car interior) cannot be treated in a fast and low cost way.

Applications of ICIs

ICI applications range from industry (Figure 3), entertainment (Figure 4), medicine (Figure 5), and research purposes (Figure 6). In all these cases, users expect to “encounter” a surface to touch or manipulate in a Virtual Environment. In the case of industrial applications, these devices are considered for virtual prototyping that requires to have haptic feedback in several locations to recreate workspace or expected location for objects to be manipulated. In the case of entertainment, they are used to recreate elements that can come in contact with the users when interacting with a virtual environment [61]. In the case of medicine, its often used for remote body-palpation and surgery practice [42]. The use in research purposes often looks for leveraging the device’s capabilities for rendering more complex surfaces and objects [74].

Figure 3 – Application in Industry by Lobbybot [104].

Figure 4 – Haptic-go-round from one direction to another while fishing in the virtual scene. [61].

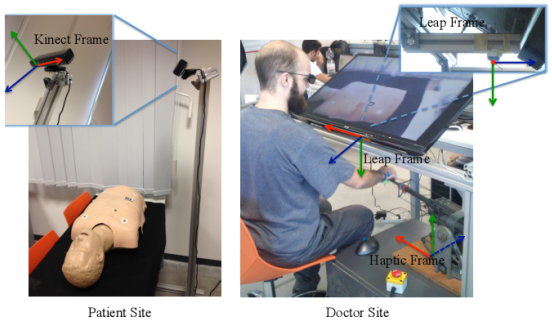


Figure 5 – The virtual palpation system. [42].

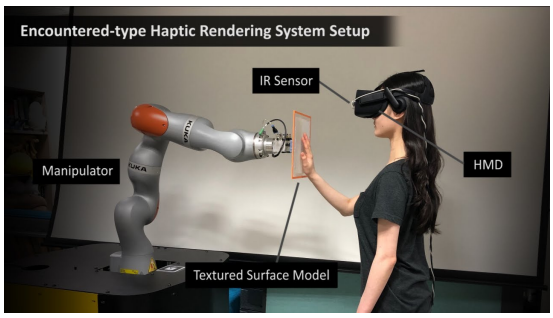


Figure 6 – Synthesizing the Roughness of Textured Surfaces for an Encountered-type Haptic Display [74].

Challenges

In order to leverage the capabilities of ICI, several challenges must be addressed, these are addressed in different dimensions such as: **User safety and security** and **User immersion experience**. The challenges are divided into research questions (RQ) that this thesis addresses in the following chapters, namely; (RQ1) Robot placement; (RQ2) User model; (RQ3) Trajectory planning and (RQ4) User prediction techniques. These questions are further presented below.

Challenge 1. User safety and security

The concern to avoid unexpected collisions with users has been present ever since the early days of Human-Robot Interaction research. Measures for addressing this issue normally consider path-planning algorithms that help the robot to actively avoid the user in cases where both user and device could come into contact involuntarily. However, the use of path-planning often causes the device to move more slowly. This delay affects response time and user's perceived immersion in the virtual environment, and thus, it has been recognized as an issue to be addressed by the research community.

As an alternative to solutions exclusively relying on path-planning for avoiding collisions with users, we propose additional techniques like the placement of robot to minimize the workspace interaction, and creation of multiple zones for robot motion. Research concerning user safety for ICI systems needs to look for strategies that help to avoid any undesired contact that could break users perceived immersion in the virtual environment. The main objective remains to ensure the safety of the operator. With virtual reality, the operator has a modified perception of the robot and of the risk of collision. A visualization in the virtual world of the robot reduces the immersion but increases the safety. In the intersection zones between the space possibly scanned by the robot and the human, the robot (UR-5) will be placed in cobot mode, i.e. it will stop and switch to gravity compensation mode if a contact effort is detected. Thus it cannot injure the operator. However, the speed of the robot is then limited.

Research Question 1. Robot placement

Robot placement holds the promise of removing the need for a highly dedicated and structured workspace, as well as responding more quickly to environmental changes. Within the systems, dynamic and robust placements are crucial and strategically important, since they are often done in early stages in the process.

A complete set of operations consists in performing a specific task/operation by a robot on a set of task points. Often, after the robot returns to its starting configuration, a user is introduced to the system, and operations are performed. Since these cycles are repeated several times, it is very important that they are executed as fast as possible in order to increase immersion.

Once a set of specific points is assigned to a robot, the layout has limited freedom to optimize the robot workstation:

— robot's base placement (translation and rotation); The orientation of robot TCP at each task points is important as it effects the immersion of the user. So the robot should be able to reach the task points with the required orientations. The study started with no information on the robot base location in the virtual environment.

— visiting order of the work-points; In this approach the user decides the order of interaction with the task points. To improve user safety , the robot has to be closer to the user hands near the task points and as far as possible to the human body (torso, head and especially

neck). Major concerns were to restrict the operation of the robot to one aspect to avoid crossing singularities while performing the tasks.
— robot's home configuration in the station (six joints); initial approach is to achieve one single base location and have single aspect of robot to connect and move between all the task point.

The last three ones may be modified by changing the robot program, whereas the first has to be completely decided before installing the robot in the workspace.

Research Question 2. User model

There is no fixed technique or standard procedure to use a tracking system to explore user actions in VR. Human movement tracking systems are expected to generate real-time data that dynamically represent the pose changes of a human body (or a part of it), based on well-developed motion sensor technologies. Generic tracking systems employed within these systems adhere to the human body in order to collect movement information. These sensors are commonly categorized as mechanical, inertial, acoustic, radio, microwave, and magnetic-based. In such systems, the user is generally equipped with sensors on his hands, that give the sense of interacting with a virtual object. Additionally, these sensors don't give information about the position and orientation of the user's location in the system. Visual marker-based tracking is a technique where cameras are applied to track human movements, with identifiers placed upon the human body.

The whole idea of ICI is to increase the immersion and remove the sensation of robot presence in free space. Since the robot has a plan to move between task points without colliding with the user. For safety concerns, the position of the user is very important. However, we cannot have complex tracking systems to locate the user. Normally the use of external cameras and markers are used to locate the user's position. But such systems increase the number of sensors attached to the user, which reduces the immersion. So the goal is to use less additional sensors as possible to give the best immersion experiences while getting the accurate location of the user for planning trajectories of the robot.

Challenge 2. User immersion experience

Haptic feedback for ICIs goal is to improve the immersion experience of users. The major idea is to eliminate the mechanical link so as to have perfect transparency in free space. But even trying to achieve this experience comes with the challenges like planning the robot motion between the VO being touched (without colliding with the user), and predicting which VO object the user intends to touch in the environment.

Research Question 3. Robot motion planning

In the given scenario the user is immersed in Virtual reality. The user has no information regarding the motion of the robot. The robot's motion has to be safe for the user, experiencing the VR environment. It should avoid collision with the user and also any other obstacles in the environment. The robot motion should take into consideration collision with the user based on the tracking system used. It's not only for the user's safety, but also to have a better immersion of the user, the robot must place a real object at the place where the user want to touch the VO as quickly as possible, and before the user hand reach the contact location.

Research Question 4. Target object detection.

Since they touch the user only when force feedback is required, intermittent contact interfaces, and in particular close-tracking-type devices, aim to provide more realistic interactions with virtual environments than classical contact haptic interfaces (CHI). User intention prediction in ICI remains a challenge to be properly addressed by the research community.

Previous systems force the user interact with the VO, selected by it. These systems have to use control algorithms and interaction techniques to make the haptic rendering as efficient as possible. For the new system we want to give the user freedom to select the VO to interact and the robot system should adapt to align itself w.r.t to the VO. The goal of such a system is to predict the VO the user wants to interact with as soon as possible. However, for these new systems the two solutions (control algorithms and interaction techniques) are not enough to render the haptic sensation.

Thesis road map

The main research axis considered as contributions of this thesis is improving user safety and immersive experience for users of an ICI haptic device in human-robot interaction.

The user safety axis is improved by addressing three contributions, the first contribution (C1) consisting of a set of safety techniques based on robot placement, the second (C2) creation of a user model for tracking user information, and the third (C3) techniques for robot trajectory planning.

Then, the second axis immersive experience is improved by making sure the user does not lose the illusion of the environment. This aspect can be addressed by making sure the user does not wait a long time for the robot to reach the desired interactive location, this is achieved by addressing 2 contributions, first contribution (C3) techniques for robot trajectory planning and second contribution (C4) techniques for user goal predictions, designed to optimize the response time in ICI systems.

The relationship between the contributions and the research axis can be seen in Table 1.

Contribution	User safety	Immersive experience
(C1) Safety techniques based on robot placement.	x	
(C2) User model for tracking user information.	x	
(C3) Techniques for robot trajectory planning.	x	x
(C4) Techniques for user goal predictions.		x

Table 1 – Contributions to the research axis

In the following chapters the details about how the contributions addressed the research questions will be presented.

Contributions

The contributions of the thesis are depicted in chapters which are described hereby:

Chapter 1 presents a literature review concerning present ICI haptic devices. In the first part, the history of these devices is narrated. The second part presents an analysis of the hardware used for these devices. The third section presents the types of haptic perception used

in literature. The fourth part discusses application scenarios. The literature review concludes with a discussion of the presented research works.

Chapter 2 presents a set of safety techniques for users based on the placement of the robot (C1). This chapter introduces a design space for safety techniques using visuals restricting the interactive workspace of the robot and the user, in order to reduce potential unintended collisions. The dimensions of the space focus on where the user wants to interact with the virtual environment. Explanation on how it protects the user. Using this design of robot placement, a set of workspaces was developed to explore variations of the prop orientations. An evaluation focusing on the best solution for robot placement is done. Safety was evaluated and the ideal location of the robot was defined.

Chapter 3 presents the design and evaluation of the user model (C2). The model was designed to estimate the user location in the ROS environment. The chapter introduces first the need for such a model and different ways of achieving the goal. The model created had to have features such as movement of both arms and also the mobility of the user around the hip. These major three moving parts provided better information about the user's location, improving his safety in using an ICI system. A user study was designed to assess test the performance of these motions to test the accuracy and reliability of creating user model. Results suggested that the model is accurate for the given scenario in locating the user in the environment.

Chapter 4 presents an approach for the trajectory planning of robots. The goal is to increase immersive experience (C3) by reducing the time taken by the robot to reach the contact area. This approach renders large surfaces and multiple textures through the use of a rotating prop, that couples the prop's rotation and position with the users' hand position when exploring a textured surface in VR. A use-case scenario was designed for contextualising this approach. Later, a user study was conducted to validate the approach haptic rendering performance.

Chapter 5 presents a novel approach for the prediction of user intention based on head gaze and hand position. Different strategies are presented to predict where the user wants to interact with the virtual environment (C4). The chapter starts by introducing the different approaches and by describing their main features. Each approach presented is evaluated on four key features: (1) time taken to prediction, (2) time taken by the robot to reach the contact area, (3) distance traveled by the robot, and (4) safety of the user. These approaches integrate an interaction technique for contact area selection and release. An evaluation concerning the speed of the system is presented. Finally, the results are presented and discussed.

Chapter 6 concludes the thesis manuscript summarizing the thesis contributions as well as providing perspectives for future work for ICI.

Collaborations

A rehabilitation project entitled "Development, engineering, prototyping of a low-cost robotic system for postural rehabilitation with intuitive interface using consumer VR-AR technology and wearable IoT sensorization of the patient". Collaborators: Stanley Mugisha, Matteo Zoppi and Rezia Molfino, University of Genoa, Italy.

Chapter 1

State of the art and theoretical background

1.1 Introduction

This chapter forms the basis of this doctoral thesis. It puts forth the state of the art and explains the fundamental concepts used to analyze and understand the behavior of the studied herein.

First, a brief literature review is presented. Then, the human senses are introduced, starting from vision to touch. Furthermore, the usefulness of virtual reality, and how we achieve immersion using the senses in virtual reality is described. While this offers significant immersion, there are some problems to be addressed. To overcome these problem, a new system called ICI is introduced. Different ways to implement such a technique are then presented.

Finally, a system is defined for the given context of study and how to achieve such a dynamic system is explained in short.

There is an increasing demand for simulators aiming to artificially recreate real world environments. These systems are used for various applications such as assembly verification, e.g. virtually checking that a new system can be easily assembled before launching the production, training, e.g. fight simulators, or simply entertainment, for instance immersive video-games.

An ideal simulation environment should stimulate each one of our senses (sight, hearing, smell and touch) to give us the illusion of being present in the recreated environment (immersion) and allow us to interact with it. However, recreating the corresponding sensations requires complex devices while all of them are not always required.

Indeed, it can be observed in practice that visual and touch feedback are sufficient to cover a wide scope of applications, even when fine interactions are required, e.g. dexterous manipulation of virtual objects (VOs). Visual rendering is obtained with systems ranging from simple monoscopic screens to more complex configurations like head mounted displays or multi-screen immersive systems such as CAVEs, with techniques allowing to recreate photo-realistic images. The study of the sense of touch is unfortunately less advanced. It is addressed with haptic interfaces, i.e. complex mechatronic devices able to provide either tactile or force feedback.

Force feedback interfaces are widely employed in virtual reality (VR) environments, e.g. for the training of novice surgeons in medicine, for the simulation of a product assembly, for education purposes or for scientific research. However, despite the wide range of applications of these devices, they still require to be improved in order to provide force sensations that are totally realistic.

Figure 1.1 – First HDM by NASA. [91]

Recently, human robot collaboration become popular and interesting in the field of engineering. For robotics, environments that require both humans and robot to work simultaneously to improve the performance in industries.

1.2 What is virtual reality ?

Even if virtual reality sounds as a very modern concept, it has been around for more than half a century.

VR accounts for an interactive computer simulation allowing the user to visualize a virtual environment (VE) which recreates a scene of the real world. Interactive means that such simulation acquires, by means of a tracking system, information on the user's state and is able to provide sensory feedback to one or more senses. The main objective is to make the user feel as being immersed in the VE [93].

Four basic characteristic elements of VR can be identified: the virtual world (or VE), immersion, sensory feedback (as a response to user's actions) and interactivity.

- A computer-based VE contains the description of the simulated objects as well as the rules governing their interactions, e.g. the effect of gravity making virtual objects to fall as it does with those in the real life.
- In the present context, immersion implies the notion of being present in the VE. In simple words, it is the belief of having left the real world to be present inside the VE. Immersion can be achieved if the VR system is able to provide realistic stimuli to all human senses [90].
- Sensory feedback requires proper displays, e.g. a head-mounted display (HMD, see Figure 1.1 and Figure 1.2) able to visually immerse the user in the virtual world, and a force feedback interface, hold by the user, that will apply on him/her the forces generated when VOs are being touched.
- Finally, interactivity refers mainly to the fact that the VE is responsive to the user's actions (inputs). It requires a tracking system able to determine the movements of the user to ensure consistency between the real world and the VE. As an example, the visualized scene will adapt so that the user has the impression to move inside the VE.

The first practical implementation of the VR paradigm was developed in 1957 and patented in 1962 by Morton Heilig, who is considered as the father of VR. In such a system, known as Sensorama (see Figure 1.3), a single person could perceive, e.g. a motorcycle ride through a city via sight, smell, vibration and wind [120].

Figure 1.2 – Classic HMD and VR glove environment. [54]

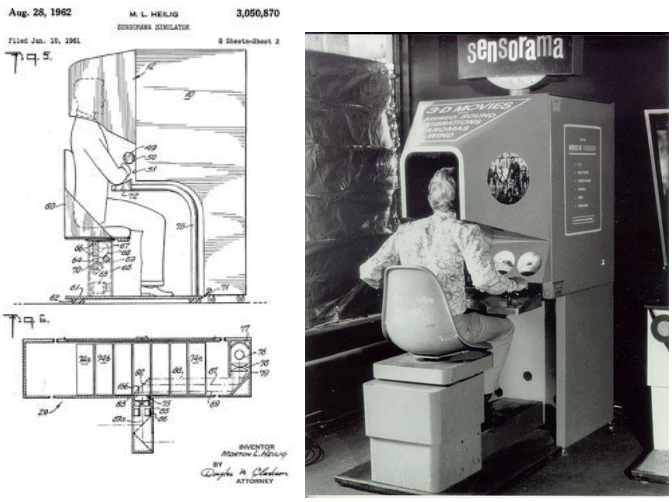


Figure 1.3 – First developed and patented VR system. [120]

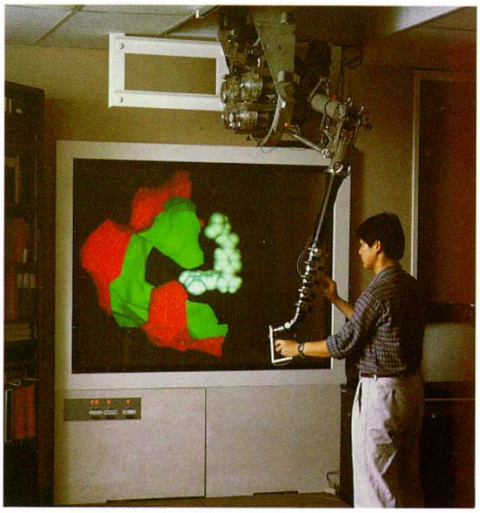


Figure 1.4 – GROPE-III haptic display system. [17]

Figure 1.5 – Example of a CAVE. [54]

Several other VR systems were developed. The GROPE I to III systems, created in the late 60s and early 70s, allowed the user to move and feel interaction forces between molecules thanks to a haptic interface (see Figure 1.4). Other examples of VR applications using force feedback will be described in future sections. As stated in the above paragraphs, one of the main goals of a VR system is to create the sensation of "presence". A step forward in this direction was achieved during the 90s with the concept of CAVE (Cave Automatic Virtual Environment). It consists of a room whose walls are in fact screens displaying a VE (see Figure 1.5).

This technology is however very expensive and requires an important volume to be installed, e.g. four equally sized walls of $1.8m \times 1.8m$. As a consequence, the current tendency is to use HMDs instead. This technology have made huge progress in the recent years and is now both efficient and affordable (see Figure 1.6).

Another important concept in VR applications is the notion of avatar, which is the virtual representation, for instance, of a human being as well as objects of interest regarding the task or the scene (see Figure 1.7). Finally, Figure 1.8 shows how the aforementioned elements are linked.

The following section will provide an insight on the importance of the sense of touch and, in particular, the role it plays when non-direct manipulation of virtual objects and interaction with VEs takes place.

(a) HTC VIVE (b) Oculus Rift

Figure 1.6 – Commercially available HMD. [54]

Figure 1.7 – Principle of avatar.

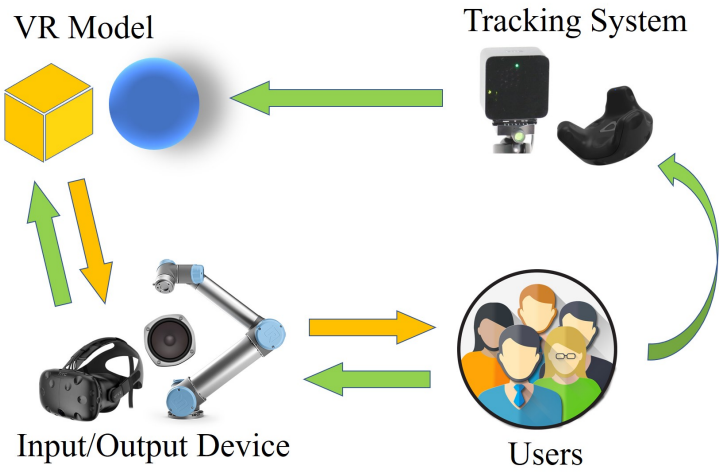


Figure 1.8 – Virtual reality feedback loop.

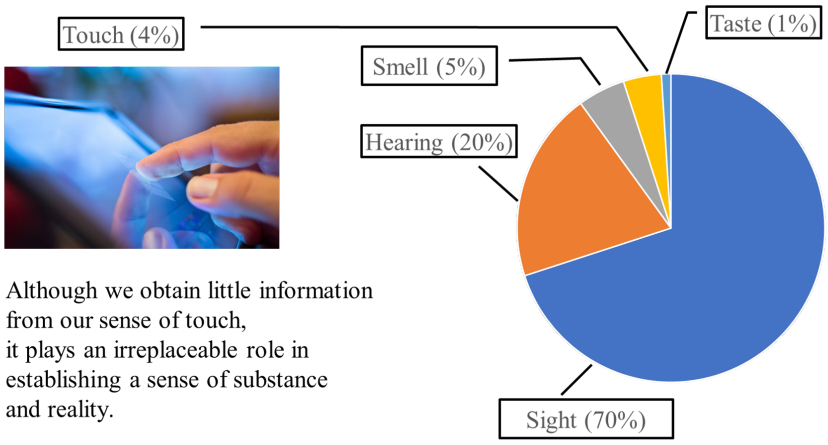


Figure 1.9 – Ratio of information provided by each of 5 human senses.

1.3 Human senses

Our senses are physiological tools to perceive information about the physical world. As defined and classified by Aristotle, human beings have at least five senses. These are sight, hearing, touch, smell and taste. Sensory processing is the capability to acquire, elaborate and incorporate information through our senses. There is a minimal point below which any stimulus does not produce any effect on sensory organs. Above this level, a stimulus' minimum perceptible variation is proportional to the absolute value of intensity (Weber's law). A specific sensory organ such as eyes, ears, skin, nose and tongue receives a stimulus which is above the minimum perceptible level. These sensory organs are the starting point for information transmission through the body through the nerve tracks. The afferent message from the sensory receiver is transferred to the nerve centres that are spinal cord, brain and cerebellum [100]. Once information is integrated and processed in the brain, nerve centers pass the nerve messages to the related sensory organs (skeletal muscles, eye muscles, vocal cord muscles, etc.).

Morton Heilig, in 1952, studied the senses in terms of their capacity to activate human attention [57]. It is essential to know the contribution of each sense while perceiving the world. Figure 1.9 clearly shows that the sight provides most information and captures the majority of our attention. The stimulation of the visual system, therefore, plays a significant role in fooling our the senses. Hearing, which is also often taken into account, is the second most crucial sense. Touch does not in general play an important role comparing to sight and hearing, even when it is essential for specific manipulation tasks. Due to the marginal role and difficulties of implementation, smell and taste are currently under development in most VR systems. However, immersive VR system should produce different signals and stimuli in several ways to replicate the physical world and use various display devices to immerse the user.

1.3.1 Sight

Human use sense of sight to assess the position of the objects with respect to them, so they can approach the objects, grasp them or avoid them for survival when needed.

Without eyesight, dangerous situations can be quite challenging to identify. Sight can therefore be used to avoid dangerous situations and materials.

Since sight is a sensory channel that provides most information and captures the majority of our attention, it is essential to achieve visual immersion that enables users to see the virtual world in a realistic way. For the Immersive VR experience, ideally, visual displays are able to produce feedback equal to or above the limitations of the human visual system [57].

Visual perception is the ability to see and interpret an environment using light in the visible spectrum, which is reflected by objects in the surrounding environment. In human visual perception, the two eyes act as a sensor, the neurons function as a connecting cable and the brain acts as a processor. The eyes, as sensory receptors, are engaged in the observation of the environment.

A wide Field of View (FOV) in virtual environments can enhance the sense of immersion and performance. The term FOV is one of the essential features of the Human Visual System (HVS). FOV is an angular measure that represents an open area visible to the human eye. For a healthy eye, two FOV information together describe human vision. The visible area through the eyes is generally indicated as an angle for the horizontal and vertical component of the FOV. It is approximately 160 degrees horizontally and 130 degrees vertically when one eye is used and rises horizontally to 200 degrees when looking straight ahead with two eyes - thus we can see behind us by 10 degrees. When we rotate our eyes to one side or another, we are able to see an additional 50 degrees on each side [64]. Therefore, to cover the entire range of natural human FOV, HDM needs approximately 300 degrees FOV horizontally. Moreover, the human brain uses the horizontal change of picture location reported by both eyes to determine the depth or the distance to the virtual object displayed on the scene from the viewer. This means that FOV is also crucial for depth perception.

1.3.2 Hearing

The sound gives us precise details about our surroundings and can often help us to assess the distance and direction of objects very accurately [138]. Sense of hearing is very useful for humans and important for survival in many circumstances. We can detect a sound in the dark when we cannot see. The auditory perception is omnidirectional, as opposed to limitations on the visual field of view, a human can detect sound from any place in 3D. In view of this omnidirectional aspect, hearing directs our visual senses, or "the function of the ears to point the eyes" as Cohen and Wenzel said [26].

In Multimedia, hearing assumes at least equal importance to the user's feeling of immersion [112]. A 3D (Spatial) audio-system enables a user to perceive the position of sound sources emanating of an arbitrary position in 3D space from speakers or pair of headphones. Spatial sound processing goes far beyond conventional stereo sound technologies by allowing virtual sound sources to possess attributes like left-right, up-down and back-forth.

1.3.3 Smell

Sense of smell, Olfaction, is essential to everyday life. It is a part of the chemosensory system that enables us to distinguish between a wide range of smells and tastes [24]. Olfaction plays a vital role in making us understand the environment in real wold. It enables us to recognize food, drinks and also provides a sense of pleasure and warnings of danger. For instance, it advises us to leave a building with fire or if the food is rotten and poisonous. Furthermore, smell sense is more connected with feelings and emotions than thoughts and decisions [24]. Olfactory information can enhance the user's sense of immersion in the EV by tricking their sense of smell. Visual,

audio and tactile senses are widely included in VR, but the technology to produce olfactory stimulation is currently limited and still under development.

Olfaction has a strong influence over humans. The sense of smell is closely connected with the memory of an emotion. For that reason, it is important to achieve the goal of immersion by tricking the sense of smell. However, Olfactory simulation technologies have not reached that level yet and are still under development. Due to some limitations on the chemically based simulation as discussed, digital smell studies have grown in recent years. Real Feel [40] and Vaqso [133] devices are examples of digital smell devices which are specifically designed for VR headsets. They are able to create limited smell sensations as well as water mist, wind, heat and vibrations. However, a human being can differentiate one trillion different odours, while Feelreal can only simulate nine different, and Vaqso fifteen simultaneously. By comparing the natural olfactory system, it is quite low and not even close. For that reason, Olfactory displays are still under development to achieve the immersion by tricking sense of smell. When technology reaches or at least approaches to human smell perception, it can be used a wide range of VR applications including meditation, learning, movies and game VR applications to increase user's feeling of immersion. Game developers can use custom made smells and effects based on the content of the games. Smells and effects can also be added in VR movies without any programming skills.

1.3.4 Taste

Taste plays a major role in human life. There are nearly 10,000 taste buds on the tongue, roof of the mouth, and throat. These tiny bumps power our sense of taste. As a human being, we need food to survive. When we are eating, the taste of the foods directly influenced what we eat and how much we eat. Sense of taste, Gustation, is directly related to foods, and for that reason, it is one of the most important sense for life. Without sight, hearing or smell, people in our societies can still live a normal life. However, people who lack taste generally may not eat and may ultimately die if no medical treatment exists [24]. Research has shown that when human eat their favorite food, it triggers the production of β-Endorphin, which enhances the mood [34]. It explains why we feel happier when we eat chocolate. It is therefore stated that if food is the body, the taste is for the soul [34]. Moreover, taste also serves as a human defense mechanism. For instance, we judge the quality of the food on the basis of certain taste sensations and avoid decomposed food.

Despite the importance of the sense of taste in our daily lives, research and developments for taste stimulation in VR much weaker than for other senses. However, to be able to achieve the goal of immersion through tricking human senses, all of the human senses should be displayed in the VR.

At present taste sensations are difficult to use as a VR output by comparing other human senses, for instance, sight, audio and touch. Two main methods are in use to simulate the taste, which is chemical-based and non-chemical based approaches. Food simulator [24], Taste screen[86] and Virtual cocoon [36] are examples of the chemical-based approaches. However, there are some limitations such as limited availability of virtual tastes, refilling, cleaning, durability of the devices since the chemical content can be harmful to the device. Due to the limitations on the chemical based taste simulations, current research continues strongly on the basis of digital tasting devices. However, since digital tasting simulation technology is quite new, there is no specific method for it. Only three different, which are salty, bitter and sour achieved the simulate in Voctail whereas human tongue can approximately differentiate 10,000 tastes. The mixture of the taste sensation is difficult to predict due to complex interactions between the other senses; therefore, the sense of taste is currently considered the ultimate limit

for creating immersive VR [108].

1.3.5 Touch

Touch gives us our most simple tool to access the outside world [47]. It is a critical sense for a human being to connect with the physical world and plays a central role in human communication with the environment. Important details, for instance, warmth, coldness, softness and friction or more nuanced emotional contacts between humans can only be experienced through the sense of touch. Touch has often called "the sense" that helps us to differentiate between what is real and what is not [47]. This ensures that the tactile information given by the skin is considered real by default, and we always come to question how genuine a sensory signal is. The sense of touch leads to reduce the gap between the virtual and physical world [79]. It is important to trick the user's touch sense through haptic technology to achieve the goal of immersiveness in VR. Further on this will be discussed in Section1.4 , we begin by describing the human haptic perception that is related to haptic technologies. We then discuss tactile feedback, force feedback gloves and bodysuits. The chapter ends with a brief assessment of haptic technology for VR.

1.4 Haptic technologies for VR

Haptic technologies allow the user to touch and feel the objects that simulated in VE. Haptic word is derived from Greek "haptai" meaning to "touch" that convey important sensory information which helps users to identify virtual objects and move them to perform a task in VE [19]. The study of haptics is a consequence of advances in VR. VR is a type of HCI (Human Computer Interaction) that creates an immersive virtual world that can be experienced by direct interaction with our senses. When added to the visual and 3D audio feedback, haptic feedback dramatically improves the reality of simulation [19]. To be able to increase immersiveness in the virtual world, haptic feedback must be added. In HCI, Haptic feedback includes both force(kinetic) and touch(tactile) feedback as in the physical world.

1.4.1 Human haptic perception

Human sensation comes from the dermis layer of our bodies. The skin has a surface area of 18.000 square centimetres that is about 16-18% of the bodyweight of an adult; it is also the largest of our sense organs [94]. The dermis is lined with several sensory neurons' nerve endings [23]. When an entity touches or is touched, the sensors of the skin are activated by force. This force is then transmitted to the brain as a piece of information. A healthy human can feel about twenty different types of haptic sense of which hot, cold, pain are the most common ones. Some areas of the body contain more receptors than others, making them more sensitive to sensation. The tongue, for instance, has many nerve endings with pain and heat. For that reason, it hurts more when we bite our tongue and why we can burn our mouths quickly when we drink something hot. Other most sensitive places are palms, ears, fingertips and feet. Human haptic sensations can be classified into two major groups of kinetic(force) feedback and tactile(touch) feedback.

— Kinetic(force) feedback occurs when muscle, joints and tendons feel the forces. As a result of clinical studies [6] on human manual force exertion capability, it was found that males can exert a maximum force of 400 Newton while female hand strength is 60% to 80% of the males.

— Tactile(touch) feedback includes feedback through the skin, such as a sense of touch, temperature, texture and skin surface pressure. There are four different types of tactile sensors that are Meissner corpuscles (the majority), Merkel disks, Pacinian and Ruffini corpuscles. When they are excited, they create tiny electrical discharges that are perceived by the brain. Merkel and Ruffini are the most sensitive to lower frequency stimuli(0-10 Hz) whereas Meissner and Pacinian sensors respond to the higher frequency (50-300Hz)stimuli [19].

1.4.2 Tactile feedback interfaces

Tactile feedback provides real-time information about contact geometry, surface softness and roughness, friction and temperatures [19]. It does not actively resist contact movement of the user and does not prevent the user from moving through virtual surfaces. In the commercial side, the most common tactile feedback devices are hand controllers serving as navigation, pointing and selection. They also have a hand tracking system to increase interaction in the virtual world. However, when interacting with the virtual world with hand controllers, their tactile feedback is only limited by vibrations. HTC Vive Controller and Oculus Touch are the examples of the commonly used hand controllers.

CyberTouch glove

The CyberTouch, in Figure 1.10, glove provides vibrotactile feedback to the user. It has six vibrotactile actuators, one at the back of each finger and one on the palm of the hand. Each actuator has a plastic capsule containing a DE electrical motor. The motor shaft is vibrated by an off-centred mass that produces vibration. The vibration frequency can be changed between 0 and 125Hz. Each actuator has a small force (1.2 Newton) that can be felt through the singer's bones [19]. They can be individually programmed to vary the strength of touch sensation [30]. Whenever fingers or palms of the user interacts with the virtual object, the CyberTouch glove reads the user hand configuration and transmits the data to the host computer. Then the host computer sends the necessary informations for activating the vibrotactile actuators. In this way, the feedback loop is closed, and the user feels the vibrations on the skin. The glove enhances the user freedom of movement and sense of vibration significantly comparing to the hand controllers that require to kept on user's hands.

Figure 1.10 – Tactile interface by CyberTouch glove [30].

Temperature feedback glove

The temperature feedback gloves enable users to perceive thermal properties that can help to recognize the material of the product. These thermal properties are, for instance, temperature, thermal conductivity and diffusiveness. When grasped, materials with high conductivity feels cold and those of low conductivity, such as wood, feels warm. It is because of the heat transfer from the finger to the surface of the objects [19]. C&M Research developed a displaced temperature sensing system (DTSS) which consists of eight thermodes and control interface panel. Each thermode consists of a Peltier heat pump and a thermocouple temperature sensor attached to the skin-contact plates [19]. The fingertip temperature is determined by the thermocouple temperature sensors in real-time and then sent to the control system. The fingertip temperature is compared to the target temperature and sent to the DTSS control interface. The temperature difference between the target temperature and the user's fingertip temperatures is the input for the temperature controller system. The output is then forwarded to current amplifiers, which drive the Peltier heat pump and thermocouple temperature sensors. These thermocouple sensors are located in each fingertip and other locations such as on the palm. The host computer can modify the temperature according to the VR simulation requires.

HaptX

HaptX is another glove device which offers a convincing touch feeling to virtual objects by incorporating mid-air haptic technologies that touch the skin physically in a similar way real objects do. Mid-air haptic technology makes a simulation of texture, shape and motion of virtual objects. There are 130 points of feedback that displace user's skin up to 2mm [52]. User also can feel the size, weight of the virtual objects with the help of exoskeleton mechanism structure of the glove. It also produces force feedback to limit and resist the user's movements. However, it is almost impossible to replicate all data transmitted by touching a virtual object, including its surface and weight with HaptX glove.

1.4.3 Force feedback interfaces

Force feedback conveys information on compliance with the virtual object surface, object weight and inertia in real-time. It actively resists and can stop the user's contact movement with large feedback forces [19].

CyberGrasp system

CyberGrasp is an exoskeleton device. CyberGlove, as seen Figure 1.11, is the most well-known and precursor of all commercial exoskeletons [103]. It has a jointed structure that the user wears over his hand that transmits forces to his fingers. It is used to provide feedback from resistive force power. The system prevents the fingers from moving by applying the force resistance and not allowing the movements of the fingers when the user tries to move them. The system imitates the actual phenomenon when attempting to compress a rigid item in the physical world. The CyberGlove can also make users feel the weight to the virtual objects by using the same concept of force feedback. The interface box of the CyberGlove transmits the finger position data to the Cyber grasp Force Control Unit (FCU). The same FCU collects wrist-position data from the 3D magnetic tracker. Both finger and wrist 3D positions are sent to the host computer during the simulation. The host computer then senses inputs that are resulting from finger touch forces into the FCU. The Cyber Grasp FCU then converts the

contact force targets into analogue currents which are amplified and delivered to one of the five electric actuators in an actuator housing equipment. The actuator torques are conveyed to the user via a cable network and a mechanical exoskeleton that is placed on the top of the glove. The maximum force that can be generated on each finger is 16 N [19]. One of the main disadvantages of CyberGrasp system is their large weight and height. For instance, CyberGlove weight is approximately 0.4 kg [38]. It could be stressful for the user to do long simulations.

Figure 1.11 – Force feedback interface by Cyber glove [30].

1.4.4 Haptic bodysuits

In recent years, there have been several body haptic suits on the market; for instance, Tesla suit as seen in Figure 1.12, allow the user to touch and feel the sensation in Virtual Reality. The suit can actively stimulate heat, cold, vibration and tactile stimulation through a variety of electrical impulses by using conductive polymers and special add on materials as the primary structure of the suit. It has an ability to generate senses such as touch, pressure and pain by using two functional electrical stimulation: Neuromuscular electrical stimulation (NMES) and Transcutaneous electrical nerve stimulation (TENS) [10]. Touch sensation is mainly produced by the use of Transcutaneous Electrical Nerve Stimulation that is normally used in pain control treatments to block peripheral nerves [77]. Touch sense as tactile feedback can be simulated in bodysuits by varying frequency amplitude and pulse length. Moreover, Neuromuscular Electrical Stimulation is commonly used for muscle training and treatment for physiotherapy patients. It has the capacity to generate a sense of force or pressure on the user by producing muscle contractions, whereas TENS trigger a sense of touch depending on the virtual scenario.

Figure 1.12 – Haptic body suits - Tesla Suit [10].

1.4.5 Analysis of haptic technologies in VR

Sense of touch is the link between the physical world. It plays a crucial role in human contact with the environment. Since the VEs are designed to simulate physical worlds, it is important to achieve the goal of immersivess by tricking the sense of touch. Haptic technologies which we have discussed are capable of providing sensations not only for hands, wrist, and palms but also for the human body. However, the skin is the largest part of our body including ears and tongue. The Immersiveness is increasing when we involve more parts of the body in the simulation. Current VR haptic devices do not support HCI fully in real-time and new technological approaches needed to achieve the goal of immersion through tricking our haptic senses. However, there are a still number of limitations on all existing haptic devices. They have limited feedback capacity in comparison to the tactile sensory system in humans. The human hand consists of millions of specialist touch sensors all working in tandem, while the latest haptic interfaces typically have fewer than ten tactile feedback engines. Other limitations on existing haptic devices include a high price, large size and weight, bandwidth restrictions, latency between a human operator and force feedback, design for very specific purpose purposes and instability if the update rate is well below 1 kHz [38] . Nevertheless, it will still take some time, until VR haptic technologies achieve the goal of immersiveness through tricking sense of touch.

Immersive haptic technologies can be used in e-commerce. For instance, a human can test products by sensing the warm, cold, soft, hard, light or heavy properties of product's surfaces and textures. Since consumers usually tend to touch items (e.g. clothes) before purchasing. Haptic feedback is important for impaired people who are not able to VE through visual displays. Moreover, there is great benefit in meditation and training immersive VR haptic applications.

1.5 Existing haptic interfaces

Haptic technology, also known as kinesthetic communication or 3D touch, refers to any technology that can create an experience of touch by applying forces, vibrations, or motions to the user. These technologies can be used to create virtual objects in a computer simulation, to control virtual objects, and to enhance remote control of machines and devices (telerobotics). Haptic devices may incorporate tactile sensors that measure forces exerted by the user on the interface.

Haptic technology can communicate information to the user that could not be detected by sight or sound alone. Haptic sensation is what one first feels when touching something. Haptic sensation is also what allows us to determine that we are touching an actual object. We can use this human trait to help communicate information on substance and feel that could not be obtained through sight and sound alone. Haptic technology holds great promise as a means of bringing a greater sense of reality to games and virtual reality.

1.5.1 Haptic senses

The haptic senses refer to the perception of the variety of sensations that relate to touch: tactile sensibility, the perception of our limbs in space and the stress that they undergo, but also equilibrium, pain, and temperature [75] Those various signals are all integrated by the sensomatory system, giving us a global haptic image of our physical state and that of our environment.

Haptic technology facilitates the investigation of how the human sense of touch works by allowing the creation of controlled haptic virtual objects. Most researchers distinguish three sensory systems related to the sense of touch in humans: cutaneous, kinaesthetic, and haptic. All perceptions mediated by cutaneous and kinaesthetic sensibility are referred to as tactual perception. The sense of touch may be classified as passive and active, and the term “haptic” is often associated with active touch to communicate or recognize objects.

This section presents an overview of the sensory aspect of the human hand and focuses on the predominant sensations involved in object manipulation: cutaneous sensibility and proprioception.

Cutaneous sensibility

Cutaneous sensibility enables the perception of small-scale features such as texture and roughness. These sensations stem from mechanoreceptors, nerve endings encapsulated inside corpuscles that deform under stress, located below the superficial layer of the skin [35]. Structural variations in the capsule tissue make the four types of receptors react to different stimuli [75]. For instance, Meissner’s corpuscles are sensitive to changes in velocity. Pacinian corpuscles are sensitive to vibration and light touch. Ruffini’s corpuscles are sensitive to skin stretch and detect the direction of forces. Merkel’s disks are sensible to pressure and smallscale shapes, which helps in feeling the edges of objects. These mechanoreceptors can be found all over the body in varying quantities but the hands are among the most densely populated areas, which grants them a particularly acute sensibility [84].

The types of mechanoreceptors differ in other properties than the stimuli they respond. First, they can be distinguished according to their adaptation rate, which is the speed of the transition from the excited state to the neutral state. Rapidly adapting receptors quickly return to their neutral state. Thus, they are not fit to detect the material properties of a surface from a

static observation; which explains why we slide our fingers over a surface in order to perceive its texture. On the contrary, slowly adapting mechanoreceptors detect if an event is continuously occurring. They play an essential role in gauging the weight, balance, and slippage of the objects that we manipulate in order to ensure secure grasps.

Mechanoreceptors are also characterized by their input frequency, which corresponds to the speed at which separate stimuli can be detected. Rapidly adapting mechanoreceptors can respond to events between 20 and 300 Hz whereas slowly adapting receptors are typically limited to 10 Hz [19]. Additionally, the size of the receptive field, which corresponds to the skin area in which a contact is detectable by a single touch receptor, is larger for Pacinian and Ruffini's corpuscles. However, the larger is the receptive field, the lower the spatial resolution is [132].

Proprioception

Proprioception, or kinesthesia, is the perception of the motion of our limbs in space as well as the perception of the stress that they undergo; sensations which are essential for interacting with our hands [11].

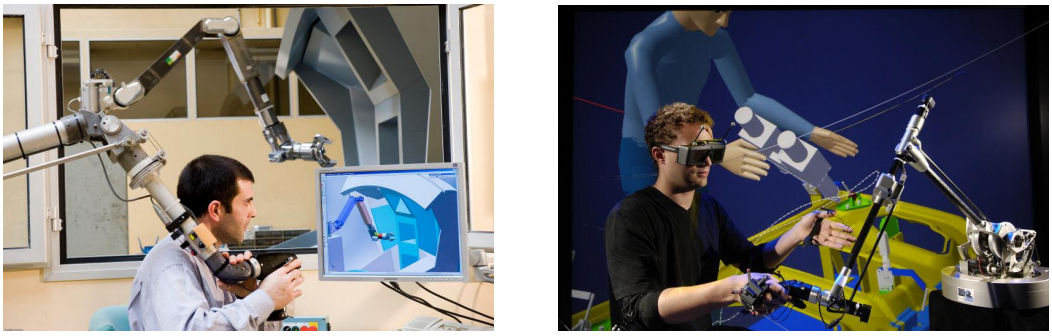
At the limb level, proprioceptive information comes from mechanoreceptors in the joint, the muscles, and the tendons. First, the same Pacinian and Ruffini's corpuscles that can be found below the superficial layers of the skin are also located in the joints between bones. The amplitude of the signal that they send to the brain informs about the joint angles between limbs whereas its frequency informs about their angular velocities [20]. Then, internal and external forces that our muscles are subjected to are evaluated by the Golgi organs located at the junction between muscles and tendons. They measure muscle tension, stabilize heavy grasps, and additionally play an inhibiting role by relaxing muscles when tension is too high. Other receptors, the muscle spindles, are located inside of the muscles of the arms and hands, and measure their length [75], which determines the shape and rigidity of the objects that we manipulate. The various mechanoreceptors of the skin responsible for cutaneous sensibility also play an indirect role in proprioception since extension/flexion of the limbs may result in a stretching/bulging of the skin adding extra information about our haptic state. At a higher level, the global position and orientation of each of our limbs is evaluated from the vestibular system in the inner ear. Canals filled with fluid subject to gravity inform about the orientation of the head. The perceived state of the rest of the body then depends on the proprioceptive links between head, neck, trunk and limbs.

The maximum frequency at which a human finger can apply output forces is between 5 and 10 Hz [16]. Regarding the input bandwidth, variable numbers were proposed: [118] suggested that it could go as high as 10 kHz and [16] supposed that a finger cannot discriminate two consecutive force signals above 320 Hz. It is however admitted that a input frequency between 20 and 30 Hz is a minimum for meaningful perception [121].

1.5.2 Haptic feedback and sense of touch

The sense of touch is very important in our daily life. Without it, it would cause serious difficulties in performing tasks such as holding and manipulating tools or making contact with surrounding objects [111]. Thanks to this sense, which gives us the ability to perceive the physical properties of our environment (shape, size, texture) and to control the forces applied to it, allows us to perform a wide range of tasks with a high level of dexterity.

However, in some situations, for example, the human operator may not have direct contact with the object being manipulated. Because the environment is hostile (nuclear, underwater,



(a) Teleoperation master arm with force feedback in a nuclear context (©CEA/Stropa)

(b) VR assembly testing for the automotive industry (©CEA/Stropa)

Figure 1.13 – Teleoperated and virtual reality (VR) systems using force feedback.

space) or not usable at the user's scale (micromanipulation, microsurgery). In such cases, tasks can be performed using a remote control system, a remote robot that mimics the movements of an operator-controlled master hand. The ability to control the movement of a remote system is also very important in virtual reality, in which case the user remotely controls the virtual avatar instead of the robot.

The concepts of remote manipulation (controlling a remote robot system), virtual reality (controlling a virtual avatar), and telepresence (makes a user feel like he/she is in a place other than their current location) are either remote (see Figure 1.13a) or Development of a system that allows the operator to feel the force and physical properties applied to an object that is virtually manipulated (see Figure 1.13a) [9], that is, i.e. by means of haptic interfaces.

Haptics was defined as a perceptual system that uses both cutaneous (including thermal) and kinesthetic inputs to derive information about objects, their properties, and their spatial layout [81]. To stimulate these sensations, users need special devices called tactile interfaces, devices that can interact with remote/virtual environments by reproducing touch using kinesthetic (force/position) and skin (tactile) receptors [51].

There are four distinguish methods for artificially creating haptic sensations : vibrotactile devices, force feedback systems, surface displays and distributed tactile displays [56]. The following section will describe their main applications with a focus on VR.

1.5.3 Examples of force feedback devices

Force feedback interfaces are used in many domains (most of them using VR) such as design, manufacturing, assembly, scientific visualization, entertainment, education, medicine, space, rehabilitation, micro-manipulation, as well as molecular biology [51]. The interface should then provide adequate kinesthetic information contributing to enhance the sense of presence in the VE [109].

In the fields of design, manufacturing and assembly, haptic feedback can contribute to changing traditional product development approaches by allowing the users to get the feeling of touching objects, perceiving the nature of their surfaces and their dynamics before producing any real prototype [139]. This way, users can gain a comprehensive understanding and accurate evaluation of the design and manufacturing process (see Figure 1.14).

In scientific visualization (see Figure 1.14b), haptic devices can be used to simulate object's physical properties, e.g. texture and/or interaction forces. This allows direct and immediate

control over simulations as well as sensing of the results of the scenario of interest [128].

The entertainment industry has also benefited from haptic developments. Force feedback technology enhances the game experience by providing more realistic sensations while playing a game [37].

Haptics is also of great interest in education. For example, a multimedia system for learning handwriting of alphabet letters/characters in different languages can make use of a haptic device to provide force feedback to guide the user's gesture at following a pre-recorded letter trajectory (see Figure 1.14c).

In medicine also, VR simulation-based training appears as a promising solution for skill transfer. Novice surgeons can acquire and/or improve their skills before operating on a real patient and experienced surgeons can learn new techniques and even rehearse when the patient to operate presents a complicated surgery case (see Figure 1.14d), e.g. a congenital anomaly or a known difficult anatomy [4].

Force feedback interfaces can also be used to intelligently guide/regulate the motion of the user during an operation [15], e.g. preventing the user to move towards a restricted region, contributing that way to improve the security of the patient and postoperative results.

More generally, robot-assisted minimally invasive surgery (RMIS) offers advantages to patients such as less trauma, shorter hospital stay and reduced recovery times [106]. Robots in master-slave configuration can cope with motion constraints of surgical instruments, improving surgeon's dexterity [105]. The surgeon is expected to act and feel as if he/she were holding directly the surgical tools being in contact with the patient. In such scenarios, any strategy to estimate/ sense the applied force is expected to improve the accuracy and dexterity of the surgeon [98] and therefore reduce the tissue trauma and organic damage. The development of force sensors for RMIS is thus still an active research topic as reported in [105].

One can also refer to physical rehabilitation which is a growing field of use for haptic interfaces. Unlike a human therapist, robots can train patients for long periods of time without tiring. Robotic systems coupled with VR simulations also bring additional improvements to today's conventional physical therapy methods, since they introduce objective measures of performance (see Figure 1.14e).

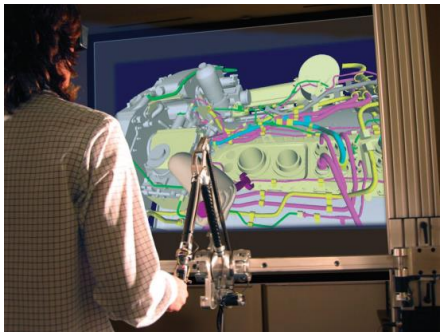
Space is an environment inaccessible to humans where force feedback systems can contribute to train operators and help them performing maintenance, reparation or exploration tasks while preventing and reducing risks. In [113] for instance, a VR platform for telerobotic on-orbit servicing missions helps to train users (providing visual and haptic feedback) for satellite maintenance tasks (see Figure 1.14f.

When it comes to manipulation of tiny objects, haptic systems allowing the user to interact with objects at the micro scale are needed (see Figure 1.14g). In molecular biology for example, the size and complexity of molecular structures make it difficult, if not impossible, to show all of their features in a physical model alone. In combination with augmented reality (AR) and voice commands, haptic feedback provides additional information about these models using a force display [116]

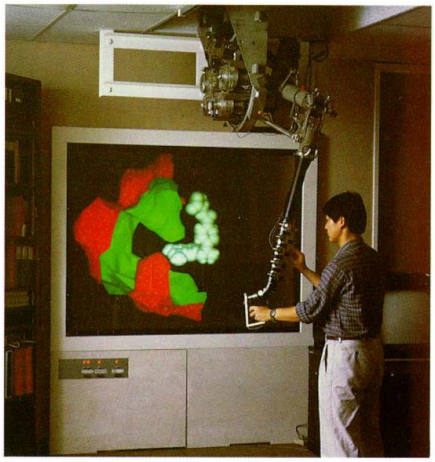
All the above mentioned applications require a device compatible with the targeted task. Several examples of such devices are presented in section 1.5.4, with focus on commercially available interfaces.

1.5.4 Commercially available interfaces

Force feedback systems are robotic mechanisms capable to measure the user's movements, this information being used to control the movements of user avatar, and deliver a force signal



(a) Virtual aircraft engine maintenability operation

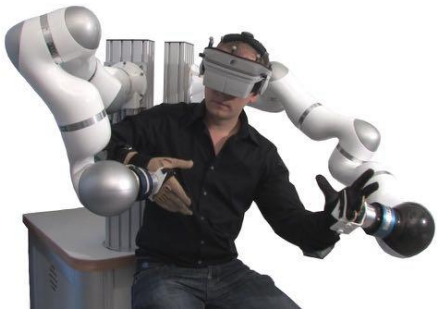


(b) GROPE III system used for scientific visualization

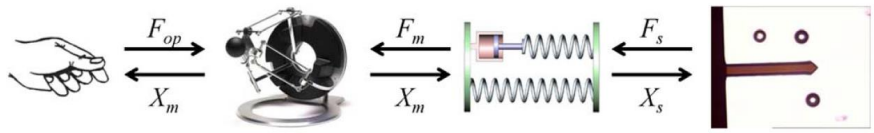
(c) Telemaque platform used for children's handwriting acquisition

(d) Training platform for maxillo facial surgery

(e) MIT MANUS for shoulder rehabilitation



(f) HUG for VR on-orbit servicing missions



(g) Teleoperation for micromanipulation

Figure 1.14 – Applications of force feedback interfaces: (a) [14], (b) [17], (c) [101], (d) [49], (e) [18], (f) [113], (g)[13]

to his/her hand, usually through a stylus or a thimble [21].

The realism of the interaction is strictly related to the capabilities of the device to allow natural movements of the human operator's hands as well as to reproduce adequate stimuli on them. An ideal force feedback system should be transparent: the user should be able to move in free-space without feeling any force, i.e. to make pure free movements, and the device should prevent him/her to move in the constraint direction if a stiff object is being touched, i.e. the device should be able to simulate any type of contacts, even hard ones [29].

It is worth noting that a perfect transparency cannot be attained in practice. Fortunately, most of the time it is not necessary to reproduce the whole bunch of physical interactions experienced everyday, and force feedback interfaces can be tuned to the requirements of the tasks of interest. It is thus important to identify what sources of kinesthetic information are relevant for the targeted tasks, and which degree of fidelity is needed [111].

Several performance metrics are usually employed to qualify a force feedback interface: workspace, position resolution, continuous force, peak force, friction, apparent stiffness, apparent inertia and bandwidth. All these notions, defined below with reference to e.g. [95] contribute to the transparency of the device.

— The workspace is the volume that the device can cover, considering both translations (ranges usually expressed in the Cartesian space in mm, cm or m) and rotations (ranges usually expressed in degrees in yaw, pitch and roll).
— The position resolution is the smallest amount of movement which can be detected by the arm sensors. It is usually expressed at the level of the handle (i.e. in the Cartesian space) in m or mm in translation and in degrees or radians for the rotations.
— The continuous force is the amount of force/torque that the device can exert for an extended period of time. It is usually expressed in the Cartesian space in N for the translations and in N.m for the rotations.
— The peak force is the maximum force that the actuators can exert, e.g. over a small period of time. The peak force can typically be between 2 and 10 times higher than the continuous force for typical actuators usually implemented in force feedback interfaces, allowing to momentarily stop the user against the environment, e.g. when taping or touching a hard object.
— The friction is a measure of the resistance of the device to movement. It is expressed in N for translations and N:m for rotations. It should be as low as possible.
— The apparent stiffness is the maximum stiffness that can be simulated by the device. It informs on the different types of objects that can be simulated. Indeed many objects in VR environments behave like a spring, i.e. they deform linearly in response to an external force. The apparent stiffness is expressed in N/m for translations and N.m/rad for rotations. The higher it is, the better solid virtual walls can be simulated.
— The apparent inertia is a measure of the mass (in kg) and inertia (in $kg.cm^2$) that is sensed by the user when he/she tries to move the handle of the robot. It should be as low as possible in order to give user the impression that he/she moves freely in free space.
— The bandwidth is the frequency (in Hz) up to which the device can be properly controlled. It should not be confused with the update frequency of the controller nor with the bandwidth of the human somatosensory system, i.e. about 300 - 1000Hz for tactile sensing and 20 - 30Hz for kinesthetic and proprioceptive sensing.

It is worth noting that, even if the afore-mentioned data are usually given in the Cartesian space, this information can be given instead in the joint space.

Six main manufacturers were identified in the market: 3D Systems, Force Dimension, Hap-

tion S.A., Quanser, MOOG Inc. and MPB Technologies Inc. Representative examples of their products will be briefly described below.

The Touch device (see Figure 1.15a, left) is a desktop interface providing 3DoF force feedback. This low power device can be used in several applications, including 3D modeling, training, skill evaluation, virtual assembly and robotic control [1]. The premium series provide 6 active DoF and cover a vast range of research and commercial applications, e.g. virtual prototyping, maintenance path planning and molecular modeling applications. Within this category, the Phantom Premium 3.0 provides a range of motion compatible with full arm movement pivoting at the shoulder (see Figure 1.15a, right).

The omega.6 (see Figure 1.15b,left) provides 3DoF force feedback. It is considered by Force Dimension as the most advanced pen-shaped force feedback device available. Focused on ergonomics, this device enables the rendering of high contact forces and high stiffness thanks to an update rate of 4kHz. The sigma.7 (see Figure 1.15b, right) is a high sensitivity 7 active DoF device from Force Dimension that can be used in applications including medical and space robotics, micro and nano manipulation, bi-manual teleoperation, virtual simulations, training systems and research [43].

Haption S.A. provides a wide range of high performance force feedback interfaces. It proposes 3 or 6 DoF devices like the Virtuose 3D Desktop (see Figure 1.15c, left) and the Virtuose 6D TAO that can be used for VR and teleoperation applications (see Figure 1.15c, right). It is particularly well suited for scale 1 virtual/remote manipulations. It is also used for comanipulation in laparoscopic surgery as well as in rehabilitation [134].

The High Definition Haptic Device (HD^2) from Quanser (see Figure 1.15d) is a high-fidelity 6 DoF haptic interface for advanced research in haptics and robotics. It is particularly suitable for the development of test beds for various emerging applications such as medical simulators and teleoperation. This haptic interface can track the operator's motion in 6 DoF and apply force to the user in 5 DoF [107].

The Desktop Haptic Interface from MOOG, Inc. (see Figure 1.15e) is a general purpose haptic interface, providing force feedback in 3 DoF. Its admittance control allows to provide realistic crisp touch and feel [31].

Finally, the Freedom 6S (see Figure 1.15f) from MPB Technologies Inc. is a high-fidelity force feedback device operating in 6 DoF, providing the user with the sense of touch in both virtual and real-world applications. It is ideally suited for medical and master /slave robotics [95].

It is worth noting that the development of haptic devices is part of an increasing market. More and more applications are being imagined every day, generating the continuous need for novel devices.

1.5.5 Limitations of existing haptic interfaces

As mentioned earlier, the user should not feel the presence of the robot in free space. All existing devices lack the design to answer this need. As a consequence, a perfect transparency is not attained.Transparency in free space could be improved by implementing accompanying control strategies.

Part of the problem is related to the fact that tactile feedback interfaces usually require the user to be mechanically linked to them in order to know the user's position and provide feedback that is consistent with the virtual scene. This link has a non negligible influence as the user experiences mechanical structure even when moving in free space. In this case the difference

(a) Touch and Phantom Premium 3.0 from 3D Systems, U.S.A.[1]

(b) omega.6 and sigma.7 from Force Dimension, Switzerland.[43]

(c) Virtuose 3D Desktop and Virtuose 6D TAO from Haption S.A., France.[134]

(d) HD2 High Definition Haptic Device from Quanser, Canada.[107]

(e) Desktop Haptic Interface from MOOG, Inc., U.S.A.[31]

(f) Freedom 6S from MPB Technologies Inc., Canada.[95]

Figure 1.15 – Examples of common commercial force feedback interfaces.

between free space and contact is less distinctively felt and muscular fatigue is expected to appear in case of long manipulations.

A relatively novel paradigm known as intermittent-contact aims to cope with the limitations of conventional devices in order to improve the realism of the haptic interaction and relief the user when long manipulations take place. Its principle and the advances in this field of research will be described in the next section 1.6.

1.6 Intermittent-contact interface

The intermittent-contact (IC) model proposes to remove the mechanical link between the interface and the user in free space, obtaining a perfect transparency and letting the user touch the haptic device only when a contact occurs in the VE. This way, intermittent-contact interfaces (ICIs) aim to improve the realism of the haptic interaction.

Two categories of devices implementing the IC model can be identified. The first one is known as Encountered-Type haptic interfaces (ETHI) and the second one as Close-Tracking-type haptic interfaces (CTHIs). In the former, a robot manipulator, which end-effector has the shape of the simulated object, is controlled to make it be encountered the user's hand at the position where the VO is found. In the later, the device follows the user at a short and constant distance, without contact, and touches user only when a VO is reached.

These devices are particularly interesting to reduce the fatigue during long manipulations and to perceive small interaction forces as it is the case in medicine, e.g. the force applied on an organ. Their principle and relevance will be discussed in sections 1.6.1 and 1.6.2 respectively.

1.6.1 Encountered-type haptic interfaces

The ETHI principle consists in imagining that the virtual space is superimposed with the real environment. The robot is then controlled to be positioned along its surface at the nearest position of the user's hand and simulate the VO properties (see Figure 1.16). This way, the user remains away from the robot in free space and he/she touches its end-effector only when and where he/she is supposed to touch the VO, a perfect transparency is therefore achieved. To do this, the user's position must be tracked, e.g. using video cameras or body-mounted position tracking sensors, and the robot must be controlled so that it can rapidly and accurately anticipate the user's movement intentions.

The ETHI approach was first proposed in [87]: a robot manipulator, which end-effector has the shape of the simulated object, is controlled to make it encounter the user's hand at the position where the VO is expected to be found. This way the user is totally free in free space and he/she can feel the corresponding interaction forces.

The ETHI approach was also used in [127] for the construction of a haptic space: the user's arm motion is measured by a passive master arm, allowing to compute the position and orientation of the user's fingertip. This information is used by a computer to calculate the nearest object to the user's hand in the virtual haptic space (see Figure 1.17). The local geometry of the simulated object near the contact point and its mechanical impedance are displayed by a 6DoF impedance controlled manipulator and a shape approximation device (SAD). The SAD possesses curved surfaces as well as convex and concave edges and its positioning and orientation are modified in order to simulate the desired object configuration.

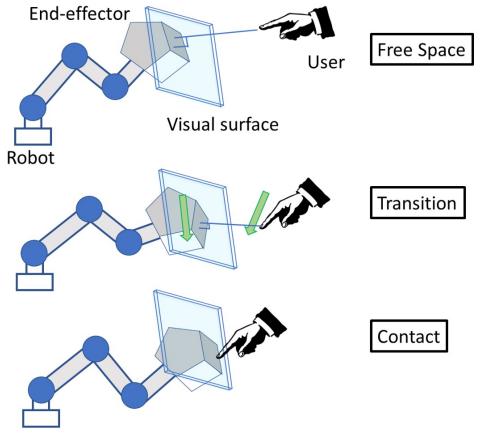


Figure 1.16 – Principle functioning of encountered-type haptic interfaces.

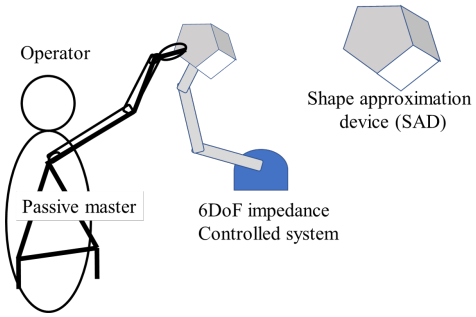


Figure 1.17 – Virtual haptic space representation (adapted from [127]).

Another ETHI system was proposed in [140][141], called WYSIWYF (What You can See Is What You can Feel), this concept allows the user to feel the object precisely where the user sees it. Employing vision-based tracking, the system can blend live video with the virtual scene, i.e. a portion of the user's hand is extracted from the captured image and is superimposed on the virtual scene (see Figure 1.18).

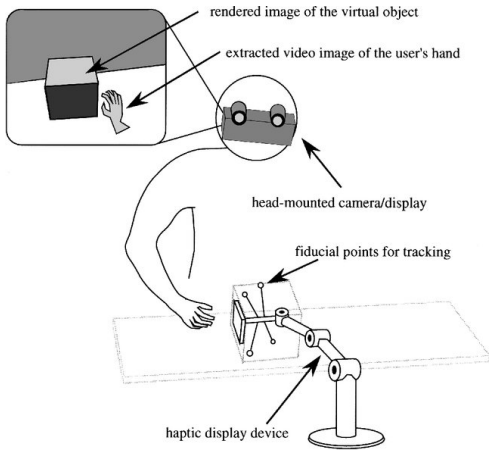
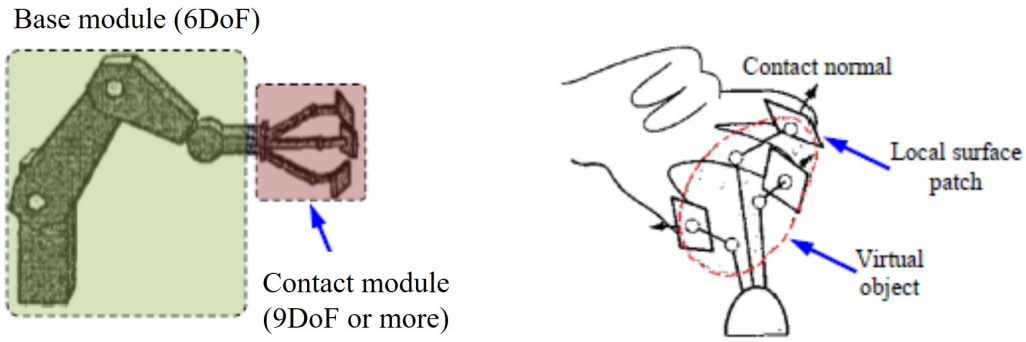


Figure 1.18 – Virtual haptic space representation [141].

In [143], an ETHI device allowing the simulation of precision grasping tasks using three fingers (thumb, index and middle fingers) was presented (see Figure 1.19). The system consists of two modules, a base module composed of a robot manipulator (see Figure 1.19a) and a specifically designed contact module for finger interaction (see Figure 1.19b), together accounting for the necessary number of DoF to place a local surface patch any where in space. The positioning and orientation of the system are predicted based on the measure of the wrist speed profile and aperture distance between the fingers.

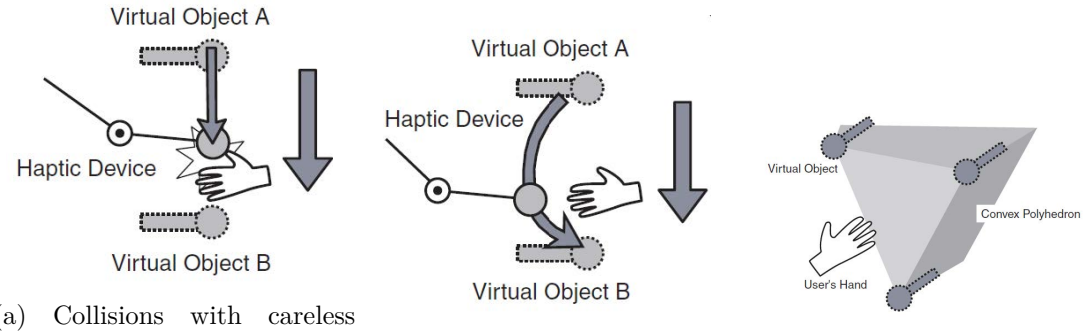


(a) System modules (b) Contact module diagram

Figure 1.19 – Multiple finger ETHI [143]

When working with ETHI devices, the user's hand is a kind of moving obstacle (see Fig.1.16a) that should be avoided (see Fig.1.16b). Miscalculations or system slowness can lead to painful collisions. In order to cope with this issue, path planning techniques are necessary.

In [142], a path planning algorithm for ET-HI devices that renders multiple VOs in 3D space was proposed. Here a convex polyhedron is constructed from the reference points of the virtual objects (see Fig.1.16c). As long as the user's hand is outside the polyhedron, the device stays on its surface. On the other hand, if the user's hand penetrates in it, the device avoids the user's hand and goes inside the polyhedron if necessary. Such algorithm not only considered the security of the user but also the efficiency of the device movement.



(a) Collisions with careless path-planning (b) Collision avoidance (c) Polyhedron principle

Figure 1.20 – Path-planning issue in ETHIs [142]

As previously explained, ETHIs have the advantage over classical type haptic interfaces to

be perfectly transparent in free space as they are not at all in contact with the user's hand. With this approach however, the robot may reach very high speeds, which is dangerous for the user, e.g. if bilateral contacts with concave surfaces have to be simulated. Another issue is that the amount of objects that can be simulated is limited to a restricted set of pre-manufactured objects, which often display only very simple mathematical shapes.

1.6.2 Close-tracking-type haptic interfaces

The CTHI approach is another way to implement the IC model. Contrary to the ETHI principle where the device follows the surface of the VO, it proposes to let the device follow the user at a short and constant distance, without contact, and touch user only when a VO is reached (see Figure 1.21).

The CTHI principle requires to measure the distance between the device and the user's body (typically user finger), using e.g. sensors installed on the end-effector. The device will then closely follow the user's movements without any path planning as long as the user remains in the reachable volume (workspace) of the interface. Despite this limitation, the security of the user is improved. Several examples of such devices will be given below.

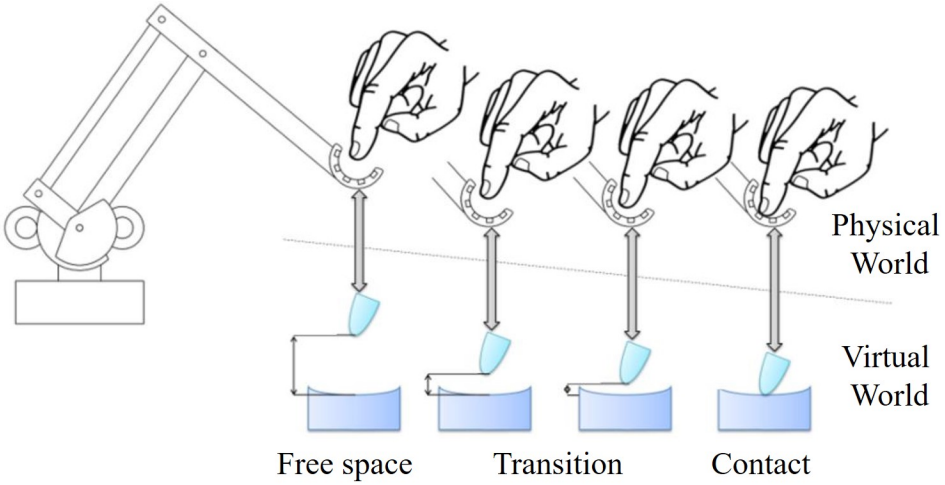
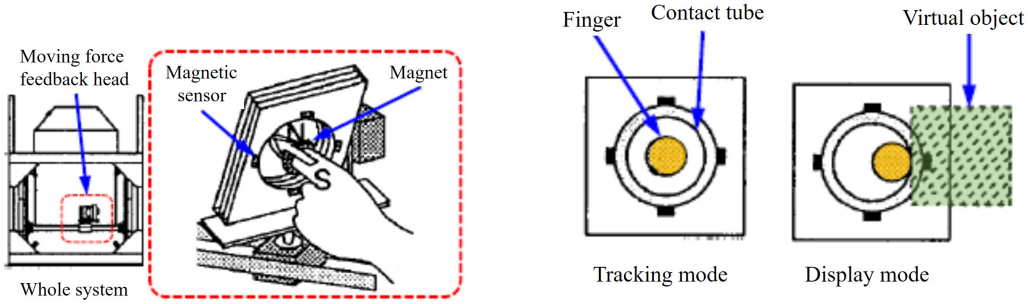


Figure 1.21 – Principle of functioning close-tracking-type haptic interfaces.

In [59], the existence of an object is simulated thanks to a mechanism for surface displaying (see Figure 1.22). 3-DoF tracking of the index finger is accomplished by means of magnetic sensors integrated in a moving head and magnets mounted on the finger (see Figure 1.22a). When the finger is being displaced in free space (i.e. no VO is being touched), the moving force feedback head follows it (tracking mode).

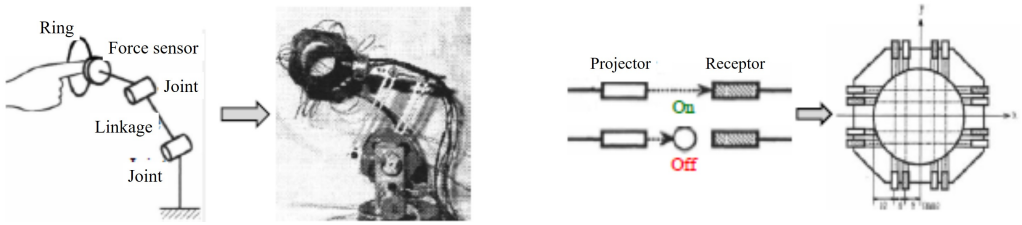
When a VO is about to be touched, the display mode is activated, the force feedback head stops at the VO position and waits for the finger to contact its inner surface composed of a tube (see Figure 1.22b).



(a) Detail of force feedback head (b) Control

Figure 1.22 – Surface display [59]

In [144], a 2-DoF arm (see Figure 1.23a) allows finger interaction with virtual worlds thanks to a ring-like end-effector equipped with a set of eight lightweight optical on-off sensors used to roughly estimate the fingertip position in the ring (see Figure 1.23b). When the finger is far away from a VO, the ring's position is controlled to keep the fingertip in its center. When the fingertip is near a VO, the ring moves closer to it. Finally, when the fingertip is in touch with a VO the display mode is activated (see Figure 1.23c). A force sensor placed between the ring and the tip of the arm allows then to measure the force applied by the arm on the fingertip.

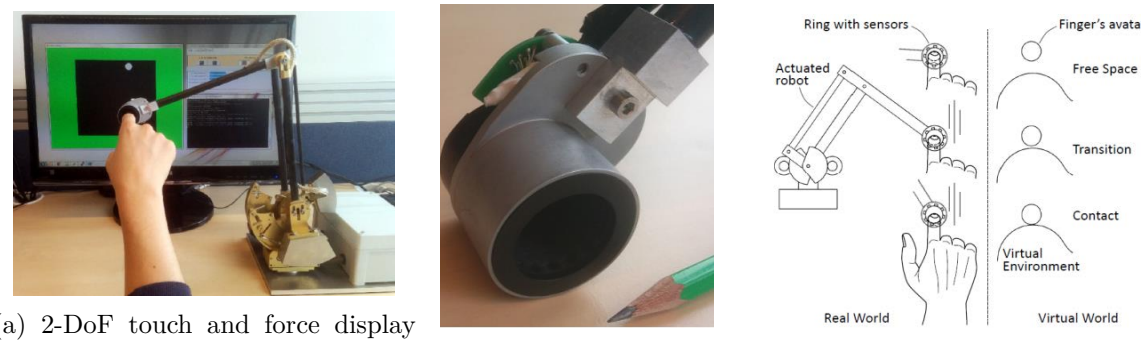


(a) 2-DoF touch and force display system (b) Sensor principle

(c) Control

Figure 1.23 – 2-DoF CTHI systems, adapted from [144]

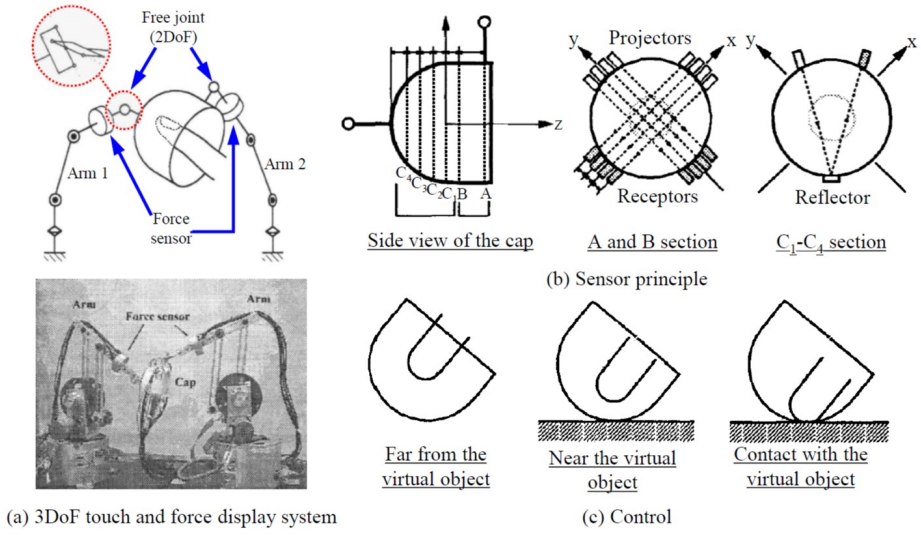
A similar approach was used in [48] for the development of a 2-DoF CTHI (see Figure 1.24a). The tracking is however improved to get better performances: the end-effector is instrumented with infrared proximity sensors allowing to reconstruct the finger shape and to precisely estimate the position of its center using distance measurements (see Figure 1.24b). The control of the device is more precise, especially regarding the transitions between free space and contact (see Figure 1.24c).



(a) 2-DoF touch and force display system (b) Sensor principle (c) Control

Figure 1.24 – 2-DoF CTHI system, adapted from [48]

In [144], a three-dimensional haptic display composed of two arms and a cap-like end-effector attached to their tips was presented (see Figure 1.25a). The user can insert user finger in the cap to interact with the virtual world. The device's sensor system is composed of optical glass-fiber on-off sensors, allowing to roughly estimate the 6-DoF configuration (position and orientation) of the finger in the cap (see Figure 1.25b). In free space the device tracks and follows the fingertip without contact. At contact (when a virtual object is touched), the cap is controlled so its inner surface touches the finger (see Figure 1.25c).

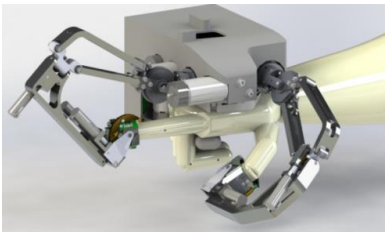


(a) 3DoF touch and force display system (b) Sensor principle (c) Control

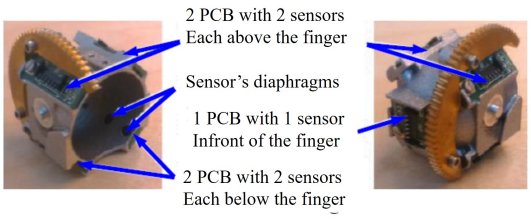
Figure 1.25 – 3DoF CTHI system, adapted from [144]

Finally, a recently developed CTHI presented in [22] allows two finger dexterous interactions (index and thumb fingertips) with digital mock-ups in VR (see Figure 1.26a). This interface should be able to precisely measure and follow both fingertips remotely without equipping the user with any marker on skin or nail, and serve as a contact surface when force feedback is required. To perform this, two 6-DoF end-effectors were specially designed and tested. Each end-effector integrates 9 proximity sensors, 8 placed around the finger in two planes, the last one being in front of it (see Figure 1.26c). A 6-DoF (position and orientation) robot is associated with each finger and connected to a base plate fixed on the hand palm. All these elements are

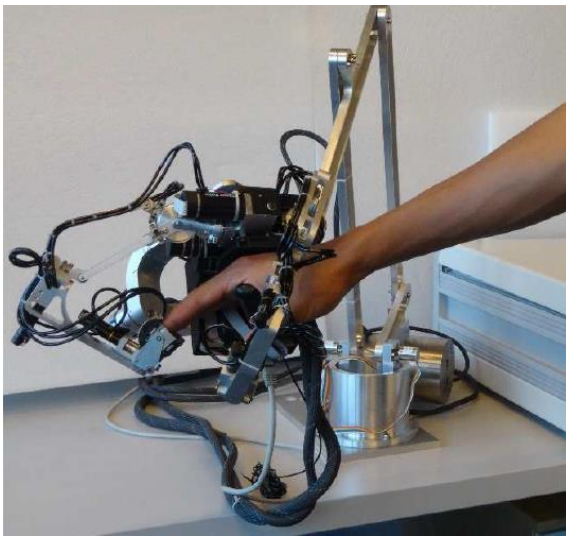
associated with a positioning arm (see Figure 1.26b) in order to allow for as free as possible hand movements.



(a) Two fingers close-tracking-type haptic interface.



(c) Prototype end-effector with 9 proximity sensors.



(b) Interface main elements.

Figure 1.26 – Two finger dexterous CTHI, adapted from [22].

1.7 Equipment

In this thesis, it is expected that specific hardware and software will be used. Some basic knowledge of this software is also required and challenges regarding installation are also expected. As explained in "Description of Context", this thesis is a part of Renault Group project. The selection of hardware components like robot, VR systems and software platforms are already established. Since the consortium had selected Universal Robots UR-5 as the collaborative robot and the use of HTC Vive for Virtual Reality display. Similarly the car model used in this thesis and definition of iteration points are also established by Renault Group.

The use of Unity3D software for virtual reality scene development has also been established. Since the final goal is to collaborate and combine the thesis with Renault Group project these established hardware and software have not been challenged or changed. For robot planning in this thesis, ROS has been selected because of its versatile features. This poses a challenge in cross-platform between Windows and Linux which is also addressed in Chapter 4.

1.8 Research proposal

Haptic feedback interfaces are intended to allow a user to interact naturally with a virtual/remote environment with haptic feedback. Therefore, they should be both light and safe enough so that they do not interfere with the user's movement in free space (transparency) and mechanically stiff to provide realistic feedback. Several efforts can be made to achieve these goals, e.g. by optimizing the mechanical design or by using force sensors (in case of force feedback systems) to measure and compensate the resistance of the device to the user's movements in the direction of displacement. These solutions are however not sufficient to achieve a perfect

transparency and the performances of current feedback devices still require to be improved, in particular for tasks lasting long hours.

The ICI model, presented in section 1.6, aims to solve this problem by physically disconnecting the user for the device in free space and letting user touch the robot only when it is required, this way providing only the necessary amount of interaction. This principle can theoretically render feedback interfaces perfectly transparent, allowing to increase the realism of the interaction. It has been implemented in two ways: ETHI and CTHI devices.

In the former approach, the system anticipates the user's movements, positions the end-effector of the robot at the place where he/she is expected to touch a VO and waits for the user to encounter it. User tracking is commonly achieved by means of a motion capture system or a passive mechanical master arm attached to the user's arm. Such approach is however dangerous, as without meticulous path planning, the robot may follow trajectories near the user. Furthermore, if bilateral contacts are simulated the robot may attain very high speeds, compromising the security of the user.

Therefore intermittent contact interfaces, should be safe enough so that they do not interfere with the user's movement in free space (transparency). Several efforts can be made to achieve these goals, e.g. by optimizing the mechanical design (robot base placement wrt to the task points, planning and control of the robot trajectories)or by using sensors to measure and predict the user's movements in the direction of displacement. These solutions are however are a step forward to achieve a perfect transparency and the performances of current haptic feedback devices.

The later approach proposes to closely track (advantageously with a system integrated in the end-effector of the robot) and follow the user's movements (in position and orientation) without contact in free space and stop the robot when a VO is being touched to make the user feel the interaction forces. With this approach, the security of the user is improved, even if user movements are restricted to the workspace of the robot. However the system restricts the interaction of user with only maximum of 2 fingers. For this reason, in the present work we will focus on Encountered-type haptic interfaces.

It is worth noting that, despite their potential advantages, ETHI devices still suffer from some limitations. The objective of the doctoral thesis presented here is to implement a system for the texture perception. It is organized around three key questions that, according to the explored literature, have not yet been addressed for this type of devices. Each of these issues will be addressed in a specific chapter of this thesis, as explained in "Introduction".

Chapter 2

UR-5 configurations, workspace and placement

2.1 Introduction

Serial manipulators are used in many robotic systems like manufacturing, handling material, and teleoperation. There is an increasing number of these robots worldwide with some big names in that domain such as ABB and Kuka.

In recent years, Universal Robots have developed a series of robotic manipulators that are now widely used by many universities and industries. These are a special kind of serial robots which have a cobot mode. The ability for the robot to have a protective stop when violating a set defined boundary. Additionally this robot is claimed to be fast, easy to program, flexible, safe, and offers low-level programming access to the robot controller with a high cycle time [1]. Among the UR products, the family of UR-3, UR-5, and UR-10 have received great attention within the robotics community and industries specifically by the robotic research community.

UR-5 robot, Figure 2.1, is a well-known 6-degree-of-freedom (DoF) robotic manipulator manufactured by Universal Robots Company [130]. The most renowned feature of this robot is its agility due to its lightweight, speed, ease of program, flexibility, and safety. One of the main characteristics of the UR-5 is that the last three joints on it do not act as a coincidental wrist. Therefore, all its six joints contribute to the transformational and rotational movements of its end-effector.

In Figure 2.2 the schematic of the robotic arm and the allocation of each joint's frame are illustrated. It is mentioned that the DH parameters that are used in the kinematic model are based on these frames. The DH parameters of the UR-5 for the specified joint frames in Figure 2.2, are presented in Table 2.1.

Table 2.1 – Denavit-Hartenberg parameters for the UR arms [130].

i	a_i	α_i	d_i	θ_i
0	0	0	–	–
1	0	$\pi/2$	d_1	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3
4	0	$\pi/2$	d_4	θ_4
5	0	$-\pi/2$	d_5	θ_5
6	–	–	d_6	θ_6

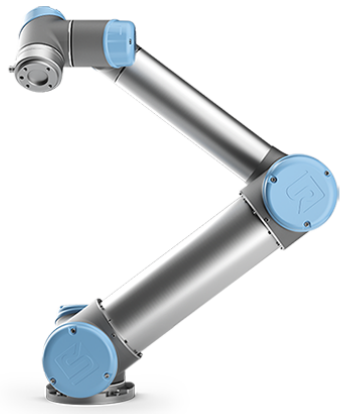


Figure 2.1 – The real UR-5 manipulator [130].

The context of study is the evaluation of perceived quality of a virtual car interior during the first design phases.

In a given scenario, the user sits in the real world with a visual virtual reality experience inside the car. The user wears a helmet and cannot see the robot, which explains the safety problem (Figure 2.3). While the user is trying to interact with the virtual object in the environment, the robot must come and position a sample of the material associated with the local surface, to provide a tactical sense of touching the object. Currently, the prop can carry six different materials (Figure 2.4). There is a cushion surface, a glossy glass surface, two different textures of rubber, a simple plastic surface and a smooth leather surface.

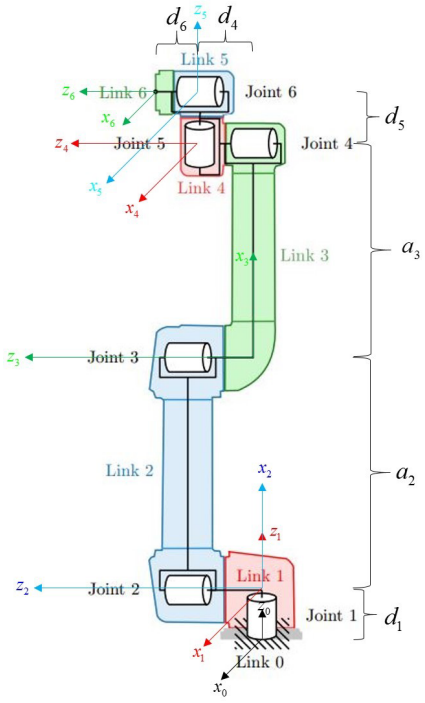


Figure 2.2 – Coordinate frames for UR-5 manipulator [130].

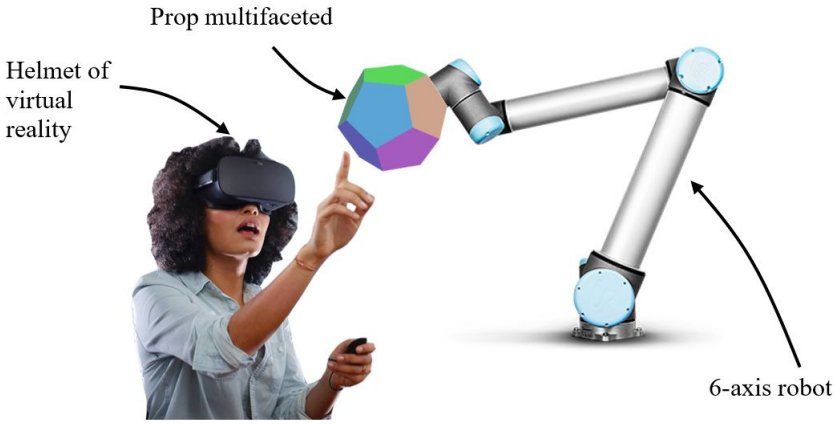


Figure 2.3 – Conceptual scheme of the experimental platform.

(a) (b)

Figure 2.4 – (a) The UR-5 robot with a prop attach to the end-effector and (b) The prop used to carry six sample textures

So in this scenario, we want the robot workspace to cover all the virtual environments (in this case the interior of the car) the user could interact. Figure 2.5 depicts inside a Dacia Duster and three areas to touch.

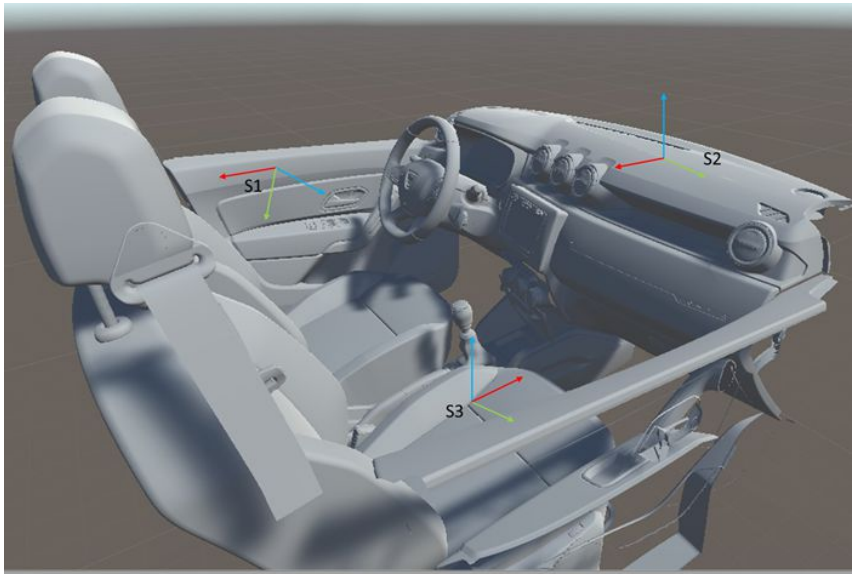


Figure 2.5 – Example of virtual environment and a set of task to be reached.

Figure 2.6 – The complete system setup for human robot interaction.

2.1.1 Safety standards

Possible collaboration of humans and robots in shared workplaces without perimeter guarding opens up new possibilities and concepts. The standardization and legal situation allows for human-robot collaboration (HRC) within defined limits.

Basics of Human-Robot Collaboration (HRC)

A collaborative robot is designed for direct interaction with a human within a collaborative workspace, meaning a common workspace where the robot and a human work simultaneously. In comparison to a traditional robot system, operators can work in close proximity to the robot system while power to the robot's actuators is available, and physical contact can occur within a collaborative workspace. The term collaboration refers to cooperation between humans and robots. This cooperation is limited to a precisely defined collaborative workspace.

Multiple definitions for levels of interaction between humans and robots exist. In one of the first definitions Helms et al. identified four types of interaction between a human and an industrial robot [58]:

- **Independent**: The human and the robot work on different workspace as in a traditional robotic cell.
- **Synchronized**: The human and the robot work consecutively on the same workspace in separate areas.
- **Simultaneous**: The human and the robot work on the same workspace at the same time without having physical contact.
- **Supportive**: The human and the robot work cooperatively together on the same workspace.

Aaltonen et al. proposed a new framework based on a literature review and case study examples [2]. They suggested four levels of HRC which are further divided into sublevels through the factors: workspace sharing, robot's activity when the human is present, type of joint effort, and physical contact. This framework includes a wider variety of possible forms of HRC but encompasses a high level in detail and complexity. Wang et al. classified the relationship between human and robot based on five characteristics [137]:

- **Shared workspace**: working in the same workspace without fences.

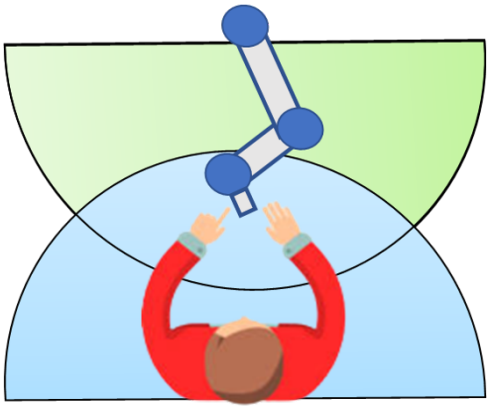


Figure 2.7 – Interaction between robot and human.

— **Direct contact**: direct physical contact between both entities exists.
— **Shared work task**: working on the same task but not simultaneously.
— **Simultaneous process**: working at the same time on the same or a different task.
— **Sequential process**: working one after another, meaning in sequential order.

In this thesis work, the term HRC refers to combination of supportive interaction with simultaneous process having direct contacts, shown in Figure 2.7.

2.1.2 Safety in human robot collaboration

The four modes of collaborative operations were first introduced in ISO 10218 and have been further detailed in ISO/TS 15066. Collaborative operations should include one or more of the following four safety methods: 1) Safety-rated monitored stop, 2) Hand-guiding, 3) Speed and separation monitoring (SSM) and 4) Power- and force-limiting (PFL). ISO/TS 15066 defines them in detail as follows [62]:

— Safety-rated monitored stop; robot stops when operator enters the collaborative workspace and continues when the operator has left the collaborative workspace.
— Hand guiding robot; movements of robot are controlled by the user.
— Speed and separation monitoring; contact between operator and moving robot is prevented by the robot.
— Power and force limiting; contact forces between operator and robot are technically limited to a safe level.

The industrial HRC installations are limited by the current safety legislation. The existing standards provide rather guidance than clear definitions and do not clarify what sufficient safety for the operator in collaborative operations is. They are loosely defined and complex. A high inter dependency exists, meaning the standards build up on each other so that it takes longer to look up the related definitions from previous standards. Further, different use case scenarios are not specified, which makes the current standards too general and allows for multiple interpretation possibilities. As a result, companies cannot follow defined guidelines and depending on the company's internal safety level they interpret the recommendations in ISO/TS 15066 differently.

Safety schemes in HRC can be defined as pre-collision and post-collision systems [82][76]. These terms are directly related to the safety strategies collision avoidance and impact force limitation.

— Pre-collision systems: Collisions are prevented by stopping the robot motion, altering

its trajectory, alarming the operator with an audio output and moving back from the operator. Workspace monitoring systems detect the position of the human with external safety sensors.

— Post-collision systems: In case of an unexpected or unavoidable collision the harm to the human is minimized. This can be achieved with integrated joint torque sensors in the robot's internal safety control systems for collision detection which initiate a robot stop and mechanical compliant systems such as lightweight robot structures and robot skins which limit the maximum transmittable energy to the operator during an impact.

The disadvantages of post-collision systems are 1) incorrect defined limits might result in serious accidents, especially in combination with sharp tools or parts 2) frequent production stops in case of collision occurrence. The advantages of a pre-collision system are 1) visual or acoustic signals can warn the operator of a coming robot stop 2) a speed reduction can delay the time point of a motion stop. However, external sensors require more space around the robot because of the sensor detection time.

Contacts between robots and humans

In this thesis, for HRC we want to have contact with the human. So the robot and human have to work in the same workspace with complex fence boundaries (as the interactive surface is complex). From the classification discussed in previous section we can say the goal of this new system is to have a hybrid of the existing classification. We establish a new safety constrains taking into the ISO/TS 15066, requirements of having at least one or more safety methods. In this process the robot should avoid unwanted collisions with the user. In this context we have two types of contacts: (i) task contacts and (ii) unwanted contacts. The goal is to have a the set safety methods in these 2 scenarios. The implemented safety methods are as follows:

1. Unwanted contacts are collisions that happen when the robot end-effector or robot body comes in contact with the user while it is not expected in the task description. Multiple safety standards are implemented, explained as below:
   — The idea is to have more free space (space outside the reach of user workspace), possibly to move the robot at higher speeds in these non-contact space;
   — To have limited robot speed depending on user workspace boundary, robot, tool and workspace;
   — To have a fast and reliable collision recognition.
2. Task contacts are contacts that are related with the actual task of the robot and therefore they are expected and mainly initiated by the robot. They are located on the frontiers of the virtual environment and occurs between the end-effector and the human's hand.
   — To have low robot speeds in areas where contact is possible;
   — To limited contact force depending on region and shape of robot, tool and workspace;
   — Making sure the velocity and force limits do not exceed.

Speed and force limiting must be implemented. Without risk assessment, human-robot collaboration cannot take place. The overall application must always be considered, i.e. not only the robot. Safety functions must be implemented in accordance to determined requirements. The initial step towards safety is based on the location of the robot. That has to be far during the no contact phase and be able to reach th humane during the static contact. The process on how we achieve this discussed in this chapter.

To respect safety constraints and limit the risk of injury when the robot effector is in contact with the operator, the positioning of the robot must be optimized. This placement should also

take into account the size of the environment in which the operator has to interact in immersion. Constraints will also come from the desired haptic rendering. Indeed, during interactions, the properties (crossing singularity, self collision and precision of movement) of the robot must be adapted to the interaction modes required. The interface placement will be fixed for a given experiment if the size of the environment to be simulated is smaller than the cobot workspace. So the initial objective is to find a placement of the robot that allows to reach all the desired zone of interactions. If this is not possible, then to constrain the robot to limited workspace of the robot, it is possible to considered several placements adapted to the task.

2.1.3 Methodology to achieve the safety standards in HRC

Before beginning any robot task, users must position the robot's base, a task that was performed on user intuition (in the beginning of Lobbybot Project). While slight perturbation is tolerable for robots with movable bases, correcting the problem is imperative for fixed-base robots, especially if some essential task sections are out of reach. For non-mobile manipulation robots, it is necessary to decide on a specific base position before beginning manipulation tasks. The manual work of identifying potential base location is defined effectively, so that, where the position of the robot and workspace ensure the maximum amount of work performed, where the maximum amount of workable area can be reached. Since complicated robotics tasks require the base to be placed at unique poses based on task demand and also to consider time efficiency and range of applicability.

Collision in the base placement planning is vital, as the output base location may be in collision with manipulated workspace, worktable, and other surroundings. Also, the base placement does not depend only on the reachability of the task poses, the cost or minimum joint motions should also be considered.

To manage these collisions, we come up on a strategy for safety of user in ICI scenario. We consider multiple control factors like planning, workspace /working posture and placement.

Planning: There are two aspects of planning of robot motion in this scenario: (i) reaching the desired next ICI location safely, (ii) being able to track the user hand at the frontier of the virtual environment without having collision with the user, to prepare a future motion of the ICI to its desired location as soon as possible. Having a better planning control helps in avoiding collisions.

Workspace / Working posture: Constraining the robot's working posture for the entire workspace reduces the risk of collisions. In practice this avoid reconfiguration of the robot to change of working posture (such as elbow up or elbow down configuration) and such large amplitude motion can lead to collision.

Placement: The placement of the robot with respect to the user and the environment has to be defined in order that the robot is able to carry the prop to the limit of the virtual environment that the user wants to explore while avoiding collision with the human. It is better if all the part of the robot is keep outside of the interior of the virtual car.

Find a convenient location for the robot is an not easy task and it is the main goal of this chapter.

2.2 Robot configurations

For the UR-5 Workspace there are eight configurations that form a complete workspace. The computation of these configurations is based on the number of inverse kinematics solutions

obtained for a given point of end-effector.

Definition: *Configuration Space*: All the possible values of θ_i it can take. For a six-axis robot is the set of all θ_i attainable with out collision.

Definition: *Workspace*: The set of points that can be reached by its end-effector without having any collision. The physical volume it can cover.

2.2.1 Inverse kinematics

In this section we will start with an analytical model that give all the solution of the inverse kinematic model. The analytic inverse kinematics problem is to find the set of joint configurations that satisfies desired end-effector position and orientation. Given the input of the end-effector position and orientation, we get the different set of possible joint configurations to reach the same end-effector position and orientation (For UR-5, 8 possible set of joint configurations). A detailed explanation of the UR kinematics and inverse kinematics can be see in Appendix A. In this section we will only deal with the key final results to understand the classification of the configurations.

From inverse kinematics, for finding θ_1 results in:

$$\theta_1 = \text{atan2}({}^0p_{05y}, {}^0p_{05x}) \pm \cos^{-1}(\frac{d_4}{R} + \pi/2) \tag{2.1}$$

There exist two solutions for θ_1, where the shoulder is left or right. Using the function atan2 is essential for insuring correct signs and behavior when ${}^0p_{05y} = 0$.

Given a particular θ_1, we can solve for θ_5. The final equation for θ_5 is:

$$\theta_5 = \pm \cos^{-1} \frac{p_x s_1 - p_y c_1 - d_4}{d_6} \tag{2.2}$$

Again, there are two solutions for θ_5, which correspond to configurations where the wrist is in/down or out/up. After computing θ_5, θ_6 can be computed based on the equation below:

$$\theta_6 = atan2(\frac{y_y c_1 - y_x s_1}{s_5}, \frac{x_x c_1 - x_y s_1}{s_5}) \tag{2.3}$$

The other three joints can be derived easily, considering that they act as a 3-RRR planar arm. Once the previous 3 joints found, the location of the base and end-effector of this 3-RRR arm is available, then these 3 joints can be solved. There is two possible configurations, elbow up or elbow down. No solutions exist when the distance to the 4th joint exceeds the sum $|a_2 + a_3|$ or is less than the difference $|a_2 + a_3|$. If $a_2 = a_3$, a singularity exists when $\theta_3 = \pi$, making θ_2 arbitrary.

$$\theta_2 = \alpha_1 - \alpha_2 = \arctan(-{}^1p_{14z}, -{}^1p_{14x}) - \arcsin\left(\frac{-a_3 \sin\alpha_3}{|{}^1p_{14xz}|}\right) \tag{2.4}$$

$$\theta_3 = \pm \arccos \frac{|{}^1p_{4xz}|^2 - {a_2}^2 - {a_3}^2}{2a_2a_3} \tag{2.5}$$

$$\theta_4 = \arctan({}^3x_{4y}, {}^3x_{4x}) \tag{2.6}$$

Based on the IK computation, we can summaries a total of 8 solutions exist in general for the general inverse kinematic problem of the UR-5: $2\theta_1 \times 2\theta_5 \times \theta_6 \times 2\theta_3 \times \theta_2 \times \theta_4$. The different configurations of robot as shown in Table 2.2. Figure 2.8 shows the visualization of the different

Table 2.2 – Different configurations for UR-5

Configs	q_1	q_{234}	q_5
Config0	Left	Up	Out
Config1	Left	Down	Out
Config2	Left	Down	In
Config3	Left	Up	In
Config4	Right	Up	Out
Config5	Right	Down	Out
Config6	Right	Down	In
Config7	Right	Up	In

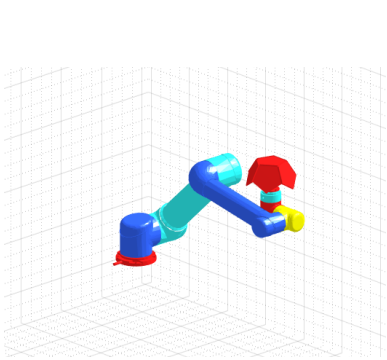
configurations based for a single end-effector position and orientation. It can be seen that to have maximum safety for the user, it is better to avoid some configurations like 0, 1, 4 and 5 as these configuration have the part of the robot (yellow component) closer to the human than in other configurations. Also configuration 1, 2, 5 and 6 are not desirable as they have a elbow down configuration which is prone to collision with the robot platform and human. From Table 2.2 and Figure 2.8 it can be summarized that Configuration 3 and 7 are ideal, with key difference being orientation of θ_1 (left/right). The selection of configuration is very important because the maximum reachability of the workspace is crucial to avoid changing configurations between execution of task. A detailed study on the workspaces generation based on configurations and workspace reachable spheres will be dealt in further sections.

2.3 Generate workspace

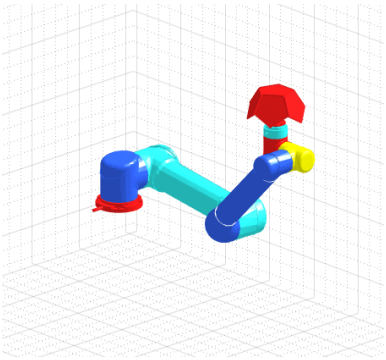
The next step is to construct the workspace of the robot. Further we constraint the workspace to one configuration and analyze them. The construction of the workspace is done taking into consideration both the base table and prop of the robot. Adding this information in universal robot description file (URDF). URDF is an XML file format in ROS to describes all elements of the robot. First we construct a workspace with a single fixed configuration (eg: Config 7). From the URDF we have the information of the dimensions of the robot, creating a rough spherical workspace W with those dimensions (Algorithm 1). Now, the workspace W is discretized with squares of size of 5 cm (Figure 2.9a).

Creating squares of 5 cm and creating a sphere at its center (Figure 2.9b). Now for each sphere, we have a set of poses that are to be checked. By poses, we mean different orientations to reach the same sphere. These poses are shown using the arrows in Figure 2.9c. A range of poses is considered, with an interval of 30°. The range of Azimuth is from [0,360°] and elevation of [0,180°]. So for every pose of the sphere, we find the IK solutions. If a solution exists for a given pose then the sphere is reachable. But for the present case, we want a specific solution to exist, as we want to construct the workspace for a desired configuration defined among config 0 to config 7, illustrated in Figure 2.8. So we check if the solution we get is of the desired configuration, then we store the sphere s_i and pose p_i, and next, we compute the reachability index (RI). This is a factor that determines how many poses are reachable in a given sphere with the desired configuration. Using this RI value, we color code the whole workspace.

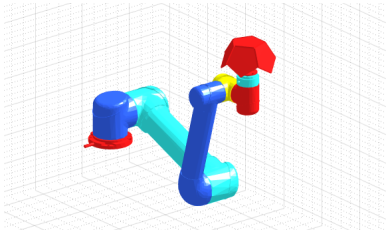
$$RI = \frac{\text{(No. of poses with solution)}}{\text{(No. of total poses checked)}} \times 100 \tag{2.7}$$



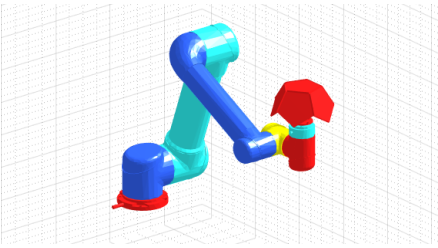
(a) Configuration 0



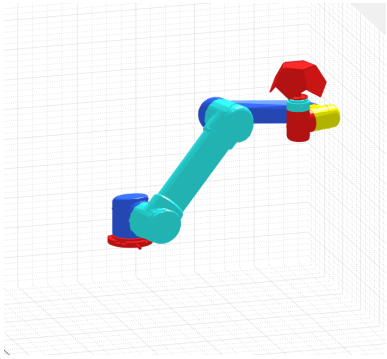
(b) Configuration 1



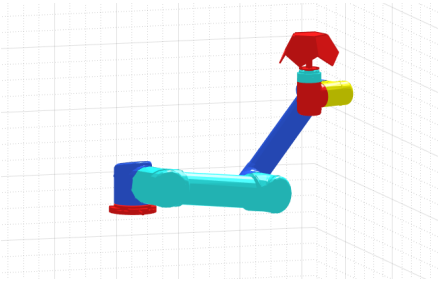
(c) Configuration 2



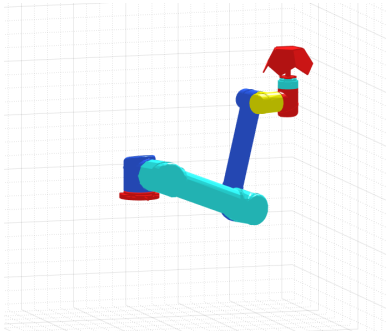
(d) Configuration 3



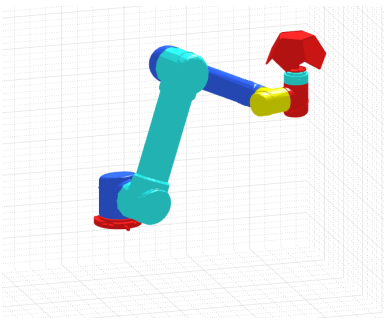
(e) Configuration 4



(f) Configuration 5

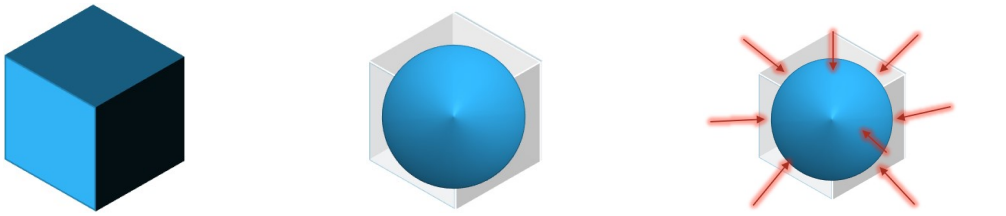


(g) Configuration 6



(h) Configuration 7

Figure 2.8 – All the configurations of the robot for same end-effector position and orientations.



(a) V_i of workspace W (b) sphere s_i for V_i (c) poses P for sphere s_i in V_i

Figure 2.9 – Discretization of the workspace into voxels.

Algorithm 1 Construction of workspace

Input: URDF of Robot, discretization size - n
Output: create hypothesized workspace Z.

```
V_i - Discretization of W, with size n
for each V_i in W do
    create spheres s_i for V_i
    check for self collision
    save s_i in S
end for
for each s_i in W do
    Generate poses P
    for each p_i in P do
        Find ik solution
        if solution then
            store (s_i, p_j)
        end if
        compute ri = reachable poses / total number of poses
        store ri with (s_i, p_j)
    end for
end for
```

Table 2.3 – Color code for reachable sphere in workspace

Color	Percentage
Blue	91-100 %
Light Blue	51-90 %
Green	31-50 %
Yellow	6-30 %
Red	1-5 %

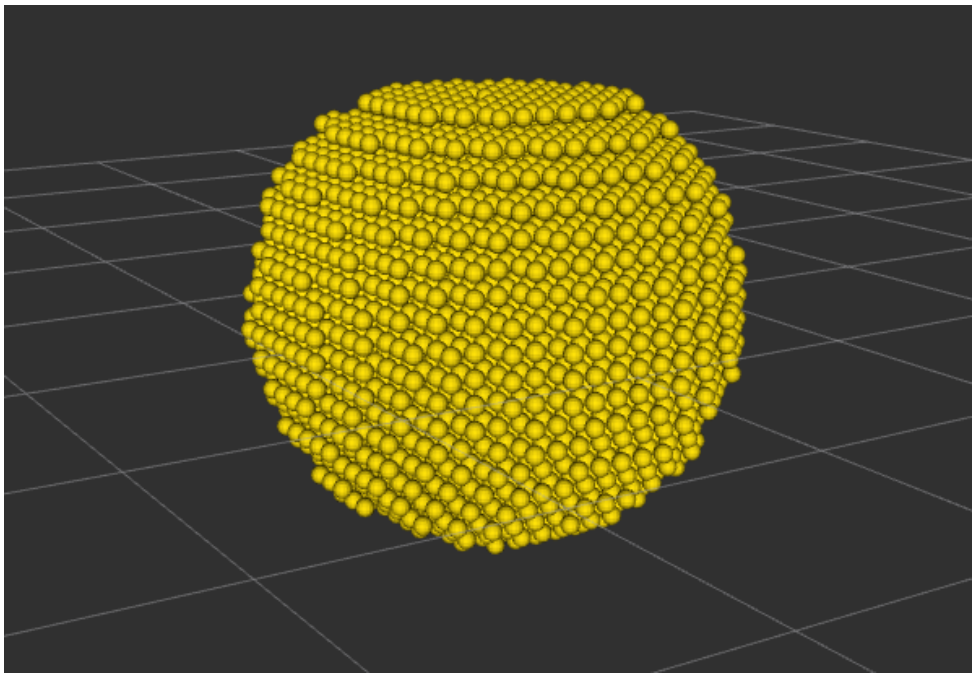


Figure 2.10 – Construction of complete workspace of UR-5.

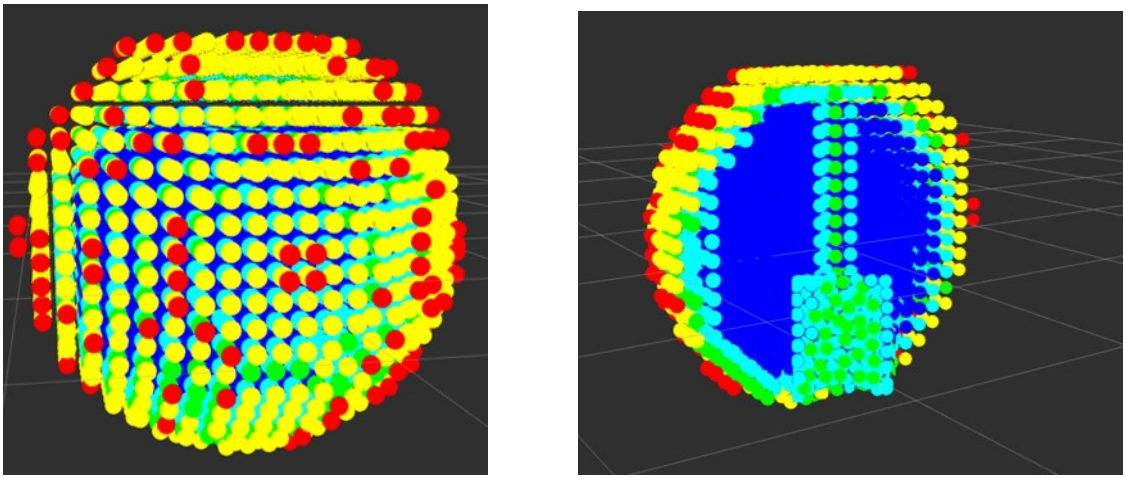
A workspace construction comprising of all the configurations was done (Figure 2.10). Further workspaces were constructed for each configuration (from config 0 to config 7). These configurations and the corresponding reachable spheres in the workspace is summarized in Table 2.4.

Figure 2.11 shows the workspace generated for Config7. The full workspace, and the dissected one. We consider the prop and the base table as our robot collision objects. A comparison of workspace for different configurations (in Table 2.4) shows that not many spheres are lost when selecting one specific configuration (Config7). The total number of spheres in Figure 2.10 are 4584, and that in Figure 2.11 is 4519. The color code (Table 2.3, in this case of configuration 7 is based in the reachability index.

So after the workspace has been constructed for a specific configuration the next step is to select the task points and find the base placement based on these task point. This will be dealt in the next section.

Table 2.4 – Reachable spheres in workspace.

	Reachable Spheres
Whole Workspace	**4584**
Config0	3311
Config1	3311
Config2	4484
Config3	4484
Config4	3310
Config5	3310
Config6	**4519**
Config7	**4519**



(a) Complete workspace for config 7

(b) Half sliced Workspace for config 7

Figure 2.11 – Generated workspace for Config7, considering the prop and base table.

2.4 Regions of interest

To interact between the virtual environment and robot tool, we can specify two fixed orientations for the TCP. These orientations under test are obtained by considering three major regions in the interior of the car, for interaction. These regions are define as S_1, S_2 and S_3. The reason for selecting these regions for computing the base location is, that they are the extreme regions in the virtual environment: the door, the dashboard, and the seat next to the user/driver.

Also, we will be discussing the surfaces of the prop that will be used for these tests. The prop that we consider has 6 surfaces. Figure 2.12 shows the major frames of interest in this test scenario. The TCP frame is the frame at the end of the robot without any attachment. We will need to consider two transforms from the TCP to the surface of the prop in contact. These 2 surfaces are Surface 5 and Surface 6 shown in Figure 2.12. Since joint q_6 of robot (UR-5) has [-2π,2π] limits, all five sides (Surface 5) of the prop are reachable by simple rotation. So when one surface is reachable, all the rest side surfaces of the prop are also reachable. So the problem is simplified and only one surface (Surface 5) is used.

So, transforming from TCP to Surface 6 is a simple translation in Z direction,

$$T = TransZ[15\ cm] \tag{2.8}$$

Transform from TCP to Surface 5 is a translation in Z, followed by rotation in Y and a final translation in Z.

$$T = TransZ[2.87\ cm].RotY[63.44°].TransZ[7.75\ cm] \tag{2.9}$$

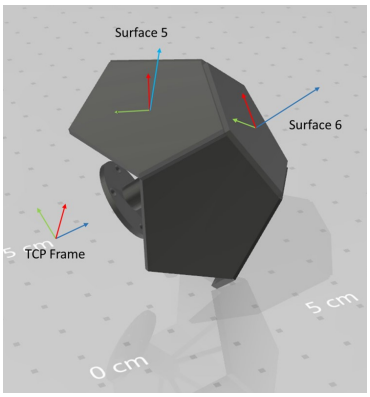
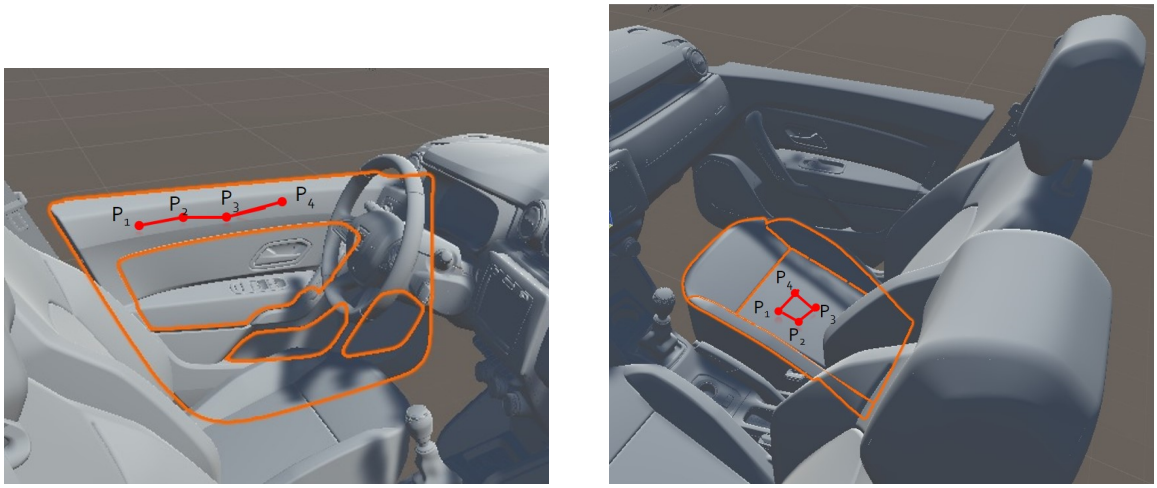


Figure 2.12 – The prop used in the experiment.

For S_1 region as shown in Figure 2.13a, which is the door close to the user, we want for the robot being able to reach this surface with the prop surface 6 orientations for the given set of points P_1 ... P_4. This correspond to the orientation 1. The points must belong to the workspace W_1 shown in Figure 2.15a.

For the S_2 Region as shown in Figure 2.14, we can have a maximum surface inter-action with robot prop, we imposed that the surface 6 or surface 1 to 5 of the prop can be presented in order to test several material. A rotation of the last axis of the robot allows to change the surface of the prop from surface 1 to 5. To achieve this, the points has to be reachable by the two orientations. The points must belong to the workspace W_3 shown in Figure 2.16.

Finally, when considering S_3 region, the seat next to the user/driver, as shown in Figure 2.13b all the 5 surfaces of the prop were tested. So the orientation selected is Surface 5. In all these



(a) S_1 - Region (b) S_3 - Region

Figure 2.13 – Different regions in the virtual environment.

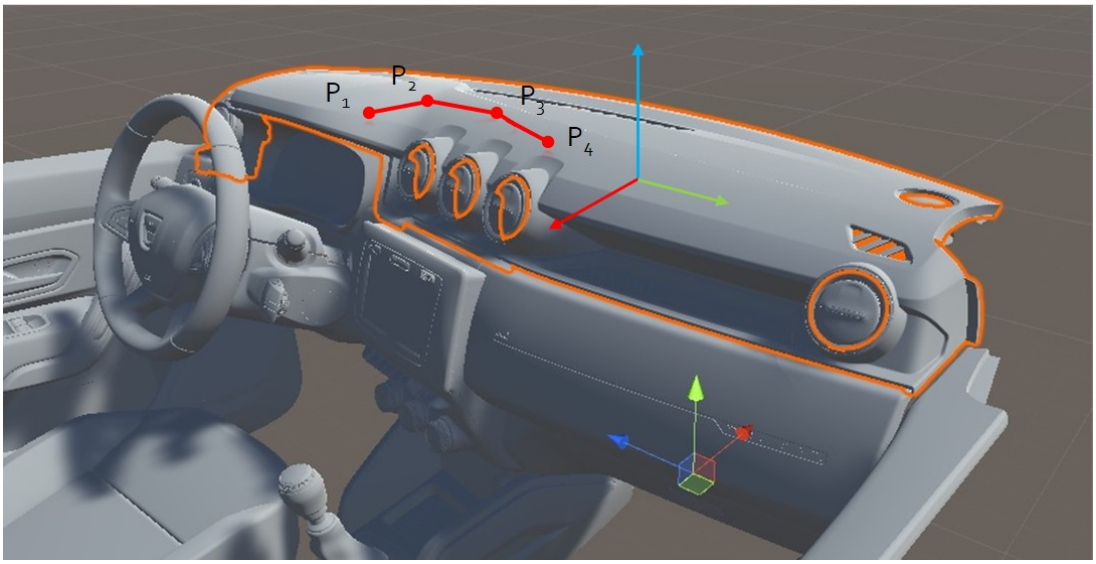
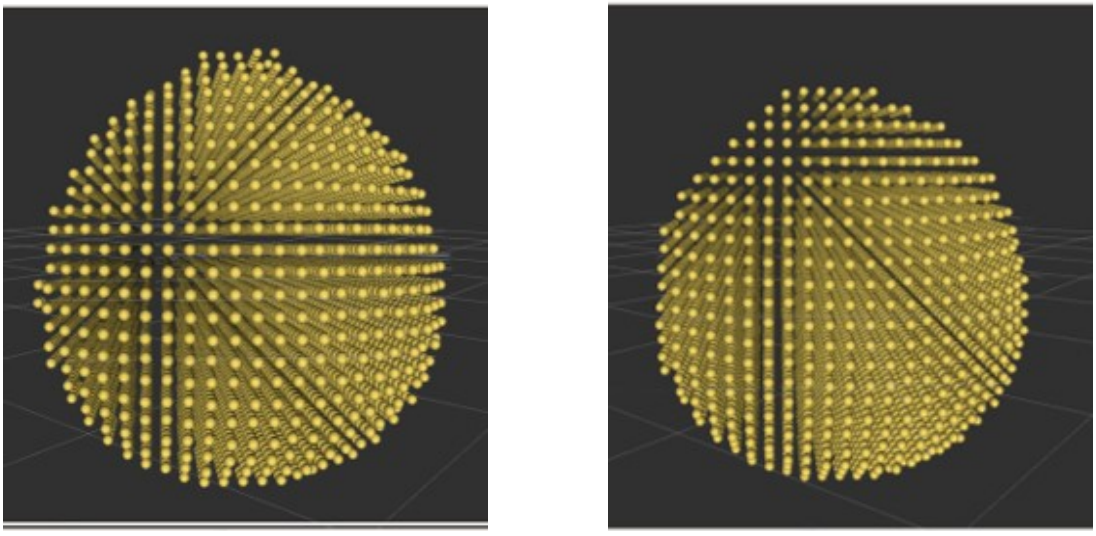


Figure 2.14 – S_2 - Region



(a) Workspace build for orientation 1 (denoted W_1).(b) Workspace build for orientation 2 (denoted W_2).

Figure 2.15 – Workspaces with S_1 and S_3 region orientations.

3 regions S_1, S_2 and S_3 we define 4 points P_1... P_4 that we want the robot to reach in each region. For this task, the points must belong to workspace W_2 shown in Figure 2.15b.

2.4.1 Workspace for given orientation

Now, as the region of interest for robot is set up and the workspace of the robot is created. The next step is to create workspace of the robot based on the fixed orientations for the robot.

The workspace describes the reachability of a given robot model by discretizing its environment, creating poses in the environment, and calculating valid IK solutions for the poses. The poses which are reachable by the robot are associated with discretized spheres. The reachability of each sphere in the environment is parameterized by a Reachability index.

Workspace is generated for a given set of orientations and a given configuration of the robot. The output workspace map will be stored all the sphere the robot can reach. Figure 2.15a and Figure 2.15b show the created workspace for the selected orientations (Surface 5 and Surface 6) using the configuration 7 of the robot.

Then, we generate the intersection of both workspaces that can reach both orientations. Figure 2.16 show the region that is reachable with both the orientations of the prop (all six surfaces of the prop). This intersection will be used in our further tests for the base placement of the Robot.

2.5 Base placement of the robot

2.5.1 Requirement

The base placement of the robot, is defined in order to be able to place the prop at the point that the human want to explore. Since the robot arm studied has dimension which are close to the dimension of human arm, the ideal robot location would be in the location where the user is seated. However to avoid collisions, we cannot place the robot there. In fact, we would like

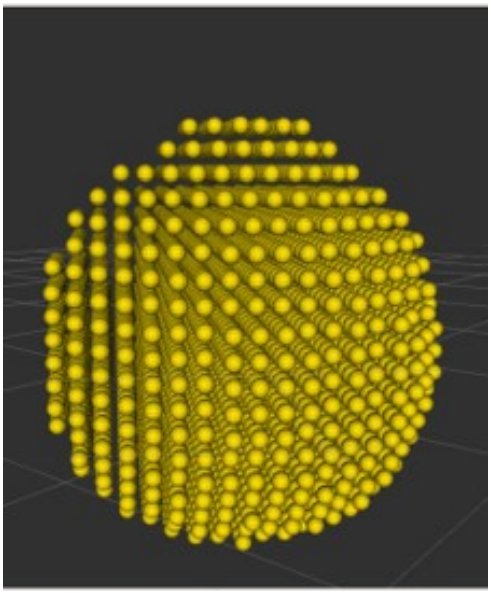


Figure 2.16 – Workspace for both orientation by intersection of W_1 and W_2 to obtain W_3.

the robot to be located outside the passenger compartment of the virtual car. It is expected that the user body (specially the chest and neck) should be outside the workspace of the robot, such that there will be no possibility of contact between body parts of the user and the robot's arm. The robot placement should be such that it can reach all the desired contact regions in the required orientation.

2.5.2 Methodology

The cobot used is UR-5 from Universal Robots [99]. It is a 6 DOF robot able to interact with humans in a shared space or to work safely in close proximity. We constrain the posture of the robot to stay in a single aspect, i.e. a particular configuration. In this case, it is elbow up. This helps us in avoiding crossing singularities, during motions.

Now to find the best base placement of the robot, we discretize a set of all the possible base locations that can reach a given point. The set of possible location are expressed as a set of sphere of 8 cm of diameter that represent the paving of the space.

If the robot is fixed on the floor, the paving can include only sphere at the level of the floor. In more general case, as the one studied here, a 3D paving can be considered. The vertical coordinate of the best pose of the robot will defined the good height for the table supporting the robot.

The task is defined as previously by the set of points that we want to reach with a given orientation. The workspace generated for each orientation of region is also used. A point can be reach if it belongs to the workspace associated to the desired orientation (for example W_1).

The methodology can include any number of task points. For each point associated with one orientation i.e. one workspace W_i, all the sphere inside of the workspace is covered and placed at the point studied, and the corresponding pose of the robot is registered. The location of the based is associated to the sphere paving the possible placement of the robot. The number of point reachable from this base location is increased.

After considering all the point of interest (12 point in our study), the possible placement of

the robot to achieve the task is defined as the sphere where the number of reachable is maximum.

From this set of base locations, we remove the base locations that coincide with the location of the user. Depending on the number of desired points that can be reached, the points are categorized. This gives the possible robot base location more suited for this scenario.

The algorithm developed is based on the library Reuleaux [85] available via ROS and used the tool [70, 39].

Algorithm 2 Base placement of the robot

Input: Reachability Map, Task Points.
Output: Set of robot base locations.
function PROCEDURE 1(create basemap)
for each task point P_i **do**
INVtransform from P_i and obtain b the base location of the robot
cluster b and assign the spheres S_k
increment the index associated to sphere S_k
save S_k
end for
end function
function PROCEDURE 2(base placement)
access spheres S_k
find spheres with max base index
exclude the spheres from inside the car
end function

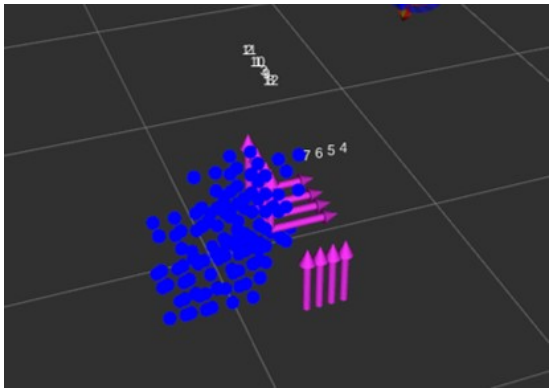
2.5.3 Results

Tests were performed to find the base placement for a given set of points the robot has to reach with a defined orientation. Results were found in robot base placement locations, outside the interior of the VR car. Different combinations of the regions were taken to analyze how the base location of the robot changes.

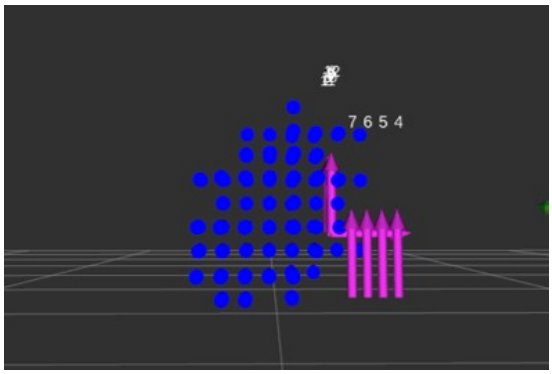
In the tested scenario, the dimension of the table under consideration is 75 cm high and 53 cm long and wide. The base of the robot is located in the middle of the table to ensure its stability. A total of 12 task points have been defined: four task points in S_3 region (Chair), four task points in S_1 region (Door), and four task points in S_2 region (Dash board) with two sets of different orientation. The possible robot base location are illustrated by blue spheres. In all the test cases the base location are outside the passenger compartment of car. Multiple robot base locations were found that could reach all the desired positions.

S_1 and S_2 Region

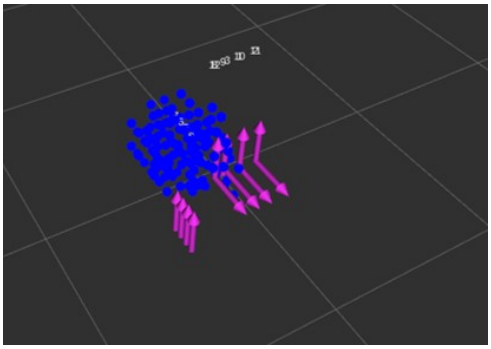
A total of 8 task points have been defined: 4 task points in S_1 region (Door), 4 task points in S_2 region (Dash Board), with 2 sets of different orientations. All six surfaces of the prop were tested to be reachable. It can be seen that the base location is not inside the interior of the car. This analysis also helps in deciding the base table dimensions which will be discussed in further sections.



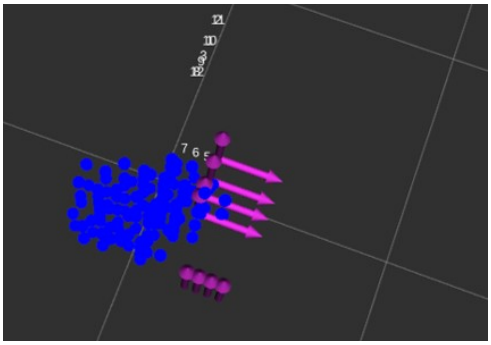
(a) Base Placement for regions S_1 and S_2.



(b) Side View.

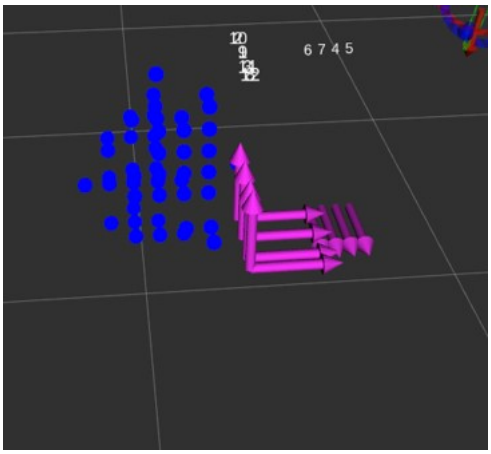


(c) Front View.

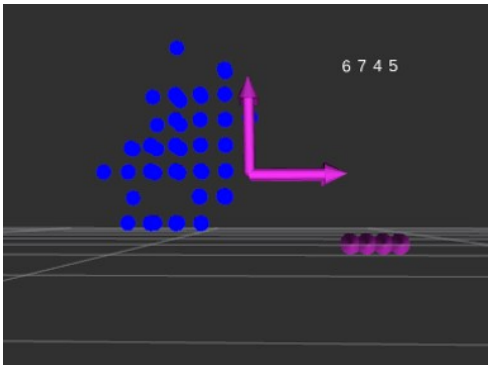


(d) Top View.

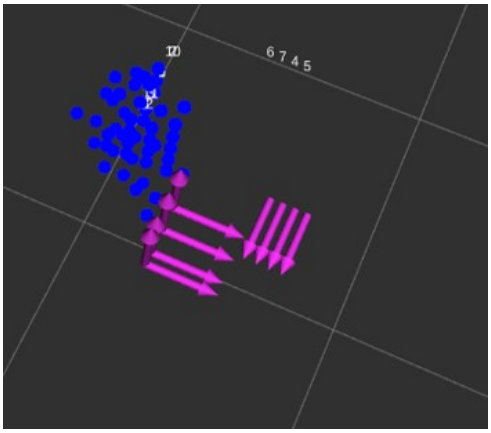
Figure 2.17 – S_1 and S_2 region task points and base location.



(a) Base Placement for regions S_2 and S_3.



(b) Side View.



(c) Top View.

Figure 2.18 – S_2 and S_3 region task points and base location.

S_2 and S_3 Region

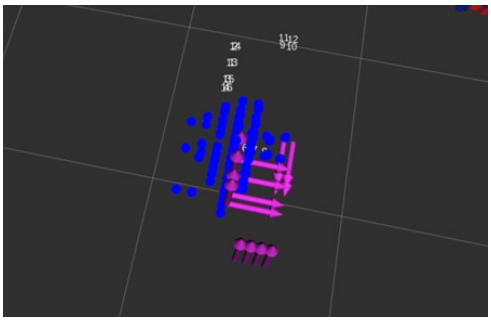
A total of 12 task points have been defined: 4 task points in S_3 region (Chair), 4 task points in S_2 region (Dash Board), with 2 sets of different orientations. All six surfaces of the prop were tested to be reachable. It can be seen that the base location is not inside the interior of the car. each sphere here is at a discretization of 8cm.

However when compared to the previous result the position of the base to more towards the right (Figure 2.18), unlike it was left in the previous case (Figure 2.17).

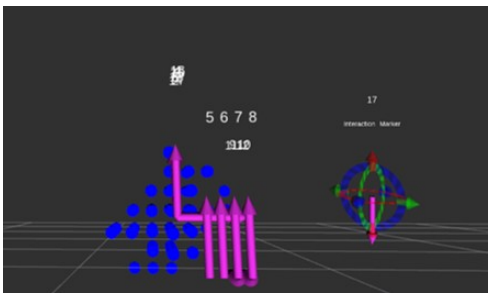
S_1 S_2 and S_3 Region

A total of 12 task points have been defined with 4 task points in S_3 region (Chair), four task points in S_1 region (Door), and four task points in S_2 region (Dash Board) with two sets of different orientations. All six surfaces of the prop were tested to be reachable. It can be seen that the base location is not inside the interior of the car, also the number of base locations, when compared with Figure 2.17 and Figure 2.18, is reduced.

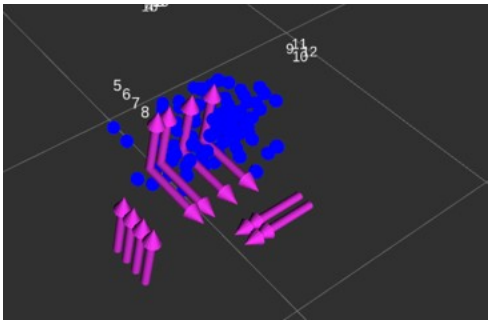
Unlike the previous results, the base location is a bit closer to the user. The maximum distance of the robot from the car is not ideal for the given base dimension of the table.



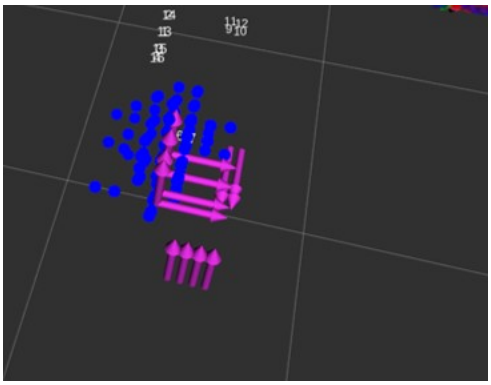
(a) Base Placement for regions S_1, S_2 and S_3.



(b) Side View.



(c) Front View.



(d) Top View.

Figure 2.19 – S_1, S_2 and S_3 region task points and base locations.

2.6 Obstacle avoidance and dimensions of base table

In the previous study, the obstacle of the environment (the chair and the human) was not taken into account since the location of the robot was not known. Thus the next step is to check for the selected pose location (in Figure 2.19a) that the points can be reached when the real environment are taken into account.

In this environment, we introduce a model of the chair and the human in rest position. For the user's body, the dimensions considered, for user cover 75 percentile of men and women of age group 25-55 in France. The volume dimensions of the cuboid are 185 cm in height and 60 cm in width and 30 cm in thickness (Figure 2.20c).

First the location of the robot corresponding to the actual height of the table is considered. For each pose location, the following task is considered: go from home position to each points successively.

From several candidate poses of the robot, the task cannot be achieved. For others, the task can be achieved. Two main problems were encountered:

An illustration of the result is shown for the case illustrated in Figure 2.20a. The configuration of the robot to reach a point on the passenger chair is shown in Figure 2.20c, the robot is close to the legs of the human. The configuration of the robot to reach a point on the dashboard is shown in Figure 2.20d, the robot configuration is convenient, it is far from the human.

To improve, the robot base placement, we can change the height of the table used. Several tests were done to determine the effect of changing the height of the table on the position of the base and its effect on the inclusion of the robot in the passenger compartment of the car. For

performing the tests, the task points shown in Figure 2.21a are considered.

When the robot tries to reach the chair task points, we can see from Figure 2.20c that the robot goes to the passenger compartment of the car. Conversely, the same problem does not occur in Figure 2.21c, as the height to the table is increased by 5 cm. Similar result is shown in Figure 2.21d with modified base table dimensions. It can be concluded that the current size of the table is not ideal for the scenario, and that a better design of the table can be done to improve the results.

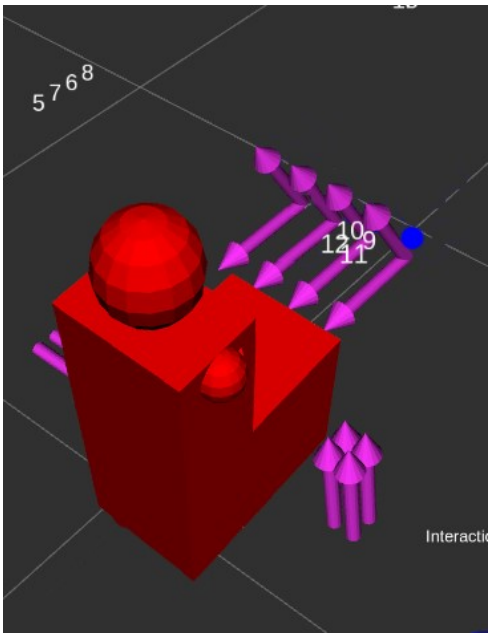
It can be concluded that it is better to use a table of 80 cm height than 75 cm height for the given scenario.

2.7 Conclusions

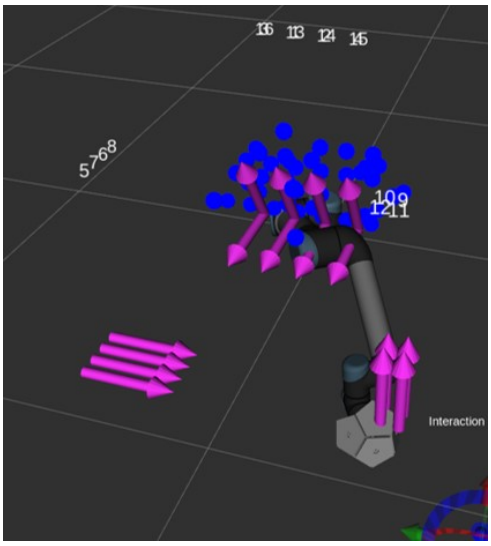
Complete modeling of a well-known robotic manipulator UR5 was presented. For this purpose, first the robot's kinematic was characterized and calculated. Workspace and different configurations of the Robot were analyzed and visualized. The selection of configuration is very important because the maximum reachability of the workspace is crucial to avoid changing configurations between the execution of tasks.

The work presented allows to choose a robot placement(Figure 2.22) so that the end-effector reaches the areas of interest in the environment while ensuring the absence of collision between the robot and the user. An algorithm has been proposed define convenient placement of the robot base in order to reach several areas of interest. First results are obtained using a given height of the table that support the robot base. The results show an accessibility of the task but also show a robot insertion in the passenger compartment. Changing this height setting prevents this intrusion and therefore improves user safety.

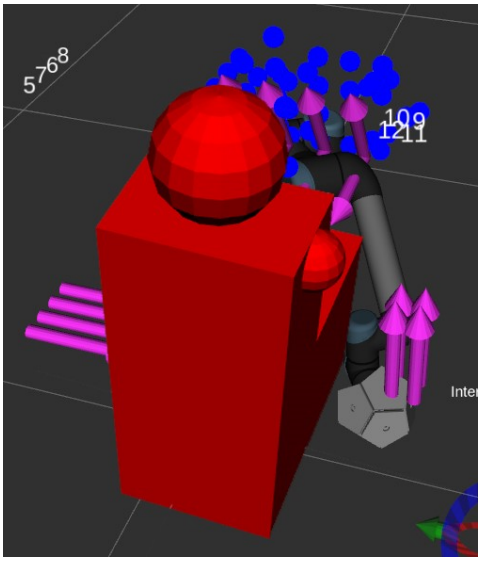
Based on the study performed the following results were found. The ideal location of the robot is 80 cm above ground level (Figure 2.23).



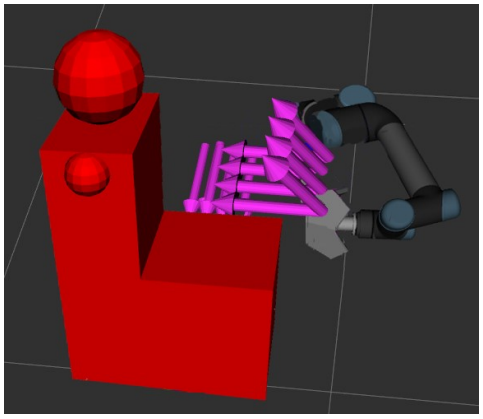
(a) Robot base locations in blue and task points with desired orientations in pink



(b) Robot interaction at chair without user model

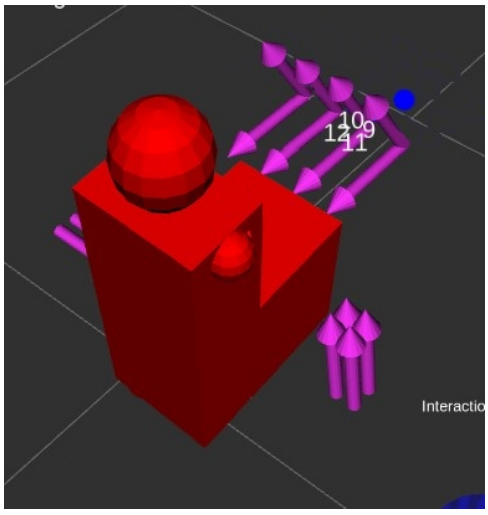


(c) Robot interaction at chair with user model

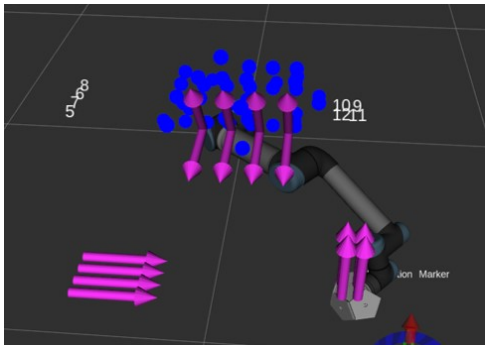


(d) Robot interaction at dash with user model

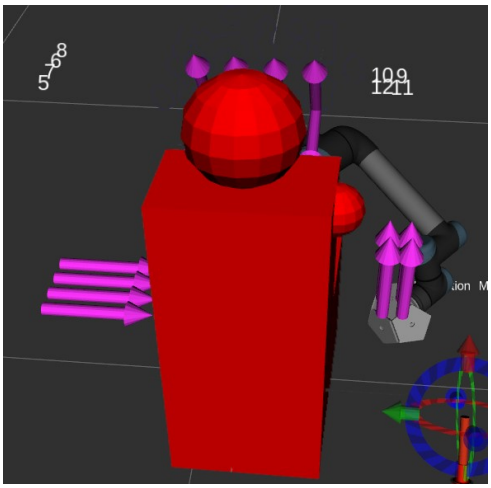
Figure 2.20 – Introduction of User model into environment - Table height of 75 cm.



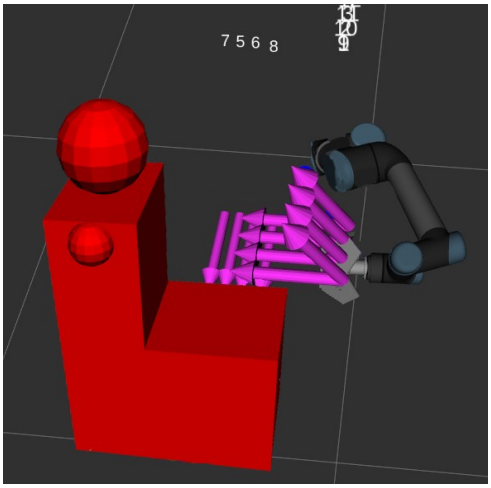
(a) New table height in Blue and task points in pink



(b) Robot interaction at chair without user model



(c) Robot interaction at chair with user model



(d) Robot interaction at dash with user model

Figure 2.21 – Analysis on the dimension of the base table of the robot (Table Height 80 cm).

(a) The computed robot placement

(b) Robot interaction with user

Figure 2.22 – Placement analysis of the base table of the robot.

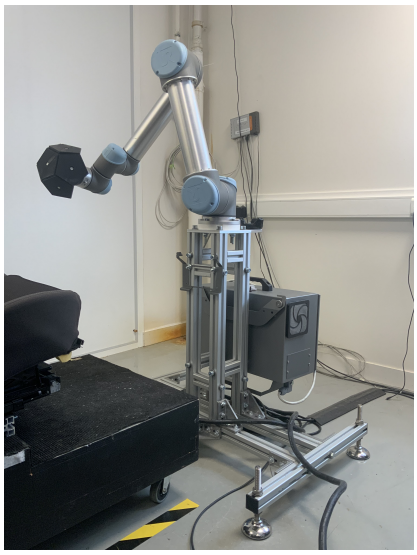


Figure 2.23 – New Table design, 80 cm height.

Chapter 3

Design of a user mannequin model for ROS

3.1 Introduction

In this chapter we will introduce the design of the user model used in this thesis. The main objective of this approach is to ensure the safety of the operator. Collecting knowledge of the scene and creating a model of the human. Alongside we also want to use the information to predict the goal state of the robot trajectory. The goal state can be predicted via the final place of the human hand knowing its current pose.

This can be analyzed by observing the movement of a human through a motion capture system. A motion capture system is a system able to track the motion of an object moving in space. The purpose of motion capture is to record only the movements of the actor, not his or her visual appearance. This animation data is mapped to a 3D model so that the model performs the same actions as the actor. In our case, the motion of interest will be that of the different parts of the body of the subject. We will use a motion capture system based on rigid bodies.

The scene (Figure 3.1) is composed of the chair of the car which is fixed, the support of the robot is fixed and known. The robot movement is known. The human is seating in the chair (torso and legs fixed), its arms are moving.

This system will provide us with the position of each of the parts of the body of the subject at different instants. The whole process of analyzing the motion of a human subject can be divided into three different steps. The first step consists of building a model of the subject. In order to do this, we need to get different measurements from the subject, which will allow us to characterize the Human model.

The second step consists to estimate the necessary joint values in order to achieve the observed movement. Finally, the third step consists of visualizing them. We want to capture or track the motion of the user's hand in the given environment. To achieve this, we use the VIVE trackers and the HTC VR set.

3.1.1 Using kinect to know the environment

Using Kinect is useful for detecting any unexpected objects in the environment. It helps to get a rough idea of the objects in the environment. However, the location of the camera is difficult to cover the entire scene and also Kinect's constructed information has no information

Figure 3.1 – Environment scene setup.

about an object obstructed by another object.

As shown in Figure 3.2, the information on the wall behind the objects is not known. However, this information can be used to know the form information of any object in the environment. This knowledge can be used for emergency stops if the robot is interrupted by an obstacle. These shortcomings in using Kinect has prompted to design a new approach to know the user presence in the environment.

3.2 Selection of human model

A human can be represented as an articulated rigid body. The arm can be a classical 7 dof arm. The pose of specific bodies can be measured via sensors (Vive tracker). In the studied scenario, it is assumed that the user is seated in the chair (the motion of the trunk is limited, the leg is assumed fixed). His/her hand motion is of specific interest.

In this section, we will describe the process of building an approximated model of the subject.

The model that has been chosen to describe the body of the subject is based on the modified Hanavan model (Figure 3.3b). The proposed model is a quite simple enough for the purpose of this project. This model approximates the human body by a set of 6 different rigid body segments, as shown in Figure 3.4.

These segments are defined as simple geometric shapes. This way, this model allows us to describe the body of a subject. For this test case, we want to measure only one dimension and relate the rest of the dimensions based on it. For this, we use Winter's model shown in (Figure 3.3a). This relation helps in measuring only the height and relating all other dimensions based on it.

From the height (measured using the HMD of the Vive set), the dimensions that define each of the segments can be directly obtained as a relation to height (Figure 3.3a). Then, from the measurements of the total height of the subject, the length of each of the segments can be estimated through those relations. This feature help in having a model that can be scaled based on the user's height (Figure 3.12).

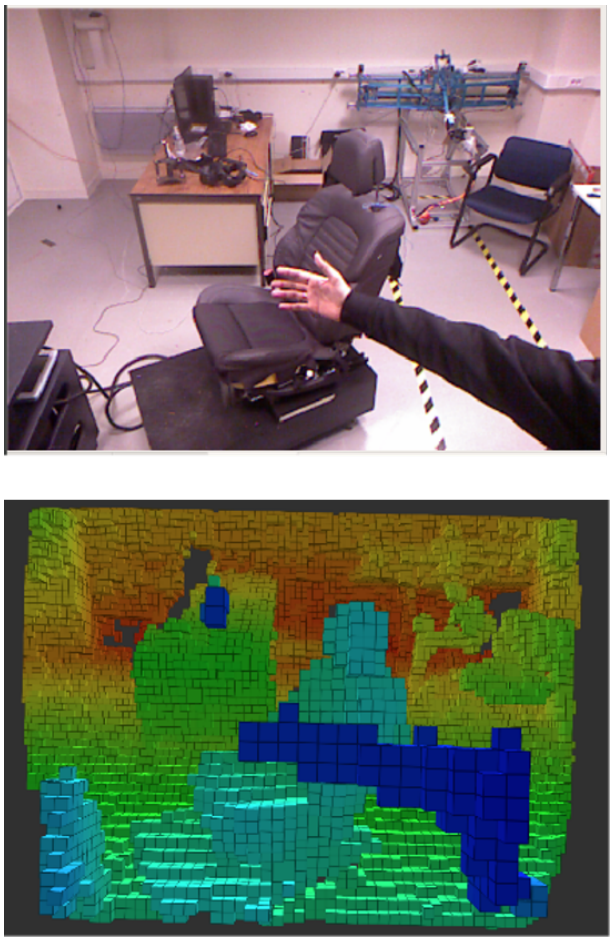
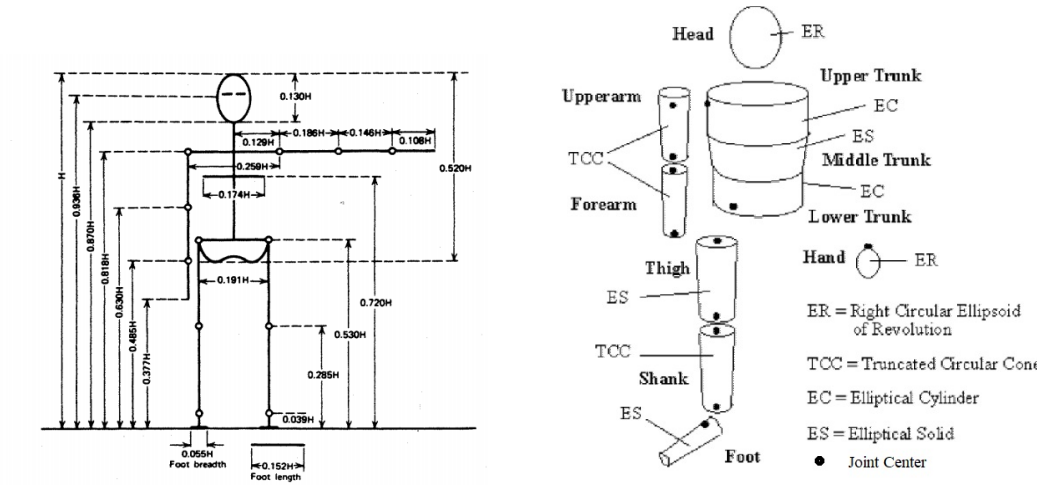


Figure 3.2 – The environment scanned using Kinect.



(a) Anthropomorphic human dimensions, c Winter, DA, 1990.

(b) The modified Hanavan model.

Figure 3.3 – Models referenced in this experiment.

3.3 Description of the motion capture system

In this section, we will describe the motion capture system used to capture the movements of the subject. This motion capture system is an indoor whole-body measurement system based on HTC Vive Technology and located in a dedicated room under some specific conditions, as we explain below. The system is used to help mimic the user motion based on a custom user model that will be designed and explained in further sections. The systems returns five frames (corresponding to the four segments of the body's model plus an additional one corresponding to the floor), as a function of time.

The placement of the sensors is shown in Figure 3.6. The motion capture system consists of two base stations (timed infrared emitting stations), two Vive-trackers are fixed to the different parts of the body of the subject, and a calibration tool.

— Two base stations also known as the lighthouse tracking system are two black boxes that create a 360-degree virtual space. The base stations emit timed infrared pulses at 60 pulses per second that are then picked up by the headset and controllers with sub-millimeter precision. They are placed at the corners of the motion capture area (delimiting the space where the subject can move) and looking towards the middle of the defined space.

— The VIVE trackers are a set of frames designed to be attached to different parts of the body. These trackers (as can be seen in Figure 3.6 and Figure 3.5) that are easily traceable by the base stations. The trackers are rigid and transmit information on the position and orientation of the segment.

— The calibrating tool is another Vive-tracker used to calibrate the position and orientation of the cameras w.r.t the real world. In order to do so, it is necessary to place this tool in a specific location, to link both the real and virtual environments. This has to be done every time the system is initialized, or for some reason if one of the base stations has somehow been moved with respect to the other.

3.4 Construction of human model

For the given scenario, we use four Vive-Trackers and one HMD. These act as markers for the tracking system. The major measurements required to construct the user are his arm dimensions and the distance of head w.r.t to base and shoulder. The model constructed (Figure 3.4), assumes that the user is seated on a chair. The whole model is defined in a URDF. Since the human arm is a 7 dof system, with two spherical joints each at the shoulder and wrist and a single revolute joint near the elbow. We use two trackers each on each arm. These trackers are positioned one on the hand, and the other closer to the elbow. From the trackers the position and orientation are collected. The positioning of the trackers is based on the key position information required (eg: accurate position of the user hand). Figure 3.7 show the frames of the model and the trackers. The tracker frames V_s and V_w can be seen at the translation in Z of 5 cm and 2 cm respectively. This translation is the approximated distance of the tracker position from the bones of the user arm. Other key dimension like W, L_1 and L_2 are calculated from the height of the user, on the basis of the Winter model.

Calculation of the joints values is done based on the tracker values. Each tracker gives the rotation and translation with respect to a fixed frame.

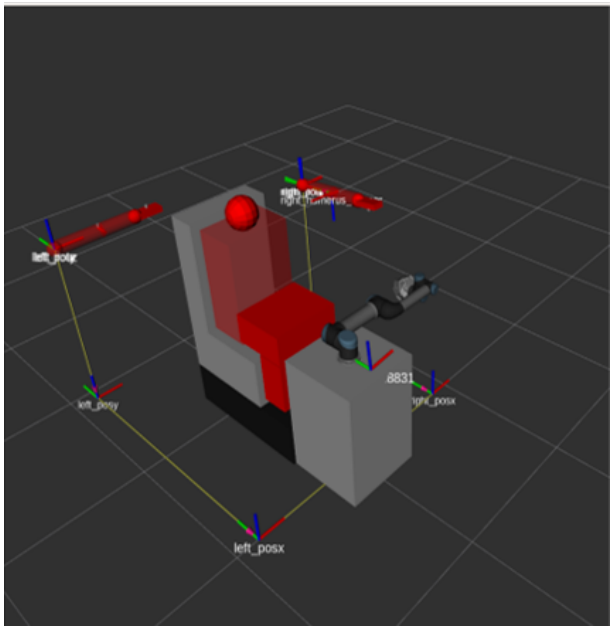
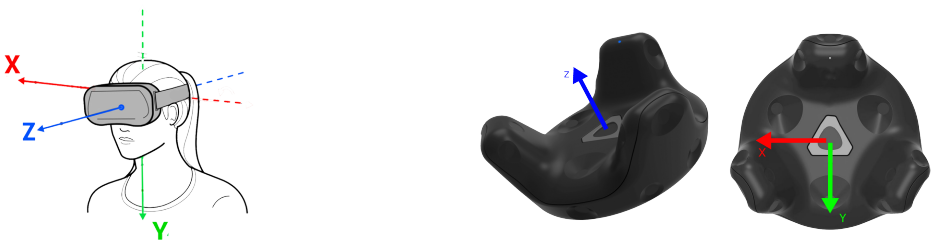


Figure 3.4 – The model visualized in Rviz.



(a) Frame on HMD.

(b) Frame on Vive-tracker.

Figure 3.5 – Tracking system used.

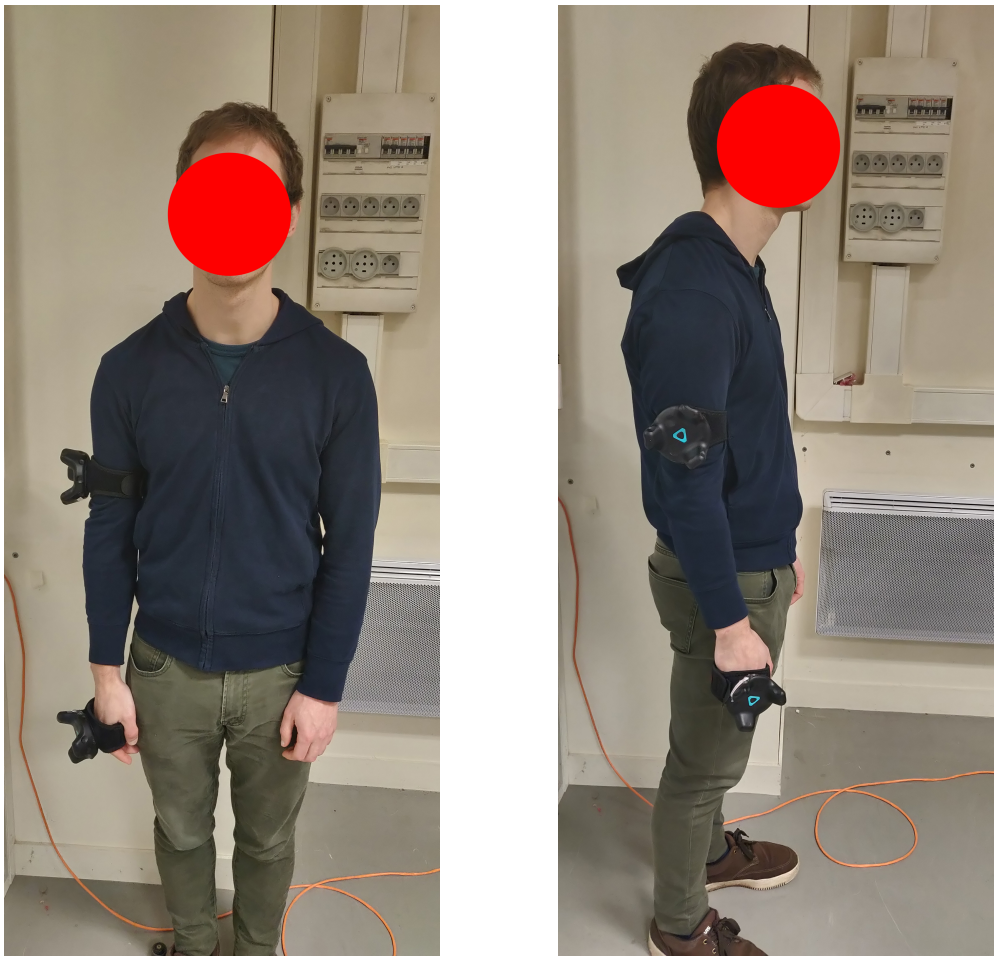
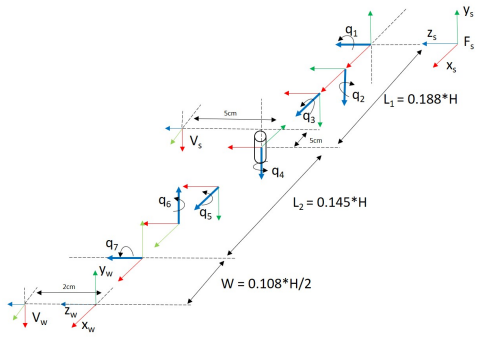


Figure 3.6 – Front and side view of the subject attached with the trackers.



(a) Transform frames for right hand.

Link	σ_i	q_i	a_i	α_i	d_i
s - 1	0	q_1	0	0	0
1 - 2	0	$q_2+\pi/2$	0	$\pi/2$	0
2 - 3	0	$q_3+\pi/2$	0	$\pi/2$	0
3 - V_s	2	0	L1-0.05	$\pi/2$	0
3 - 4	0	q_4	L1	$3\pi/2$	0
4 - 5	0	q_5	L2	$\pi/2$	0
5 - 6	0	$q_6+\pi/2$	0	$\pi/2$	0
6 - 7	0	q_7	0	$\pi/2$	0
7 - V_w	2	$3\pi/2$	W	0	0

(b) D-H parameter table.

Figure 3.7 – Model for right arm.

3.4.1 Calculation of the configuration of the arm

We are concerned mainly about the hand of the subject. So in this section, we will deduce the equations that obtain the joint values of the arm, for a given set of positions transmitted by the trackers in time.

We will analyze the mechanism of the arm system shown in Figure 3.7. This system consists of 7 joints to be controlled using two trackers. We collect information of position and orientation from both the trackers (6 by each tracker, 3 position and 3 orientations) and the objective is to obtain the 7 joint values of the arm model. The mechanism starts with the shoulder frame F_s followed by three revolute joints (q_1, q_2 and q_3). The shoulder and elbow are separated by a distance L_1. Then there is the elbow joint q_4. Frame V_s is related to the vive tracker frame near the user's elbow. From the elbow we have again three revolute joints (q_5, q_6 and q_7). These frames constitute the wrist joints. The elbow and wrist are separated by a distance L_2. From the wrist frame we have a hand link separated by W. The frame V_w is related to the vive tracker on the hand of the user. Now to compute the joints and mimic the user movement we need to actuate the joint based on the information from the vive trackers attached to the user.

Computing the transformation matrix from the F_s frame to the joint θ_3 of the arm model leads to a T_{03} matrix. Tracker information (V_s) is known and is connected to the upper-arm (the tracker information in V_s is w.r.t reference frame). The transform from reference frame (base tracker) to F_s is known. The vive tracker connected to upper arm allows to define the three first rotation of the shoulder based on the orientation information given by the vive tracker.

$$R_{03} = \begin{bmatrix} -\cos(\theta_1)\sin(\theta_2)\cos(\theta_3) - \sin(\theta_1)\sin(\theta_3) & \cos(\theta_1)\sin(\theta_2)\sin(\theta_3) - \sin(\theta_1)\cos(\theta_3) & -\cos(\theta_1)\cos(\theta_2) \\ -\cos(\theta_2)\cos(\theta_3) & \cos(\theta_2)\sin(\theta_3) & \sin(\theta_2) \\ -\sin(\theta_1)\sin(\theta_2)\cos(\theta_3) + \cos(\theta_1)\sin(\theta_3) & \sin(\theta_1)\sin(\theta_2)\sin(\theta_3) + \cos(\theta_1)\cos(\theta_3) & -\sin(\theta_1)\cos(\theta_2) \end{bmatrix} \tag{3.1}$$

$$\theta_1 = 2\tan^{-1}(R_{03}(3,3), R_{03}(1,3)) \tag{3.2}$$

$$\theta_2 = 2\tan^{-1}(R_{03}(2,3), \sqrt{(R_{03}(2,1))^2 + (R_{03}(2,2))^2}) \tag{3.3}$$

$$\theta_3 = 2\tan^{-1}(-R_{03}(2,2), R_{03}(2,1)) \tag{3.4}$$

For joint θ_4, we know the transform matrix information of tracker V_s and joints θ_1, θ_2 and θ_3. For a point on the forearm segment the translation in x-axis (in F_s frame) will be same for T_{04} and V_s. Using the tracker information from V_s and taking that position in x value and equating it with the transform matrix from T_{04}. Gives the equation 3.5, where $x_1 = 2\ cm$.

$$P_x = -(-(\cos\theta_1 \sin\theta_2 \cos\theta_3 - \sin\theta_1 \sin\theta_3)\sin\theta_4 - \cos\theta_1 \cos\theta_2 \cos\theta_4)L_2 - (-(\cos\theta_1 \cos\theta_2 \sin\theta_3 - \sin\theta_1 \cos\theta_3))L_1 + x_1 \tag{3.5}$$

This equation can be written as:

$$P_x - (C) = (A)\sin\theta_4 + (B)\cos\theta_4, \tag{3.6}$$

with,

$$\begin{aligned} A &= (\cos\theta_1 \cos\theta_2 \cos\theta_3 + \sin\theta_1 \sin\theta_3)L_2 \\ B &= -\cos\theta_1 \sin\theta_2 L_2 \\ C &= (\cos\theta_1 \cos\theta_2 \sin\theta_3 + \sin\theta_1)L_1 + x1 \end{aligned}$$

Equation 3.6 is a type 2 equation [71] and the angle θ_4 can be deduced :

$$\theta_4 = atan2(\sin\theta_4, \cos\theta_4) \tag{3.7}$$

Equation 3.7 gives two solutions but only one can be true due to the limited angle of the elbow joint ($180\ ^\circ > \theta_4 > 0$).

Now, for computing θ_5, θ_6 and θ_7 using rotation part of T_{47}, which is

$$R_{47} = \begin{bmatrix} -\cos(\theta_5)\sin(\theta_6)\cos(\theta_7) - \sin(\theta_5)\sin(\theta_7) & \cos(\theta_5)\sin(\theta_6)\sin(\theta_7) - \sin(\theta_5)\cos(\theta_7) & -\cos(\theta_5)\cos(\theta_6) \\ -\cos(\theta_6)\cos(\theta_7) & \cos(\theta_6)\sin(\theta_7) & \sin(\theta_6) \\ -\sin(\theta_5)\sin(\theta_6)\cos(\theta_7) + \cos(\theta_5)\sin(\theta_7) & \sin(\theta_5)\sin(\theta_6)\sin(\theta_7) + \cos(\theta_5)\cos(\theta_7) & -\sin(\theta_5)\cos(\theta_6) \end{bmatrix} \tag{3.8}$$

Since transform matrix information of tracker V_w and joints θ_1, θ_2, θ_3 and θ_4 are known, T_{47} can be computed.

$$T_{47} = T_{40}T_{07} \tag{3.9}$$

T_{07} is taken from the V_w tracker and T_{40} can also be known as we know $\theta_1, \theta_2, \theta_3$, and θ_4.

$$\theta_5 = 2\tan^{-1}(R_{47}(3,3), R_{47}(1,3)) \tag{3.10}$$

$$\theta_6 = 2\tan^{-1}(R_{47}(2,3), \sqrt{(R_{47}(2,1))^2 + (R_{47}(2,2))^2}) \tag{3.11}$$

$$\theta_7 = 2\tan^{-1}(-R_{47}(2,2), R_{47}(2,1)) \tag{3.12}$$

Extend the model for both arms

The model explained so far has been extended for both hands, to get complete motion capture information of both arms. This setup gives complete information about the arms that can be view as dynamic obstacles for the robot in the environment.

This can help in improving the safety of the user. Also, an additional joint is introduced at the waist to introduce two rotations at the hip and to model the rotation of the torso. These joints are calculated according to the information provided by the trackers. Figure 3.8 shows the complete structure of frames. As explained before in section 3.3 five sensors are used. The

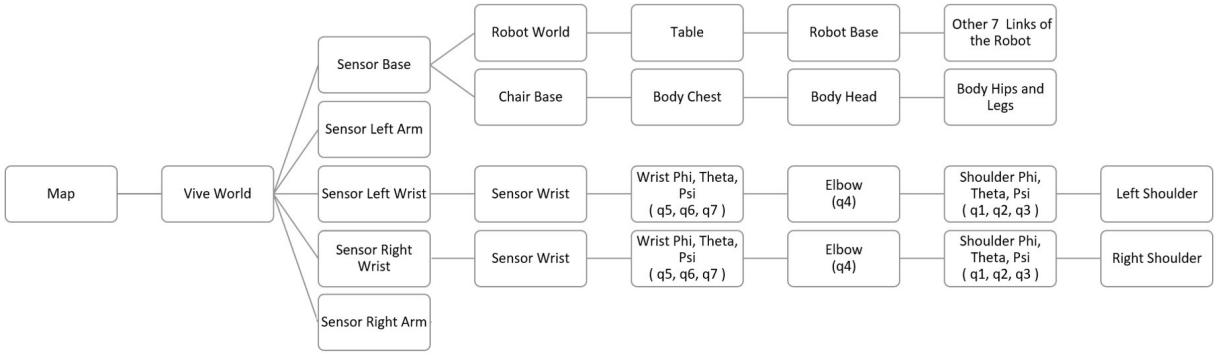
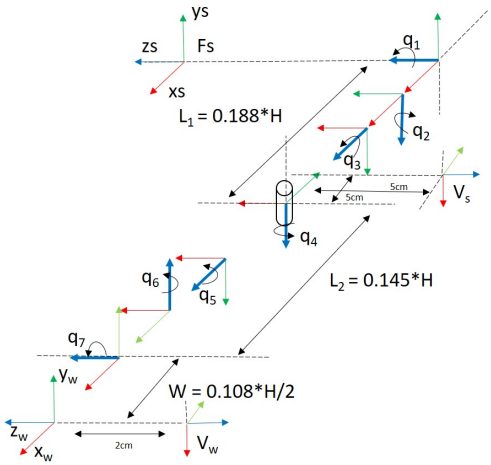


Figure 3.8 – Structure for both arms.



(a) Transform frames for left hand.

Link	σ_i	q_i	a_i	α_i	d_i
s - 1	0	q_1	0	0	0
1 - 2	0	$q_2+\pi/2$	0	$\pi/2$	0
2 - 3	0	$q_3+\pi/2$	0	$\pi/2$	0
3 - V_s	2	0	L1-0.05	$3\pi/2$	0
3 - 4	0	q_4	L1	$3\pi/2$	0
4 - 5	0	q_5	L2	$\pi/2$	0
5 - 6	0	$q_6+\pi/2$	0	$\pi/2$	0
6 - 7	0	$q_7+3\pi/2$	0	$\pi/2$	0
7 - V_w	2	0	W	$3\pi/2$	0

(b) D-H parameter table.

Figure 3.9 – Model for left arm.

connection of these sensors to the user model and physical world is shown. A base sensors acts as the reference frame for the real and virtual world elements.

A change between the models for each arm (Figure 3.7a and Figure 3.9a) is in the direction of axis, especially for the tracker frames V_w and V_s. Applying the similar approach explained in section 3.4.1 based on the model shown in Figure 3.9. Gives these below equations:

$$R_{03} = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2)\cos(\theta_3) + \sin(\theta_1)\sin(\theta_3) & \cos(\theta_1)\sin(\theta_2)\sin(\theta_3) + \sin(\theta_1)\cos(\theta_3) & \cos(\theta_1)\cos(\theta_2) \\ \cos(\theta_2)\cos(\theta_3) & \cos(\theta_2)\sin(\theta_3) & \sin(\theta_2) \\ \sin(\theta_1)\sin(\theta_2)\cos(\theta_3) - \cos(\theta_1)\sin(\theta_3) & -\sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - \cos(\theta_1)\cos(\theta_3) & \sin(\theta_1)\cos(\theta_2) \end{bmatrix} \tag{3.13}$$

$$\theta_1 = 2\tan^{-1}(R_{03}(3,3), R_{03}(1,3)) \tag{3.14}$$

$$\theta_2 = 2\tan^{-1}(R_{03}(2,3), \sqrt{(R_{03}(2,1))^2 + (R_{03}(2,2))^2}) \tag{3.15}$$

$$\theta_3 = 2\tan^{-1}(R_{03}(2,2), R_{03}(2,1)) \tag{3.16}$$

$$P_x = -((\cos\theta_1 \sin\theta_2 \cos\theta_3 - \sin\theta_1 \sin\theta_3)\sin\theta_4 - \cos\theta_1 \cos\theta_2 \cos\theta_4)L_2 - (\cos\theta_1 \cos\theta_2 \sin\theta_3 - \sin\theta_1 \cos\theta_3)L_1 + x_1 \quad (3.17)$$

where $x_1 = 2\ cm$,

$$P_x - (C) = (A)\sin\theta_4 + (B)\cos\theta_4, \quad (3.18)$$

with,

$$\begin{aligned} A &= -(\cos\theta_1 \cos\theta_2 \cos\theta_3 + \sin\theta_1 \sin\theta_3)L_2 \\ B &= \cos\theta_1 \sin\theta_2 L_2 \\ C &= -(\cos\theta_1 \cos\theta_2 \sin\theta_3 + \sin\theta_1)L_1 + x1 \end{aligned} \quad (3.19)$$

This gives the equations for θ_4 as,

$$\theta_4 = atan2(\sin\theta_4, \cos\theta_4) \quad (3.20)$$

Now, for computing θ_5, θ_6 and θ_7 using rotation part of T_{47}, which is

$$R_{47} = \begin{bmatrix} \cos(\theta_5)\sin(\theta_6)\cos(\theta_7) - \sin(\theta_5)\sin(\theta_7) & \cos(\theta_5)\sin(\theta_6)\sin(\theta_7) - \sin(\theta_5)\cos(\theta_7) & \cos(\theta_5)\cos(\theta_6) \\ \cos(\theta_6)\cos(\theta_7) & -\cos(\theta_6)\sin(\theta_7) & -\sin(\theta_6) \\ \sin(\theta_5)\sin(\theta_6)\cos(\theta_7) - \cos(\theta_5)\sin(\theta_7) & \sin(\theta_5)\sin(\theta_6)\sin(\theta_7) + \cos(\theta_5)\cos(\theta_7) & \sin(\theta_5)\cos(\theta_6) \end{bmatrix} \quad (3.21)$$

Since transform matrix information of tracker V_w and joints θ_1, θ_2, θ_3 and θ_4 are known, T_{47} can be computed.

$$T_{47} = T_{40}T_{07} \quad (3.22)$$

where T_{07} is taken from the 2nd tracker and T_{40} can also known as we know θ_1, θ_2, θ_3, and θ_4.

$$\theta_5 = 2\tan^{-1}(R_{47}(3,3), R_{47}(1,3)) \quad (3.23)$$

$$\theta_6 = 2\tan^{-1}(-R_{47}(2,3), \sqrt{(R_{47}(2,1))^2 + (R_{47}(2,2))^2}) \quad (3.24)$$

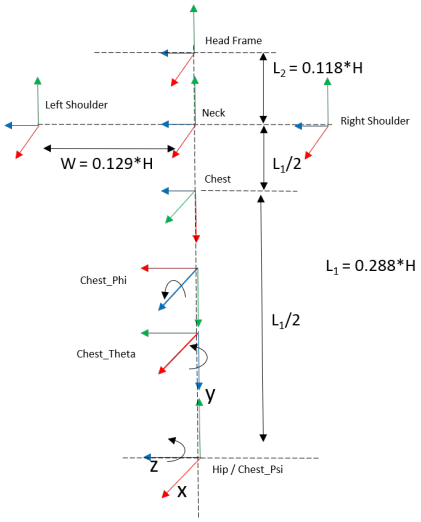
$$\theta_7 = 2\tan^{-1}(-R_{47}(2,2), R_{47}(2,1)) \quad (3.25)$$

3.4.2 Calculation of movement in torso

So far we have calculated the hand pose and arm configurations of the human model. We are also interested in the torso motion of the user seated on the chair. For torso motion, we consider a simple rotation between the hip and head/neck. We assume that the user can bend forward, backward, sideways, and also be able to turn. We assume that when he/she bends there is no curvature of his/her back.

To achieve these values we use the information from the trackers placed near the elbow. From the position of the tracker V_s near the elbow we deduce the position of the shoulder frame F_s of the model. This frame F_s is used as the main reference information. We know the distance between the two shoulder frames from the winter model, based on the user height. Now we have the position of both the shoulder frames F_s from both trackers on each hand.

The neck frame is based on the two shoulder frames. It is known that human can have cartesian displacement at shoulder and we assumed that the displacement are symmetric at the two shoulders. The hip frame is assumed to be constant since the user is seated. For the user model, we assume that the shoulder frames are always aligned in a line.



Link	Joints	a_i	α_i	d_i
Hip - Psi	θ_1	0	0	0
Psi -Theta	$\theta_2+\pi/2$	0	$\pi/2$	0
Theta - Phi	$\theta_3+\pi/2$	0	$\pi/2$	0
Phi - Chest	$\pi/2$	$L_1/2$	$\pi/2$	0
Chest - Neck	$\pi/2$	$L_1/2$	0	
Neck - Left	0	0	0	W
Neck - Right	0	0	0	-W
Neck - Head	0	$L_2+(L_1/2)$	$\pi/2$	0

Figure 3.10 – Transform frames and D-H parameters for chest.

In Figure 3.10 shows the connection between the frames. The shoulder frames Left and Right are computed based on the position of the vive trackers. Now the goal is to compute the chest_psi (ψ), chest_theta (θ) and chest_phi (ϕ) to have the information about rotations about the torso. It is to be noted that the model designed is disconnected at the shoulders. So the position of the user torso and legs is based on the base tracker that is used (explained in Figure 3.8).

Based on the shoulder frames the neck frame is calculated. From the base reference sensor, using the DH table in Figure 3.10 the position of the hip frame is defined. First, using the shoulder frames a line is drawn passing through the three frames (two shoulder frames and neck). This line S is used to compute the angles of rotation for the torso.

Projecting line S onto the chest frame gives the rotation information of the torso. Projection of line 'S' onto XY, YZ, and ZX planes of chest frame. The dot product of line 'S' and the projected line on the planes gives rotation about the respective axis. Now have all the required motions of the user, next tests were carried out to check the accuracy of the model.

3.5 Analysis of user model

3.5.1 Accuracy of arms

Within the model, two small points have been created within the humerus and palm links, which represent the location of the sensors in the user as represented in Figure 3.13. To test the accuracy in arms of the model, two additional frames (U_s and U_w) were introduced in the model which are positioned such that they align with the real frames for the tracker (V_s and V_w). The frames U_s and U_w are child frames to the frames q_3 and q_7 of the model. So the hypothesis is using the information from the vive trackers (V_s and V_w), the joints of the arms are found. So based on the joint values the position of the frames U_s and U_w are found. So to check the accuracy of the model an error between the values of the frames U_s and V_s is computed.

This analysis was done and the resulting measure of the error computed for U_s and V_s for each arm (right and left) can be seen in Figure B.1 and Figure B.2. The errors computed are magnitude less than 1 cm.

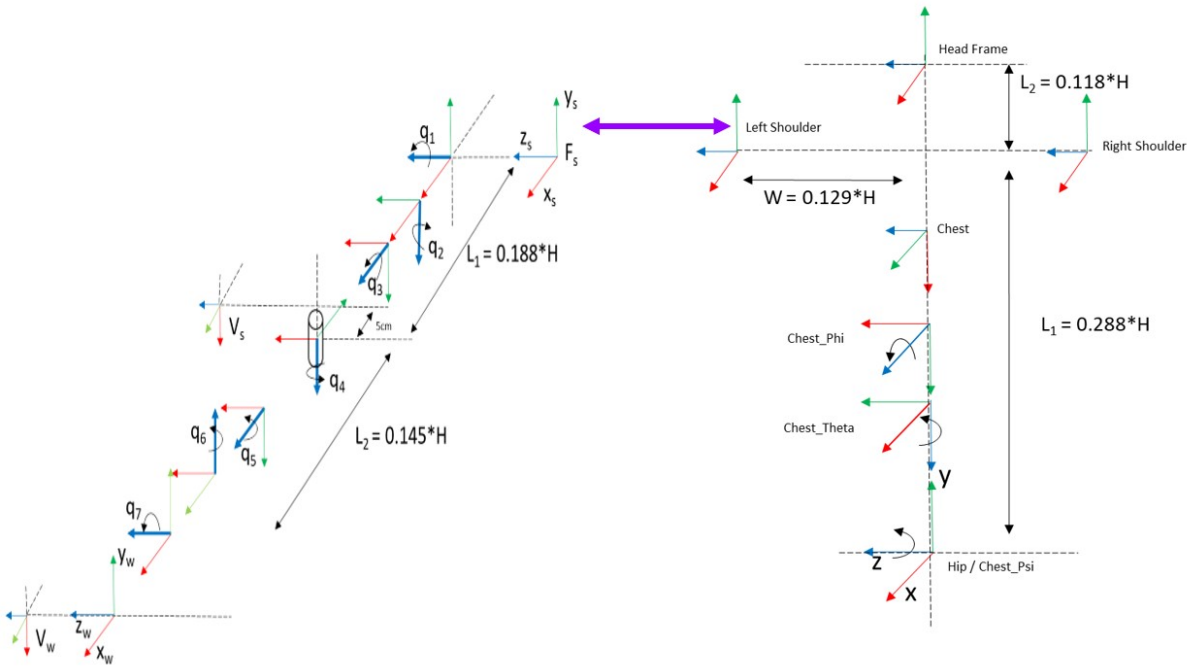


Figure 3.11 – Transform frames for estimation of error between shoulder frames.

Similar analysis was done and the resulting measure of the error computed for U_w and V_w for each arm (right and left) can be seen in Figure B.3 and Figure B.4. The errors computed are magnitude less than 1 cm.

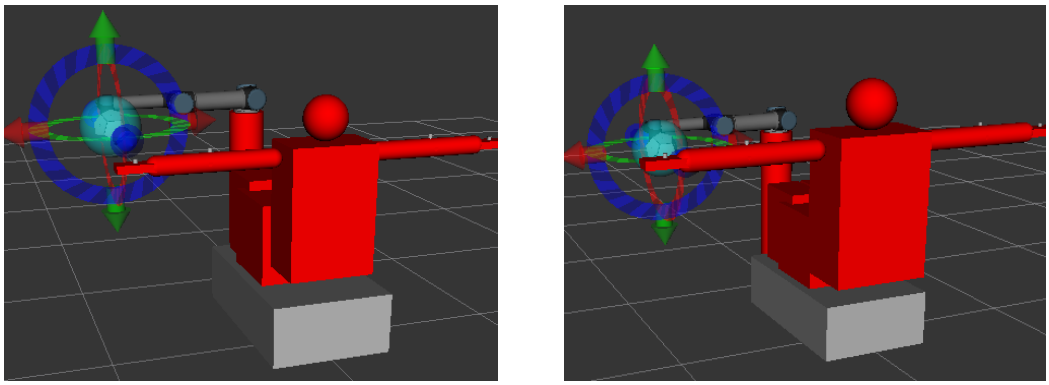
3.5.2 Accuracy of torso

The key frames computed to simulate the torso are the shoulder frames as explained in section 3.4.2. It was known that using the tracker we compute the shoulder frames (F_s). Now to make the error analysis we add two additional frames on the torso part of the user model. These are the expected shoulder frames based on the Hanavan model (ideally we want the shoulders to be connected with the body, Figure 3.11). Calculation of the error between the real and the computed position of shoulder frames was done to test the accuracy of the model. Figure B.5 shows the error computed between the real and estimated position for the right shoulder. It can be seen that the error is close to 0.1 cm. The same can be observed for the left shoulder in Figure B.6.

A video demonstrating the working of the user model can be found here.

3.5.3 Scalability of the model

The mannequin also has the option of scalability (Figure 3.12), having a different size proportionate to the user's height to have a better relationship between the real world and the model.



(a) Mannequin model for a height of 1.6 m (b) Mannequin model for a height of 2 m

Figure 3.12 – Scalability of the mannequin model for a height of 1.6m and 2m.

3.6 Conclusions

To model the user, a mannequin was defined in a URDF robot model. The hip of the model is fixed, while the arms are structured as 7 DoF serial robot with all revolute joints, where the first 3 constitute the shoulder, the 4th joint represents the elbow and the last 3 rotating joints represent the wrist of the arm.

User test was done to test the accuracy and response of the model designed. In the scenarios the user was seated and was asked to move the hand and the torso. Two types of analyses were performed. The first one was to check the accuracy of the sensors position in real life and in the model. The second was to find the accuracy in the shoulder frames computed during the calculation of the torso rotations.

Figure 3.13 – User with the HTC vive trackers

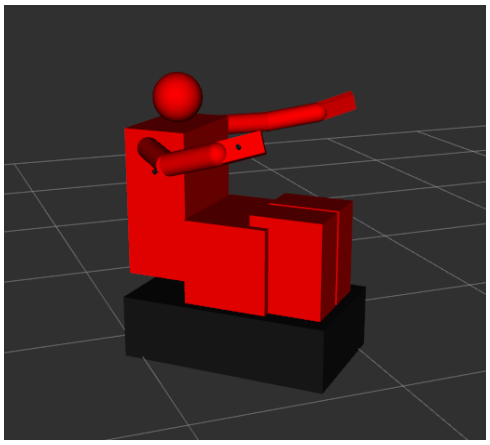


Figure 3.14 – Complete mannequin model of the user.

Chapter 4

System architecture and robot trajectory planning

4.1 Introduction

So far we have established the base location of the robot and the occupancy of user in the environment. This chapter is intended to provide an insight into the tools used in the development of the experiments, such as the software architecture, data flow between softwares used, a description of the system's environment, and communication between the systems. Then we will concentrate on the trajectory planning of the robot. Assuming that we have predicted the user's intention (which will be explained in next Chapter) through his/her gaze direction and the position of his/her dominant hand (the one touching the object) Now, the goal is to move the robot to the appropriate location. The robot motion to be performed must avoid collision with the user and also with itself.

When the user uses HMD vision interfaces and has to perform haptic evaluations, he/she no longer sees the real scene, but only a virtual world. His/her physical reference points quickly disappear except for objects he/she touches such as his/her seat and the dashboard. It is challenging to move the robot in the physical environment such that the robot doesn't cause any harm to the user.

The outline of this chapter is as follows. In section 4.2 an introduction to the proposed architecture and data flow is explained. A brief description of all the components in the system is explained. Section 4.3 deals with the preliminary setup of the systems explained in the architecture. Section 4.4 deals with path planning for the robot. A complete analysis is made between different planning algorithms and compared to find the best planning algorithm. Finally, general remarks and conclusions are commented in section 4.5.

4.2 Proposed architecture and data flow

As established in chapter 1, there are multiple systems 1) Unity in Windows which has the display of the virtual environment and the user prediction strategy (explained in chapter 5), 2) ROS in Ubuntu which is used for the robot trajectory planning (will be studied in this chapter) and user occupancy (explained in chapter 3) and 3) the robot system itself. Figure 4.1 show the flow of data between these systems. A detailed explanation of modules inside ROS (like MoveIt) will be dealt in detail in following sections of this chapter.

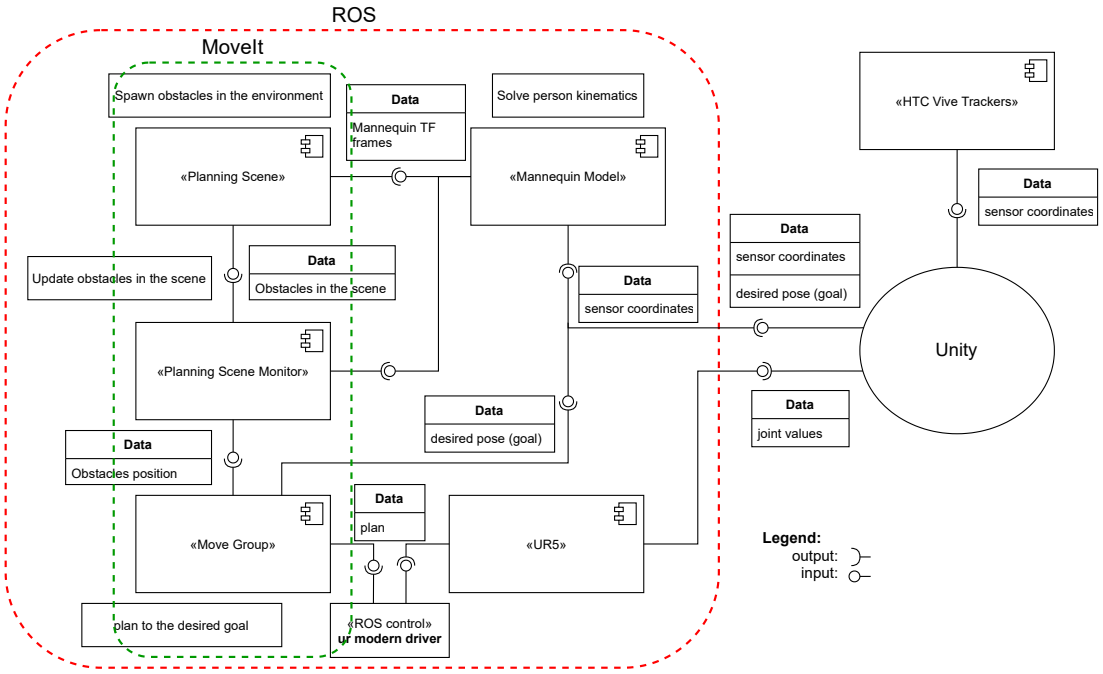


Figure 4.1 – System's Architecture.

The proposed architecture provides an overview of how the instances share information and communicate with each other, the HTC Vive sensors connected in Windows communicate their positions and orientation. The same information is passed between Windows and Ubuntu system. Additional information of the predicted desired point of intersection is also sent from Unity to ROS in Ubuntu. From the information of the desired point and location of the user model, robot planning is done and the pre-computed planned trajectories are sent to the robot controller for executing the trajectory.

ROS [5] is the middleware that communicates with the robot and Unity. Based on this information, pre-computed trajectories (using MoveIt) are selected so that the robot reaches the desired positions while knowing the current states of obstacles in the scene. Once the trajectory is selected, we communicate with the UR-5 robot. Thus, we can move the UR-5 robot with the ROS control, and send as output the current states of the robot's joints for visualization in Unity.

To move the robot to a specific position and orientation, the position and orientation of the cube are sent through a ROSBridge connection to a robot controller at a frequency of 50 Hz. This controller runs on a thread library that consists of two threads, which read data from Unity VR software and write data to the real-time data interface of the UR-5 at a frequency of 100 Hz.

The proposed architecture of the project describes the different interactions each element of the system has and provides an insight into how the instances share the information and communicate with each other as shown in Figure 4.1. The architecture is a description of the way the system works and which tasks are taken care by which instances.

This chapter explains the ROS section described in the architecture, where it receives as input the desired goal and the sensor's data, with the sensor's data we can compute the kinematics of the mannequin model, which allows the "planning_scene" (in MoveIt) to spawn the objects that correspond to the mannequin in the scene, for them to be kept up to date by the

"planning_scene_monitor" (in MoveIt).

Later based on this information and the desired goal, the "move_group" (in MoveIt) can generate a plan for the robot to reach the desired positions while knowing the current states of obstacles in the scene. Once the plan is generated, we communicate to the UR-5 robot by using the *ur_modern_driver* [131] (this driver is currently deprecated, but given the UR-5 software version available at the lab, is the one that has to be used). With it, we can move the UR-5 robot with ROS control, and send as output the current joint states of the robot to Unity system to be used for visualization of robot in the virtual environment, if required.

4.2.1 MoveIt

MoveIt [27] is one of the most known and used software for robot manipulation, used on over 150 robots. It is an open-source software allowing the development of projects in industrial, commercial, and research environments for free.

It counts with a user-friendly platform for building flexible industrial, research, and commercial applications. It allows the configuration, programming, and definition of different robot models, allowing to develop and define the environment, objects and plan different tasks where path planning is required. It allows to perform and develop complex applications while being able to determine and study the interactions of the system by incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control, and navigation.

The MoveIt architecture is based on two main nodes, the "*move_group*" node, and the "*planning_scene*" (Figure4.1). The "move_group" takes care of obtaining the parameters, the setup, and the individual components of the robot model being used, so it can provide to the user *ROS* services and actions for the users to use on the robot.

The second element which is the "planning_scene" uses the "*planning_scene_monitor*" to handle the scene in which the robot will be included. This is what will constitute the obstacles or elements the robot has to interact with. Providing information about where they are located, and any updates that happen in the robot's surroundings. It uses as well the robot interface data to make the connection between the data being processed and the objects defined in the scene.

A detailed explanation of the components and functionality of MoveIt can be found in Appendix C

4.3 Implementing laboratory setup based on proposed architecture

The laboratory setup (Figure 3.1) is conformed by the UR-5 robot system and a vehicle chair, in a face-to-face configuration. Where the robot's location and height have been determined in Chapter 2 to be 80 cm above the ground. Being a position optimal enough so the robot is able to reach all the interaction points where the system has an interest in reaching. For the user (Figure 3.13), the VR helmet and trackers are attached to the body, in the humerus, and in the palms so the data can be obtained and place the user's location within the VR environment.

4.3.1 Requirement

In this scenario, a relation between the real world and the simulated environment in ROS has to be established using the vive trackers used by the user. Additionally we want the end-effector

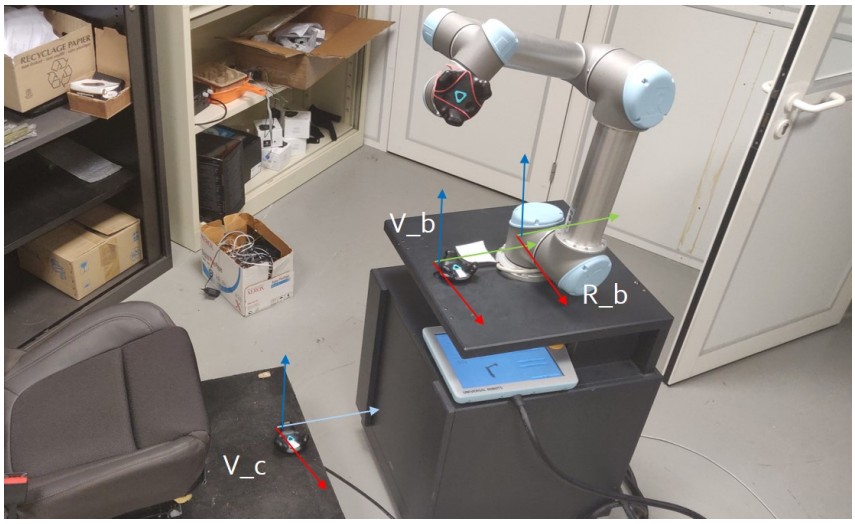


Figure 4.2 – The system setup in the physical environment

of the robot to follow the user's hand and contact it when necessary. To achieve this, we need good movement planning. We also want no part of the robot to come into contact with the user during movement tracking.

4.3.2 Methodology for calibration of real and virtual environments

The ICI system can be divided into two physical components shown in Figure 4.2, the robot and chair. Each component has a vive-tracker to localize between them. These components need to be calibrated to synchronize with the virtual and the real physical environment, this is achieved by using trackers. For this purpose, we consider some information, shown in Figure 4.3 we can see two trackers (V_b and V_e) and two robot frames(R_b and R_e) that need to be calibrated to sync the environments. The physical robot has base frame R_b and tool frame R_e, by attaching a vive-tracker at the tool end and assuming that the frames coincide (V_e and R_e), we calibrate the base tracker V_b, by assigning a translation in x (between V_b and R_b).

To achieve the requirements of safety in the present scenario, a virtual plan is introduced that prevents the robot from crossing the interior of the car. The user and robot are located on the opposite side of the plane. This virtual plan is defined in relation to the virtual car surfaces. The user's hand is equipped with a tracker, to track and follow the hand. In Figure 4.2 we can see a tracker on the chair. Two trackers are used to depicting two frames V_c(vive tracker on chair) and V_h (Vive-tracker on users hand). One tracker is used to define the car/chair plane(to differentiate the interior and exterior of the car). The other tracker is used to track the user's hand. Once the calibration between the physical and virtual environment is done, the tracker at the robot tool end can be removed V_e.

The collection of information is achieved by using OpenVR libraries.The algorithm used for collecting the tracker information can be seen in Algorithm 3

Using the OpenVR library, the position and orientation of the tracker are known with respect to a vive-world frame. Both robot and vive systems have information that needs to be linked to perform hand tracking using the vive-trackers.

To establish connection we find the transformation, assuming V_e and R_e are superposed, compute from respective systems ${T_{rb}}^{re}$ and ${T_{ve}}^{vb}$ and compute Eq 4.1. Now the systems are

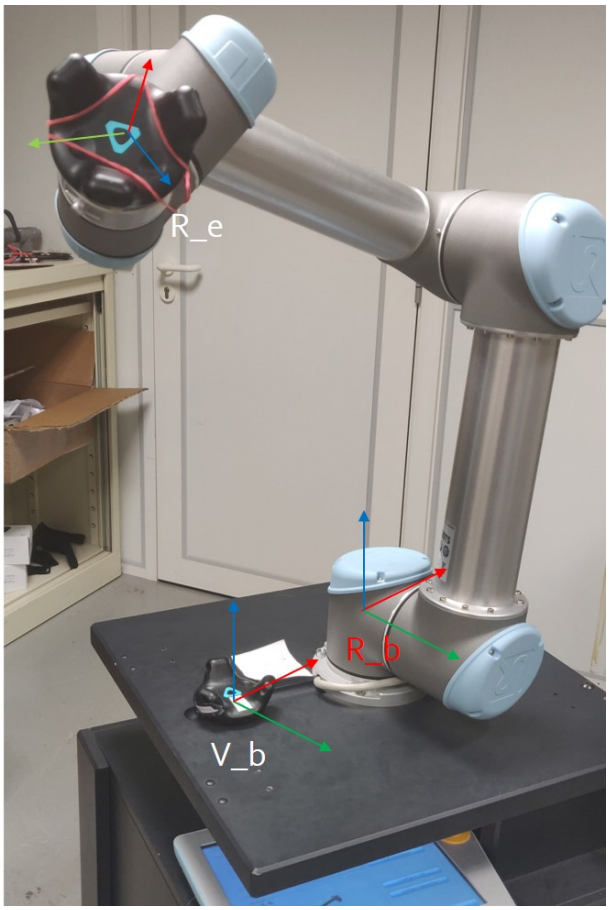


Figure 4.3 – Calibration setup of the physical components.

Algorithm 3 Tracker information in ROS.

Input: Vive tracker frames and Names.
Output: New frames of vive trackers wrt ROS frames.

```
Convert all trackers to appropriate coordinate system.

function VIVE_TRACKER.
    triad_openvr.triad_openvr().
    get_serial, device_name.
    if device_name prefix LHR then
        rotate 180 degrees in +Z axis.
    else
        broadcast frames to /tf time stamped.
        (x,y,z), (qx,qy,qz,qw), time, device, vive_world
        [parent: vive_world, child:device]
    end if
end function
```

Algorithm 4 Tracker world reference in ROS.

Broadcast a frame to /tf.

function VIVE_WORLD.
    broadcast a frame vive_world to /tf.
    translation =[0, 0, 0,].
    rotation = 90 degrees about x axis.
end function

connected.

$$T_{rb}{}^{vb} = T_{ve}{}^{vb} T_{rb}{}^{re} \tag{4.1}$$

So three trackers (Figure 4.2) are used to calibrate the systems. Next, the objective is to project the tracker in hand on to plane created with the tracker on the bottom of the chair. This is a test to check the system are connected, especially since the reference coordinate frames for robot (right handed system) and Unity (left hand system) are different. Algorithm 3 and 4 show the conversion of the coordinate systems and algorithm 5 describes the for hand tracking.

Algorithm 5 Hand tracker information in ROS.

Input: Vive tracker frames and names.
Output: Goal hand tracker.

To give the final position of the hand w.r.t to Robot base.

function HAND_TRACKING.
    listening to transforms T_h and T_c.
    T_h; parent_fr:car_tracker; child_fr:hand_tracker; .
    T_c; parent_fr:base_robot; child_fr:car_tracker;
    function PROJECTION(T_h).
        get_translation(T_h)
        set y = 0.
        return Projection
    end function
    desired position = $T_c \times$ Projection
end function

For the test, tracker in hand V_h is projected onto the tracker plane of the car V_c (which we use as reference for the plane Figure 4.4). Then the transformation to the base of the robot is calculated. The new transform is the desired location to which the robot needs to move. This desired location is on the surface of the car and not inside (interior) of the car.

When following the hand, at every instance the projection of the hand onto a plane is taken. With the robot controls, that projection point on the x-z plane is reached. Algorithm 6 describes precisely these above steps.

After calculation of projection of hand onto car plane, initialization of robot and its controllers is done to make the robot move/track user. The move group is created and named ”manipulator”. A ready position is set which is close to the tracking region and away from the singularity home position.

Algorithm 6 Motion of robot based in hand position.

Input: Vive tracker frames and names.
Output: Robot Motion.

move robot to desired hand position.

function ROBOT_MOTION.
    Initialize Robot
    Initialize move group "manipulator"
    Set velocity Scaling Factor = 0.1
    move in joint space from home to desired
    if Plan Success **then**
        Execute
    else
        Exit
    end if
    while set rate **do**
        function PROJECTION(T_h)
            return desired
        end function
        function MOVE_ROBOT(desired)
            set first point in waypoint (get_currentpose())
            set waypoint (desired)
            F = move_group.computeCartesianPath()
            if F×100 = 100 **then**
                Execute
            else
                Nothing
            end if
        end function
    end while
end function

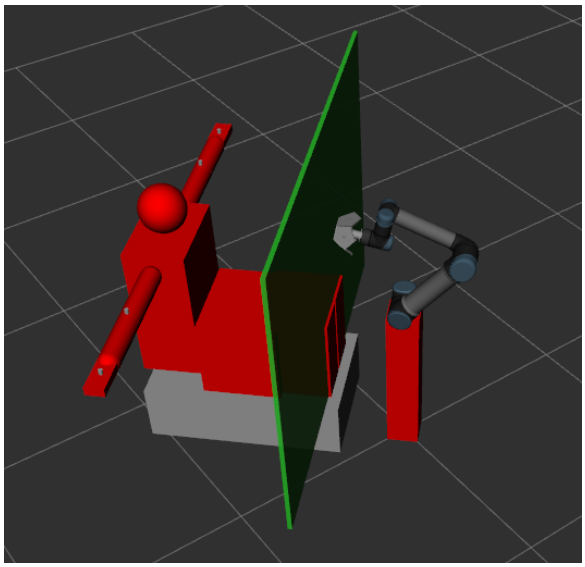


Figure 4.4 – Representation of the user's effective workspace as a plane

4.4 Computation and trajectory planning

So far we have established the flow of data between the systems and how the environment in ROS is setup. Next is to explain the planning methods used as well as the different stages in which it was developed. Starting from the setup and configuration phase, where we establish the environment and defined the planning group. Afterward, tests were run to determine which one of the multiple path planning algorithms available within the MoveIt software behaved the best on some testing experiments.

Path planning (Figure 4.5) refers to the computation or generation of a geometric path, which connects an initial point to a final one, through in-between waypoints. Waypoints are meant to be followed in order to perform a desired task or motion. This geometric computation is based on the kinematic properties of the robot along with its geometry. On the simplest case, path planning need to be performed on static and known environments. However, this problem can also be generated for robotic systems subject to kinematic constraints in a dynamic environment, which are more complex to achieve.

In the different possible scenarios, path planning can be done using a previously known map, which is defined as global planning, and it is commonly used in determining the possible paths that can be followed in order to reach the final position. This is used for the case of a known and static environment, where the position of the present obstacles is already known, and a path is generated taking this into account. This can be performed off-line, as it is based on previously known information. In the case of possible changing environments, local path planning needs to be done, which relies on sensors or any other type of data-providing interfaces in order to obtain updated information about the robot's surroundings. This can only be done in real-time, as it depends on dynamically changing environments.

4.4.1 Various path planning algorithms

The configuration space can be denoted as C_{space} [83], where inside this space, the sub-spaces where the robot configuration collides with and obstacle, are called C_{obs}, and the space where

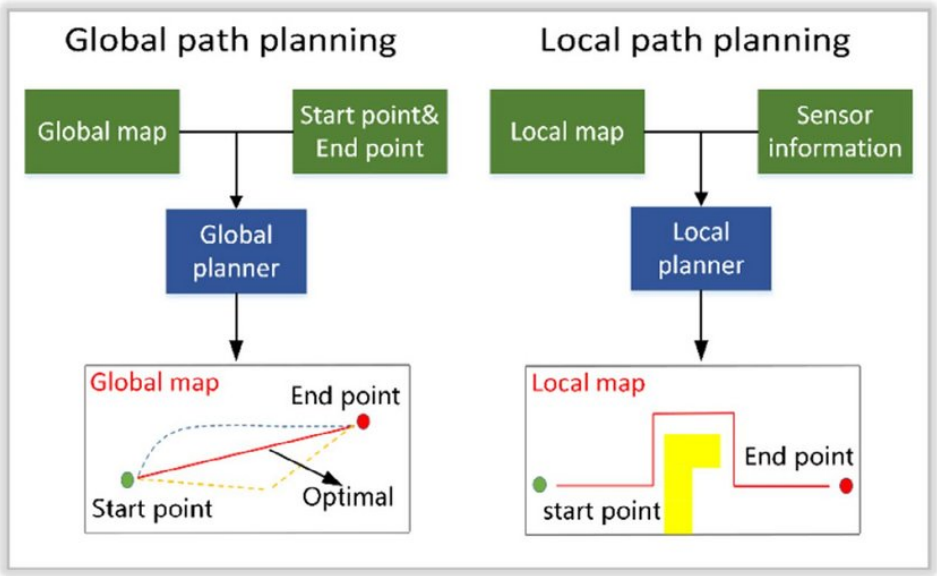


Figure 4.5 – Work logic of *global* [69] and *local* [72] path planning.

it does not collide is called the free configuration space C_{free}.

There has been multiple proposals on path planning algorithms throughout the years, from where we can provide a review of the bases and workings for the most commonly found algorithms in the robotics literature, in [46] some of these algorithms are addressed.

Artificial Potential Fields

The Artificial Potential Fields (APF) approach [72] introduced by O. Khatib in 1985 and further developed by [135] [136]. It represents the path planning algorithm as a problem where the robot is considered as a "moving ball" or point under the influence of potential fields which are artificially generated (thus the name) by the desired goal position and the obstacles within the C_{space}. In this case the desired goal generates an attractive potential and the different obstacles generate repulsive potentials, where the sum of the two contributions creates a potential field which can be translated into an artificial force that will lead the "ball" to go towards the goal, while avoiding the obstacles. The succeeding configurations of the robot can be determined given the direction of the slope of the force field which the robot is subjected to, this direction represents the most optimal path towards the goal. The force field can be expressed as:

$$\mathcal{U}_{art}(q) = \mathcal{U}_{goal}(q) + \mathcal{U}_{obs}(q) \tag{4.2}$$

Where for a configuration q, the artificial potential field $\mathcal{U}_{art}(q)$ is formed by the attractive potential generated by the goal $\mathcal{U}_{goal}(q)$ and the repulsive potential of the obstacles $\mathcal{U}_{obs}(q)$.

There are two main proposals in the literature to generate the attractive potential field of the goal in order to guide the motion in the C_{space} towards it. These are the *paraboloidal field* and the *conical or linear field* approach, for a $C_{space} \in R^2$ (Figure 4.6).

However this method presents conflicting situations such as: a) trap situation due to local minima; b) oscillation in the presence of obstacle; c) no passage between close spaced obstacles; d) oscillations in narrow passages. Some solutions to the local minima problem has been pro-

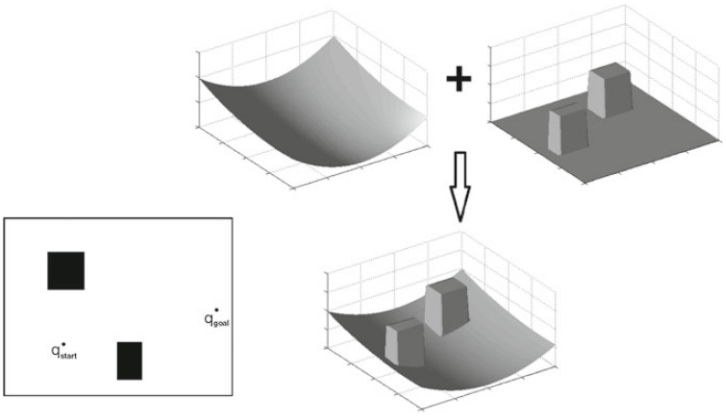


Figure 4.6 – Representation of an Artificial Potential Fields.

posed, like the usage of Harmonic Potential Functions [65], virtual local target [119] and others such as [28] [110].

Due to the way that the environment is represented, one feature of this algorithm is that it is computationally inexpensive and has good reactivity to environment changes, taking into account new obstacles that come into the C_{space} allowing it to have good real-time reactions to the environment.

Probabilistic Road-maps

The Probabilistic Road-maps approach [69] consists in generating random nodes in the configuration space (C_{space}) in order to generate a grid (so called, the road-map). The algorithm (Figure 4.7) is divided into two phases, the learning stage, where the nodes are generated, and the query phase where the generated nodes are then connected and an optimal path is found. This connections are only possible if there are no objects present in between nodes, which will lead to a road-map where all the connections are feasible paths for the robot to follow. This algorithm is considered as a Sampling-Based planner, as it does not fully reconstruct the C-space and its boundaries, but instead, checks if each sampled robot configuration is in collision or not, in order to consider it as a valid node or not. Once a sufficient amount of nodes has been generated in the C_{space}, it then performs the connections between all the neighboring nodes where no obstacle is present in between. This will generate the road-map, where multiple paths can be followed between q_{init} and q_{goal}, so another algorithm such as $A*$ [53] or Dijkstra [33] is used to find the shortest path within the road-map.

Afterwards, some other techniques can be used in order to do path improvement. Like trying to connect directly two non-adjacent nodes from within the path, checking consequently if for the following nodes a connection is also possible, this will potentially reduce the amount of connections needed within the path, reducing its total length.

The advantages on this type of algorithms is that is fast and has probabilistic completeness, which, by not fully exploring the C_{space} but instead sampling it, reduces the amount of computation needed. On the other hand it also generates a big amount of "useless nodes" as only the ones included in the final path are the ones that matters. Another problem is that it is a trial and error algorithm, to which an unsuccessful case of finding a path might just mean that there is not enough nodes in the road-map to fully connect q_{init} and q_{goal}, so re-tuning the

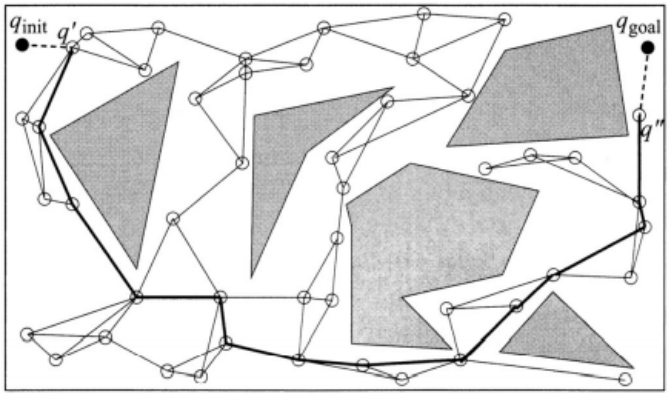


Figure 4.7 – Example of a road-map on *SE(2)*, the gray areas are the obstacles within the C_{space}, the empty circles correspond to the randomly generated nodes and the bold line is the shortest path obtained by Dijkstra's's Algorithm.

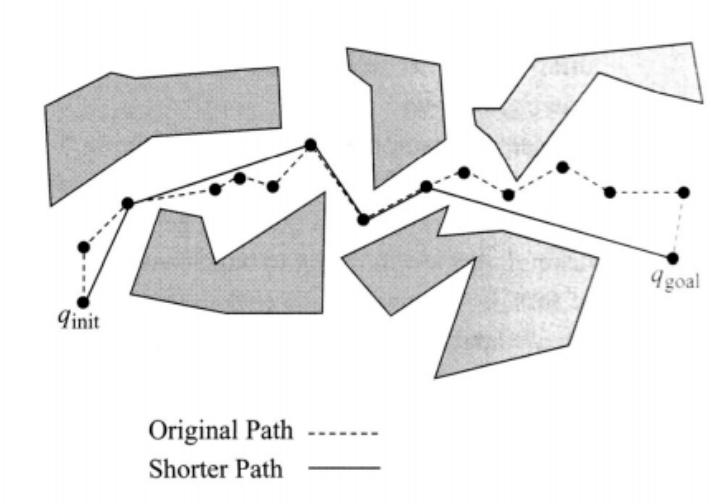
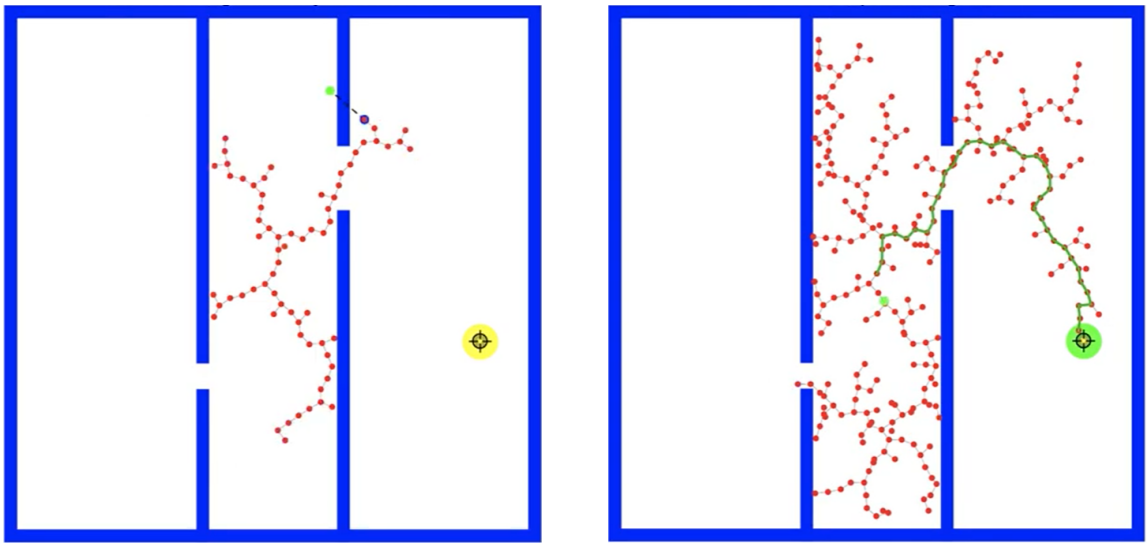


Figure 4.8 – Optimization of the obtained path by connecting non-adjacent nodes.



(a) (b)

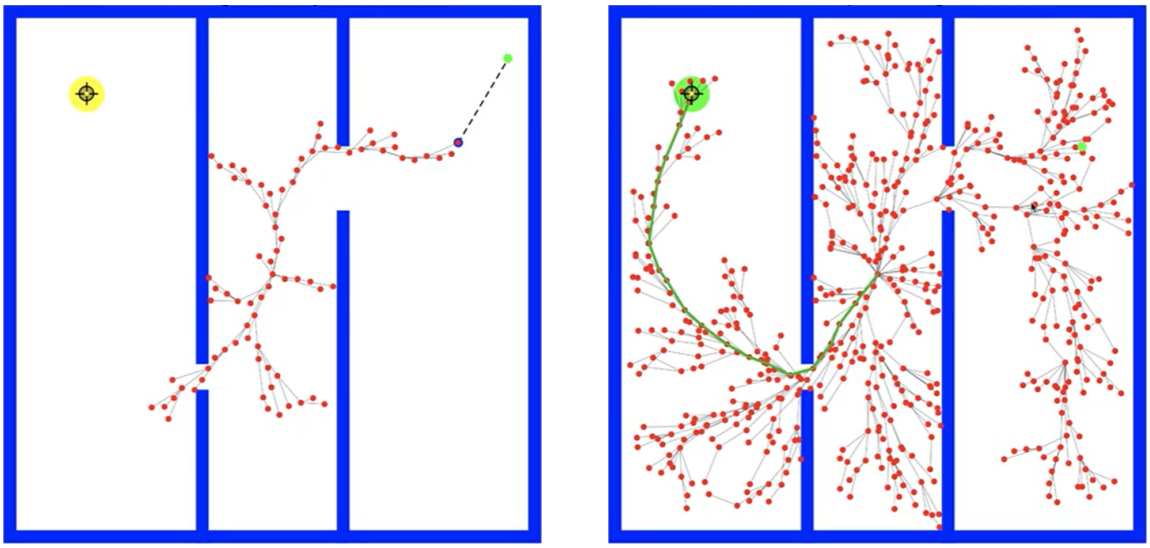
Figure 4.9 – RRT: (a) Building of the tree connecting the new node to the closest node towards the random point (b) Path obtained between q_{init} and q_{goal}.

parameters can provide a successful outcome. This algorithm is also not suitable for dynamic environments, as the road-map has to be recomputed and rewired every time obstacles moves in the environment, having to check if the previously computed nodes still belong to C_{free} or now belong to C_{obs}.

Rapidly Exploring Random Trees

Rapidly Exploring Random Trees (*RRTs*) [80], introduced by S. LaValle in 2001 as an optimization from the classical Random Trees algorithm. It consists in a tree of nodes which is generated starting from the initial configuration until it finds a feasible path towards the desired goal configuration. This node-tree (Figure 4.9) is built by taking into consideration a random point in the C_{space}, and connecting the closest node within the tree to a newly generated node in that direction and at a fixed step-size, in this way, the tree is constantly growing in a random way, but always from the closest node to the random point. The newly generated nodes are always checked in order to ensure they belong to C_{free}, if they do not belong, the step-size is reduced until they do. Due to its always increasing behavior, this algorithm is able to explore the C_{free} space in a fast and random way, and stops once it finds a feasible path between the initial and final configurations. One of the drawbacks is that, due to the randomness, the final path is not the shortest, and once it is found, the algorithm does not perform any optimization on it. Another characteristic is that the obtained path is generally not a smooth path, this last one can be slightly optimized by using spline interpolation on the obtained nodes that form the path.

Another important improvement of this algorithm is called RRT* (Figure 4.10) [68] presented by S. Karaman and E. Fazzoli in 2011, which works under the same principle as the original RRT, but with the addition of two new considerations. The first one is that in this case, each node has a cost assigned, linked to the traveled distance relative to its parent node. In this way, after the closest node of the tree to the point has been found, all the surrounding nodes of



(a) (b)

Figure 4.10 – RRT* (a) New node connected to the cheapest neighbor parent (b) Obtained path with the shortest and smoothest path.

the parent node are examined within a radius, and if a node with a cheaper cost is found, the cheaper node becomes the new parent and the connection to the new node is made. The second consideration is that, this algorithm allows rewiring of the tree. When a new node is added, its neighbors also check if being connected to the new node will decrease their cost, and if so, the neighbor nodes are rewired to the newly added node.

One of the biggest advantages of this method is that is able to find a close-to-optimal path, and the obtained path can be infinitely optimized for as long as new nodes are being added, obtaining very smooth paths. It also has the characteristic that it is a forward projection algorithm, meaning that once a connection is made between the initial configuration and the desired configuration, the path is already obtained. On the other hand, a big drawback of this method is that it is computationally expensive due to the constant checking that needs to be done for each node within the algorithm (check if the node is outside an obstacle, check the cost to confirm shortest path, rewiring of the neighbors, etc).

Other proposed alternatives commonly found are:

— Informed RRT* [45]: Where the random point generation is biased in order to place the point at the desired goal configuration a percentage of times or every set amount of iterations. This makes the tree to try to grow towards the desired goal in order to decrease the computation time.

— BRRT* [67]: Starts generating a tree from the initial configuration and another tree from the desired configuration, and when both trees meet, the path is obtained.

— RT-RRT* [97]: This is an improvement of the *RRT** algorithm where the algorithm allows changes in the q_{init} state and rewires the previously generated map so it can be used for dynamic environments and changing initial and desired configurations.

— BiTRRT [32]: Bi-directional transition-based *RRT* is a version of *RRT* where each new node added to the tree goes through a transition test. When the closest nodes of the trees are closer than ten times the extension step-size, and if it is possible to connect them following a downhill slope, both trees are merged.

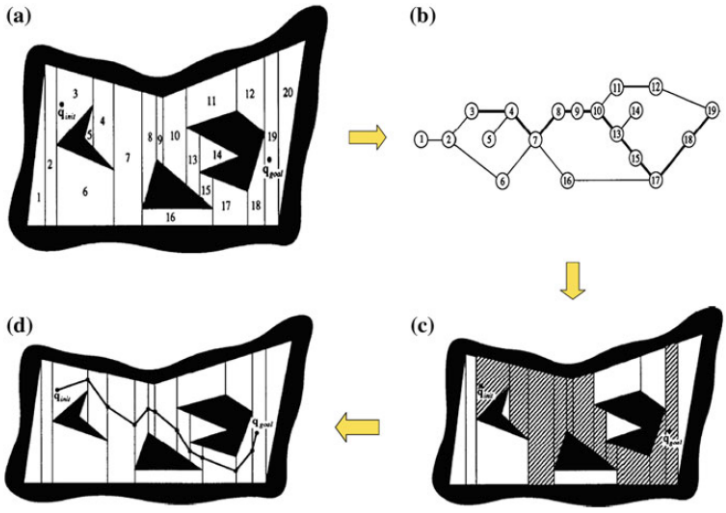


Figure 4.11 – Exact cell decomposition method (a) Nodes generated based on vertex found (b) Connectivity graph built from adjacent nodes (c) Nodes corresponding to the path found on the connectivity graph (d) Final path generated by connecting the middle points between adjacent nodes.

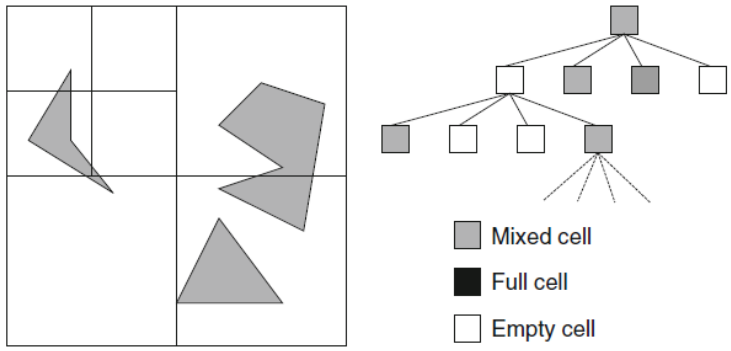
Cell Decomposition

Another approach commonly found in the literature are the Cell Decomposition algorithms (Figure 4.11) [122]. The idea of this approach is to subdivide the C_{space} into smaller obstacle free regions called cells, which then are connected by building a graph of adjacency [117] in order to find a path between the initial and final configurations. The method is subdivided also into two different categories of cell decomposition approaches: the exact cell decomposition (ECD), and the approximated cell decomposition method (ACD).

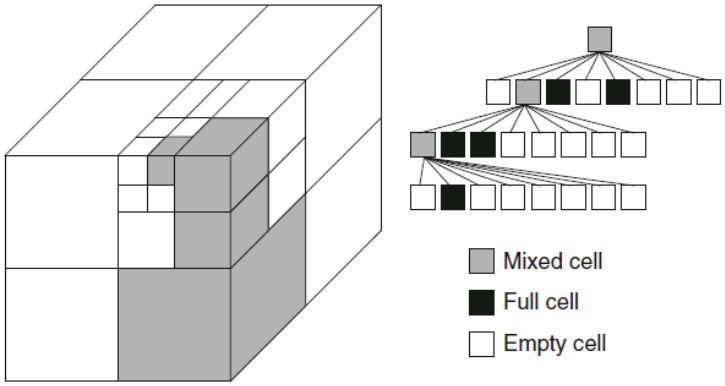
The first one, uses geometrically based algorithms to explicitly determine the obstacles and build the cells. This is done by drawing parallel vertical lines from each vertex found in the C_{space}, and considering the shape of the object and of the C_{space}, it can find the vertex corresponding either to an obstacle or to the C_{space}'s boundaries. This will allow the algorithm to build a cell-based space which represents exactly to the C_{free} space (free of obstacles). The counterpart of this approach is that it's mathematically expensive, as it is difficult to obtain the exact representation of the C_{free} space.

Afterwards, each obtained cell is numbered and represented as a node in the connectivity graph, built based on the adjacency relationship between the free cells. Then a path is computed from the node containing the initial configuration, to the node containing the desired one, by following the shortest path on the graph (using A* or similar). In this way, the cells of the shortest path are determined, and the real path is generated by connecting the middle point of the transitions between adjacent cells, in order to ensure that the final path is as far as possible from the C_{space} boundaries and the obstacles.

The approximated cell decomposition method, also called "quadtree" decomposition or "octree" for 3D spaces (Figure 4.12), was proposed due to the high computations and geometric calculations required in the exact cell decomposition method. The algorithm works by determining fully-free, completely full or mixed obstacle cells inside the C_{space}. Initially the C_{space} is divided into four equal regions, the algorithm determines the mixed cells, and subdivides them



(a)



(b)

Figure 4.12 – (a) Quadtree decomposition for 2D spaces (b) Octree decomposition for 3D spaces.

again in the same way recursively until a fixed resolution is reached or until each cell lies either as a fully-free cell or as a completely-full cell. Once one of these criteria is met, the algorithm stops decomposing the space. Finally, a path is obtained from the generated tree created by the decomposition, where the initial configuration cell gets connected to the final configuration cell through a fully-free cells path. This algorithm is resolution complete, which means that the algorithm's completeness depends the resolution of the grid, while the exact method has total completeness as it exactly represents the C_{space}.

In [3] an alternative method combines cell decomposition methods with Rapidly exploring random trees (RRT) [80], where ACD is used to initially find a path on preliminary static workspaces, and RRT is then used to validate the path and find alternatives on collisions due to changing environments. In this work, RRT and ACD algorithms are combined together in order to exploit the advantages of each of them. The RRT planner has relatively high tolerance to obstacles shapes and workspace changes. Where this feature is missing in ACD planner. In addition, the RRT is not effective in small areas or narrow passage, while ACD planner does not face this problem.

4.4.2 Comparison of various path planning algorithms

In robotics, the path planning task has proven to be one of the most challenging tasks, specially when conditions such as expecting real-time behaviors or in dynamically changing environments. A comparison of the presented planning algorithms has been presented in Table 4.1.

Table 4.1 – Comparative table between path planning algorithms.

Algorithms	Advantages	Disadvantages
APF	— Easy to implement — Allows on-line planning — Commonly used method	— Local minima and oscillation problems — Goals can be non reachable — Difficulty in narrow passages
Cell Decomp	— Has completeness for exact approach — Can be used for on-line planning	— Resolution dependant in approximated approach — C_{space} is not described in approximated approach
PRM	— Allows multi-query scenarios — Probabilistic completeness without exploring all of the C_{space}	— Generates a lot of unused paths and nodes — Needs to rebuild road-map on dynamic environments
RRTs	— Forward projection algorithm, computes and finds the path — Explores rapidly the C_{space} — Has completeness	— Computationally demanding — Single-query

From the algorithms presented, *APF* and its variations provide a good adaptation for path planning in dynamically changing environments, where any obstacle entering the C_{space} generates a new repulsive field which can be taken into account in order to generate a new trajectory. But the local minima problem requires the use of alternative algorithms in order to overcome it.

The *PRM* case, it is well know for its ability to find a path without the need to explore the totality of the C_{space}, but it is also a graph based algorithm, which requires the use of shortest path searchers as *A**. It is proven to be highly efficient in static environments and can handle initial and final configuration changes, but if the objects in the C_{space} change positions, the connections between the nodes have to be re-done. Some alternatives propose to maintain the previously generated nodes and re-check if they belong to C_{free} or C_{obs}, and rebuild the graph based on that information and find a new path. Similarly the case for cell decomposition methods, where the graph search needs to rebuilt again. Nonetheless this methods has proven to be viable real-time options that can adapt to a dynamic environment.

Finally on *RRT* and *RRT** methods and alternatives, they are known to be good path planning methods, with the limitations that the generated trees are linked to the initial configuration and they have high computation demands. With the proposal of the different alternatives very optimal real-time path planners can be obtained, the limitations on these type of algorithms is that they require large memory capacity as the whole tree needs to be stored at all times, and it only works in bounded environments, so unbounded and large distance environments are still a challenge.

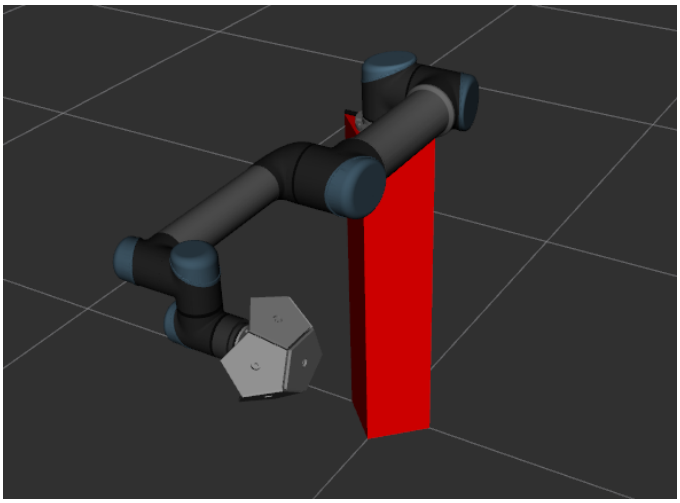


Figure 4.13 – Planning group for the robot system.

4.4.3 Selection of planning algorithm

Description of MoveIt planning group

The "planning_group" is defined as the group of elements that constitute the whole robot system. This is the UR-5 robot, the 6-faced prop, and finally the robot *pedestal* (Figure 4.13). These three elements are what the path planning algorithms need to consider as the robot, in order for them to avoid any collision state that exists with any of these elements.

The pedestal was modeled in such a way that it matches the dimensions of the real-world system. As established in Chapter 2, the determined height of the pedestal is 80 cm.

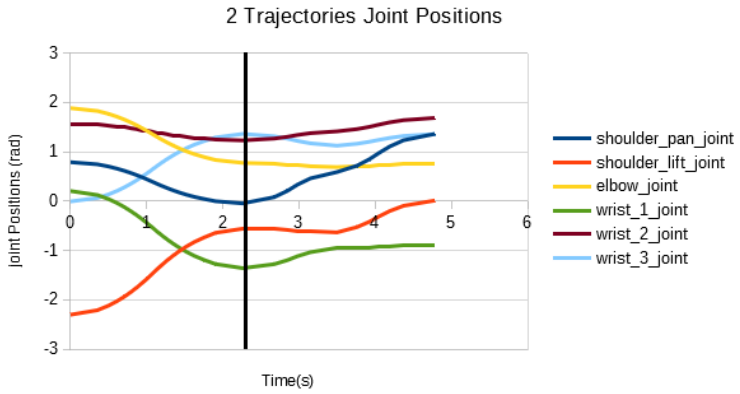
For the configuration of the "plannig_group", MoveIt has an in-built graphical interface that helps to create all the configuration files related to the kinematics, controllers, Semantic Robot Description Format (SRDF), and other files for the usage of the robot in ROS. This interface is called *MoveIt Setup Assistant*.

The MoveIt Setup Assistant creates all the mentioned files based on the robot description given to it, in this case, the UR-5 robot description files provided by [89] were taken, and modified in order to include the pedestal base (included in the URDF definition of the robot) and also the mesh file for the 6-faced end effector.

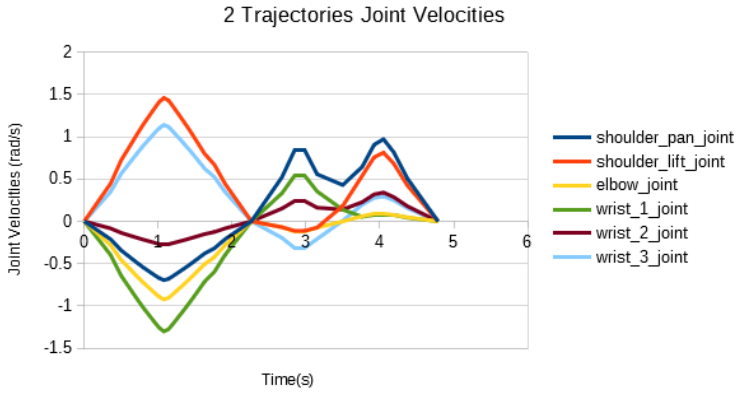
Different movement techniques

Different mobility alternatives that MoveIt API offers were analyzed. All the tasks related to the motion of the "planning_group" are handled by the "move_group" class. By being able to specify which planning group we want to consider, we can use all the different functions that the class offers for it, such as getting information on the current joint values, the target, configure the planning algorithm we intend to use, and perform the planning and execution of motions in the environment.

The "move_group" class has the option to perform path planning through different types of motions, these options can be chosen depending on the nature of the task. For example, we can set up as a goal a given pose in the space, or set it up as a desired joint value. Given the nature of the system, we will be working with joint value goals, as we expect to reach the different



(a)



(b)

Figure 4.14 – Planned paths using *move_group* with out re-planning. (a) Back-to-back plans. (b) Velocities for both plans.

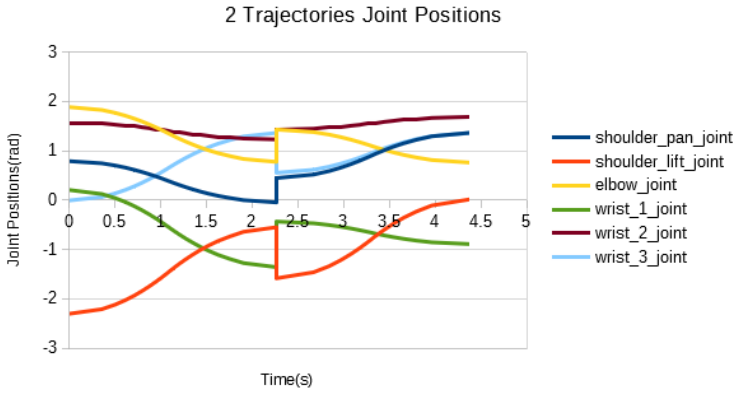
points in a specific configuration that provides a higher level of safety to the user (elbow up the configuration for the UR-5).

Another important feature is the option to specify whether we want to reach each one of the requested goals or not. As the implementation is going to be receiving constantly changing goals, the best implementation is to plan and go towards said goal, allowing for the system to re-plan if the goal changes, which means that we do not need to reach the initial goal.

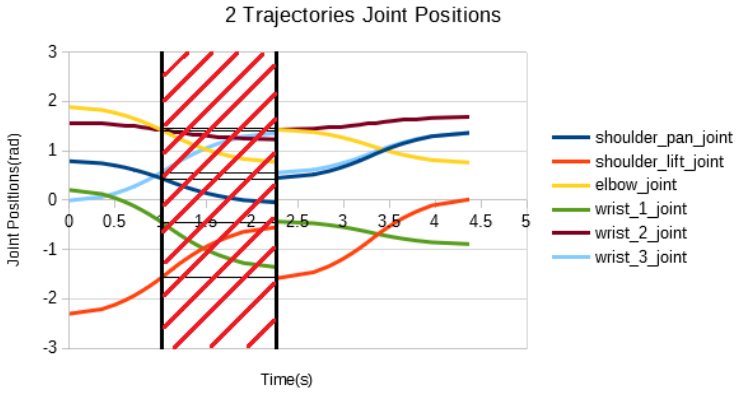
In Figure 4.14, two different trajectories are computed from an initial configuration to a mid goal and then to a final goal. In this case, we ensure that the robot is going to fully execute each one of the trajectories, reaching both goals. This is shown in Figure 4.14(b) where the velocities come down to zero, as it is performing a stop.

In the case of Figure 4.15, we computed the same two trajectories as before, but allowing re-planning during the execution of the first plan. In this case in Figure 4.15(a) we can see both plans one after the other, wherein Figure 4.15(b) we show the representation of the segment that was not executed from the first plan, as a re-planning scenario came to place. This is where the current positions of the first plan were taken as initial positions for the second plan, which gives as a result Figure 4.15(c), showing the two plans that were executed.

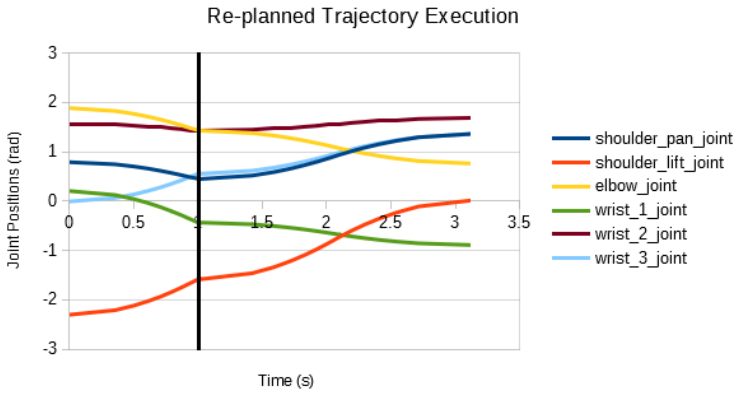
A parameter to select is the planning algorithm that suits best for the task. As previously



(a)



(b)



(c)

Figure 4.15 – Re-planned paths using *move_group* . (a) Back-to-back plans. (b) Segment not executed due to re-planning. (c) Final executed plan.

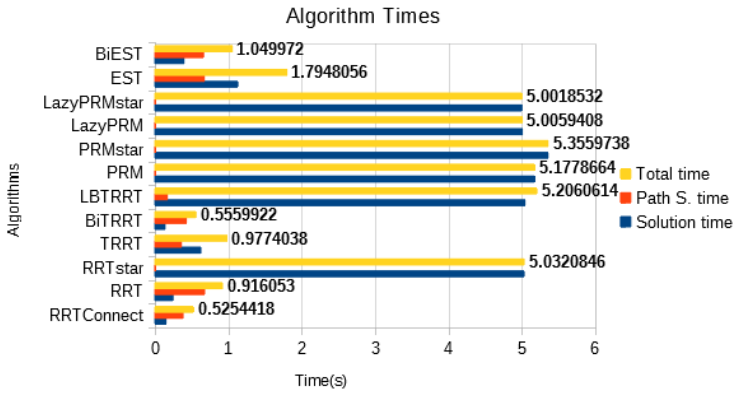


Figure 4.16 – Comparison of all planning algorithm's times.

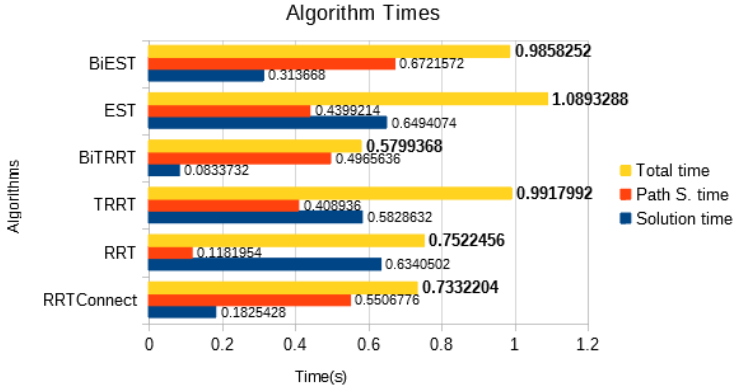


Figure 4.17 – Comparison of best planning algorithm's times

mentioned, MoveIt has multiple built-in path planning algorithms which can be used. To determine the best option, we iterated through all of the available options, and performed a planning task for the desired goal configuration, measuring the time it took for each one of the algorithms and registering the data. This was made for a total amount of five times for each one of the 12 available planning algorithms through 9 different trajectories. We then proceeded to take the average times they took to find a solution, perform path simplification (just for the algorithms that had this feature), and computed the average total time. With this data, we were able to select which algorithms behaved the best with the shortest planning times.

After doing the computations for half of the trajectories, given the big difference in planning times for some of the algorithms, we pre-filtered them in order to exclude those ones that presented the slowest planning times, in order to only consider the suitable candidates.

Another analysis that allowed us to select the algorithm which behaved the best for the implementation, was to perform an analysis of the generated trajectories with each one of the algorithms for a fixed task. Based on the pre-filtered algorithms from the previous analysis as a starting constraint, we computed the average execution time and waypoints for a set of trajectories.

This analysis was made for the same trajectories as in the previous graphs for a total of 10

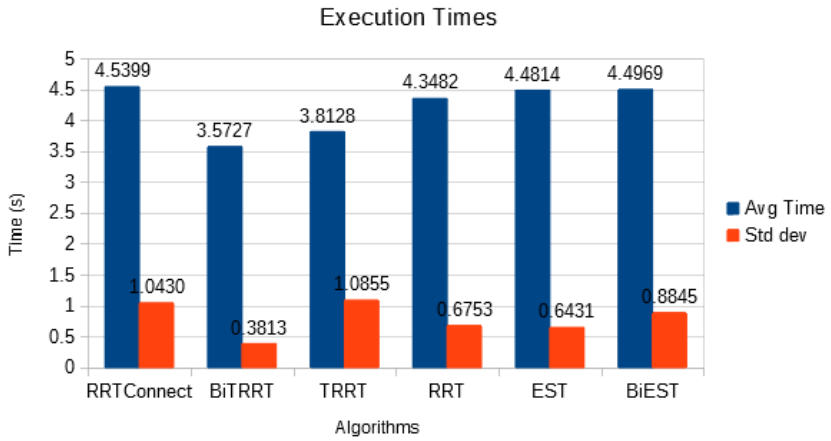


Figure 4.18 – Comparison of average execution times of the algorithms.

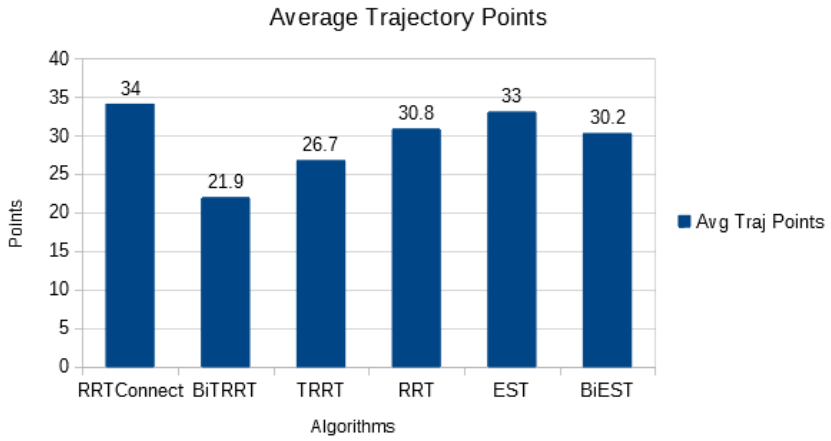


Figure 4.19 – Comparison of average amount of generated waypoints.

iterations for each algorithm, but instead of considering just the computation time (Figure 4.18), we also took into account the amount of generated waypoints (Figure 4.19). Also, the mannequin was placed in the middle of the direct trajectory for it to be avoided in the computation, to test each algorithm's capabilities in planning around it. This also allowed us to see how consistent each algorithm's behavior was.

Analysis on different planning algorithms

Following the experiments introduced in section 4.4.3, we ran two different experiments. The first one intended in determining which one of the 12 available planning algorithms were better suited to perform the path planning within the system. As Figure 4.16 and Figure 4.17 shows. Half of the algorithms included in the MoveIt plungins have long planning times, so that led us to reduce the total amount of algorithms to consider as usable to a new total of 6 algorithms. This ones being *BiEST*, *EST*, *RRT*, *RRTConnect*, *TRRT* and *BiTRRT*. With *BiTRRT* showing the lowest average total computation times, having in 4.17 an average of 0.5799 seconds for that particular trajectory. Followed by *RRTConnect* with 0.7332 seconds. While in 4.16 both of them are also the fastest algorithms with 0.5559 and 0.5254 seconds respectively. In this last, *RRTConnect* presented a slightly faster time, this two were the ones that showed the best times throughout all the 9 tested trajectories.

Due to this, in order to deepen the study and choice of the algorithms to use, we defined the second set of experiments, which considered the execution times of the trajectories generated by the algorithms. In this case, shown in Figure 4.18 we can see that the *BiTRRT* algorithm generally reaches the goal faster for the trajectory represented in the graph with an average of 3.5727 seconds, while *RRTConnect* showed and average of 4.5399 seconds, this being 0.9674 seconds more in execution time. This behavior was also similar to the other trajectories tested in the experiment, showing that even if both algorithms take more or less the same time in finding a solution, *BiTRRT* has faster execution times.

This is also reflected in Figure 4.19, where for the *BiTRRT* algorithm the trajectories computed where more "consistent", obtaining similar trajectories every time (both algorithms are based in RRT, which is based in random exploration). In the case of *RRTConnect*, the solutions found were less optimal, sometimes generating longer trajectories with high deviations of what was considered the "optimal path".This is shown by the average amount of waypoints for each algorithm, where *RRTConnect* generated in average 34 waypoints, while *BiTRRT* geneated 21.9 waypoints in average. This results were consistent as well in all the different trajectories tested, backing up the conclusion over the deviation from the "optimal path" for the *RRTConnect* algorithm, as an average of more waypoints and longer execution times are a reflection of longer trajectories. Leading us to pick the *BiTRRT* algorithm as our path planning algorithm for the implementation of this project.

4.4.4 Description of Unity's virtual environment

In parallel to the development of the project, and to further explain the developed implementation, it is important to clarify how it will merge into the project. The system will receive the desired goal configuration that is going to be the q_{goal} intended for the planning algorithm, starting from the current configuration q_{init}. This goal selection is being made in Unity by a *Point selection algorithm* which determines the point of interaction that the user intends to reach, will be dealt in detail in next Chapter 5.

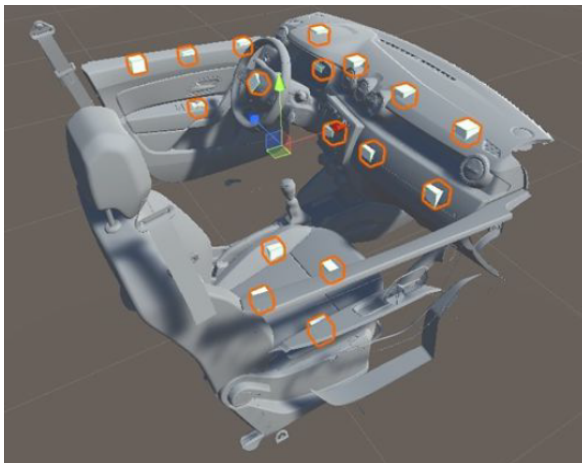


Figure 4.20 – Unity VR system and representation of interaction points

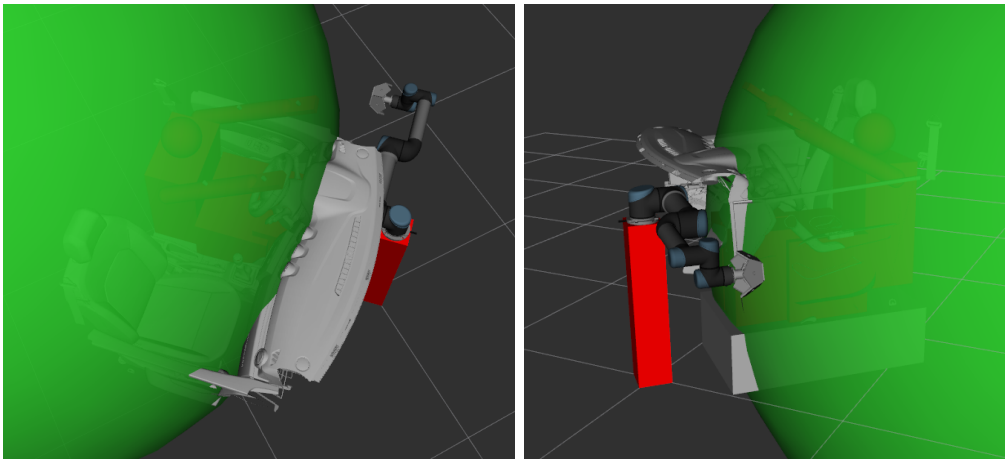


Figure 4.21 – Representation of the user's effective workspace as a sphere

4.4.5 Different mobility schemes

Based on the Unity information, two different motion or mobility schemes and scenarios have been proposed depending on the nature of the task we want to achieve at the moment. One for which no interaction with the user is required, and another one for when it is. These two scenarios have their own environment to consider, presenting in general two different behaviors.

Movement outside person's workspace

The first scenario is where a distinction for velocity zones is performed based on the persons reach/workspace, the region is divided with a plane that represents the user's reach. Based on the same idea, we represented the mannequin's effective workspace as a sphere surrounding the model.

The mobility scheme consists of alternating from different *"safe positions"*, called in this way as they are interaction points outside the user's reach, meaning there's no need in constraining the robot's velocities. Due to this, the motion from one point to another just needs to take into consideration the defined sphere, as we do not want to "collide" with it.

Following the idea, we performed a computation of all the existing trajectories between the different *"safe positions"* and stored them in a data file (assuming here that the robot starts at a point outside to the person's workspace and want to go to a another point outside the person's workspace). This allows us to perform offline path planning, and then, on run-time, based on the initial and desired goal, we can access the pre-computed trajectories to directly execute them, removing the computation time that would take otherwise by performing online planning.

The algorithm 7 show the trajectory storage.

Algorithm 7 Trajectory computation and storage.

Input: Number of points no_p. A counter for start point i. A counter for final point j

$start\_name[no_p] \leftarrow$ ▷ Store id of the points
$final\_name[no_p] \leftarrow init\_names[no_p]$ ▷ Same id's as we are iterating through all points
for $i < 0 \; ; i < no_p \; ; i++$ **do**
  for $j < 0 \; ; j < no_p \; ; j++$ **do**
    if $i \neq j$ **then**
      Ø← Plan_and_Exec_to($points[i]$) ▷ Move to initial point of the plan
      $plan\_array[i][j] \leftarrow$ Plan_and_Exec_to($points[j]$) ▷ Move to desired point and keep the planned trajectory
    end if
  end for
end for
▷ Store all as a structured message
for $i < 0 \; ; i < no_p \; ; i++$ **do**
  for $j < 0 \; ; j < no_p \; ; j++$ **do**
    if $i \neq j$ **then**
      $init\_pos\_id \leftarrow start\_name[i]$
      $goal\_pos\_id \leftarrow final\_name[j]$
      $plan \leftarrow plan\_array[i][j]$
    end if
  end for
end for

Subsequently, the second part of the scheme consists in loading up the pre-recorded data and being able to use it on demand. Expecting as an input just the desired position to reach. This is explained in algorithm 8.

Differently, in this research topic spherical obstacle (Figure 4.21) is used to divide the two zones instead of a plane (Figure 4.4) , as it allows more flexibility for the planning group to consider more configurations when computing the path between points. Also allowed to find more achievable paths for the robot to follow, as it allowed to perform easier motions for the robot.

Movement inside person's workspace

The second mobility scheme was proposed for the scenario where we need to go inside the user's workspace, this means that the motions have to take into account the user's model to avoid colliding with it. Another thing to consider is that these motions need to have their velocity constrained, to ensure safety aspect.

Algorithm 8 Trajectory upload and execution.

Input: A desired frame to go to des_frame. A home pose $home$. Number of elements no_e. Initial positions $init\_pos\_id$. Goal positions $goal\_pos\_id$. Planned trajectories $plan$. A counter i.

1: **for** $i < 0$; $i < no_e$; $i++$ **do**
2: ▷ Extract the data from the file
3: $start\_name[i] \leftarrow init\_pos\_id[i]$
4: $final\_name[i] \leftarrow init\_names[i]$ ▷ Same id's as we are iterating through all points
5: $plan\_array[i] \leftarrow plan[i]$
6: **end for**
7: Ø← Plan_and_Exec_to($home$) ▷ Move to home pose
8: ▷ Reference to home position as current
9: $init\_frame \leftarrow$ "$home$"
10: $aux\_des\_frame \leftarrow$ "$home$"
11: **while** running **do**
12: **if** $des\_frame == init\_frame$ **then** ▷ The robot is in position.
13: **else**
14: $aux\_des\_frame = des\_frame$ ▷ Update the desired position
15: **for** $i < 0$; $i < no_e$; $i++$ **do** ▷ Search in the list of plans the one that matches the init and final frames
16: **if** $start\_name[i] == init\_frame) \ \& \ (final\_name[i] == aux\_des\_frame)$ **then**
17: $execute(plan\_array[i])$
18: $init\_frame \leftarrow aux\_des\_frame$
19: **end if**
20: **end for**
21: **end if**
22: **end while**

Unlike in the first scheme, in this case, the environment consists of moving obstacles, which requires a constant update of the scene and constant tracking of the objects within it. For this reason, we used the frames of the mannequin model to obtain its current positions and orientations to be able to track their movement and link it to the objects spawned in the scene.

We also need to be able to determine if a computed plan is going to enter in a collision or not, which needs to consider multiple aspects. First, based on the computed path towards the desired goal, we check if during the execution of the plan the path remains valid. This is done by checking for all the computed waypoints of the path if the respective configurations are currently in a collision with any other object present in the scene. If it is free of collisions, then we proceed with the execution. In the case of a collision present in any of the remaining states of the non-executed path, we instruct the robot to stop the execution of the computed path and re-plan again based on the updated scene information.

To test this, we performed an initial implementation of the moving obstacle principle, to which we planned to a desired joint position, and spawned an obstacle in the middle of the trajectory while mid-execution. Then, thanks to the path validity checking we can detect that an object is in collision with the planned path, so we instruct the robot to stop the current execution and re-plan towards the same goal, taking into account the refreshed planning scene.

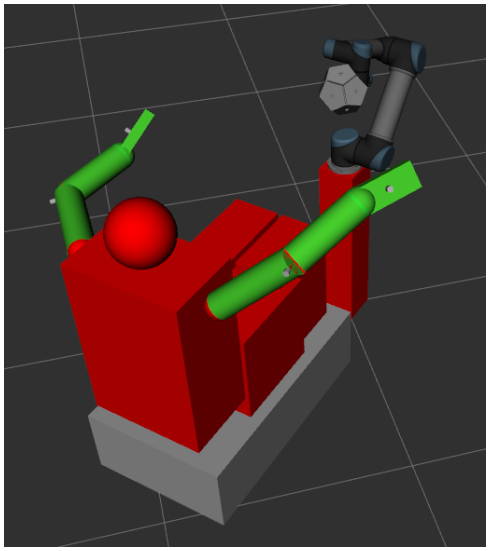
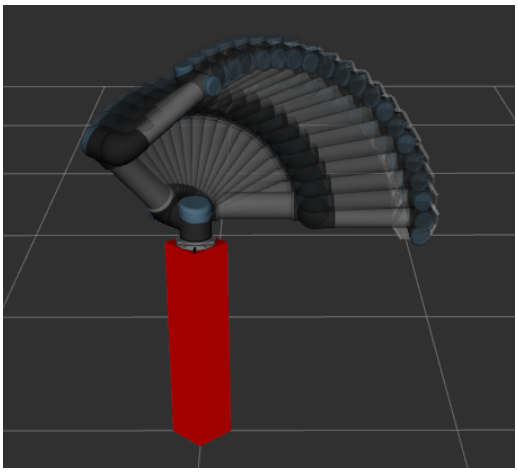
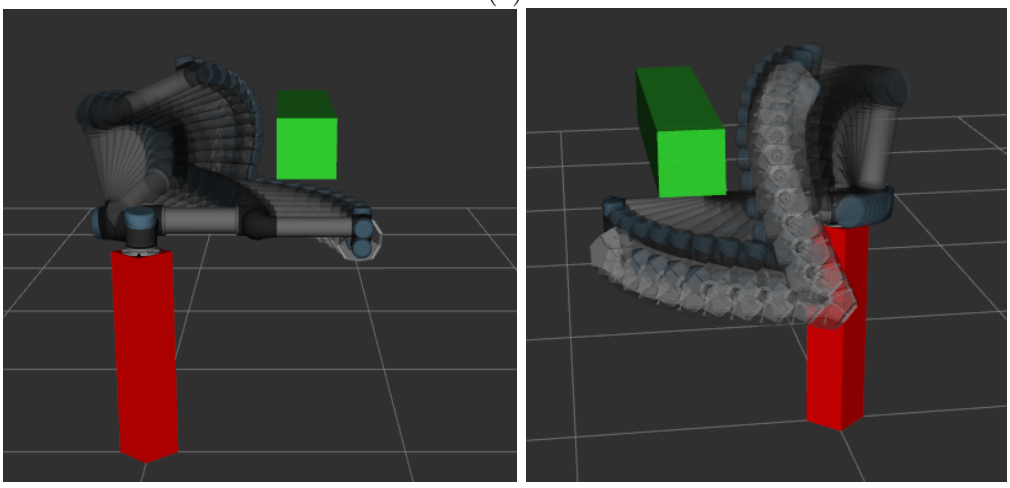


Figure 4.22 – Representation of dynamic obstacles using the user model.



(a)



(b)

Figure 4.23 – (a) Original planned path. (b) Re-planned path from detected collision

This work is intended to be extrapolated to work along with the mannequin model and its kinematics as mentioned in section 3.1. So it is capable to take into account the moving mannequin in the environment as an obstacle to avoid.

Analysis on mobility outside workspace

One of the advantages on this implementation, in section 4.4.5 is that by being capable of directly identify which trajectory we want to execute, we can access to the pre-computed plan and we perform directly the execution of said trajectory. As we are located outside the workspace of the user, there's no need to constantly plan a trajectory as once we already took into account the sphere defined in Figure 4.21 we already ensured that the trajectory will be safe. This normalizes the execution times as they are now constant.

One limitation this implementation has is that, given the nature of performing pre-recorded trajectories, the system needs to fully execute each one of the trajectories that he/she performs. Contrary to other scenario where trajectories can be re-planned as in Figure 4.23. Nonetheless, this is not a strong limitation in the system, as the prediction (will be dealt in next chapter) of the "interaction points" and "safe positions" done in Unity helps to overcome the re-planning situation.

Analysis on mobility within workspace

This movement scheme is fundamental at the moment of interacting with the user, and thanks to having defined obstacles which have been linked to the mannequin model we can take into account this elements as the objects to avoid while performing any type of planning. Another important aspect to take into account within this scenario, is that the goals received for the system can change rapidly, so for this case we want to be able to do re-planning during any plan's execution period. Allowing the system to be more fluid in the motions as well.

On the limitations it presents is that during the development, even though that scene is being constantly updated and we can know in real time the refreshed position of the mannequin (Figure 4.22), the development of this scheme has not been fully implemented, as we are able to re-plan for dynamically appearing objects as shown in Figure 4.23, but we are not able yet to check the path validity in an iterative way, limiting the reaction and interaction with the mannequin model.

4.5 Conclusion

A system architecture to communicate between the multiple systems (Unity in Windows , ROS in Ubuntu and robot system) was established. A calibration setup to connect components of the physical and virtual environment in ROS was defined. A simple test was performed to verify the conversion of coordinate systems between Unity and ROS.

Four planning algorithms were studied to plan the trajectories. These algorithms were compared for a give test case and analyzed. Finally *RRT* and *RRT** methods and alternatives, they were known to be good path planning methods, with the limitations that they have high computation demands and memory.

Another limitation, even though we are able to re-plan for dynamically appearing objects, but we are not able to check the path validity in an iterative way, which limits the reaction

and interaction with the mannequin model. To counter these drawbacks, we plan to have precomputed trajectories and only wish to access them based on the user intention prediction (explained in next Chapter).

In the next chapter we compare several algorithms for detection of user intention. To have same data as input for all prediction algorithms the analysis will be done without motion of the robot and recording the user motion. Precomputed trajectories will be used to evaluate the intention prediction algorithm as the planning computation time and execution time will be constant.

Chapter 5

Prediction of user intention

5.1 Introduction

This chapter proposes a set of different movement strategies for the robot to be as fast as possible in the contact zone while guaranteeing safety. This work uses the concept of predicting the user's intention through his/her gaze direction and the position of his/her dominant hand (the one touching the object) and safe points outside the human workspace. Experiments are done and analyzed with a Pareto front with a UR-5 robot, an HTC Vive tracker system for an industrial application involving the analysis of materials in the interior of a car.

When the user uses HMD vision interfaces and has to perform haptic evaluations, he/she no longer sees the real scene, but only a virtual world. His/her physical reference points quickly disappear except for objects he/she touches such as his/her seat and the floor.

Through the user's gaze and hand movements, as well as the position of areas to be studied, it is possible to predict the tasks that the user will perform. Eye-hand coordination is a fundamental behavior that humans use to interact with the world [55, 66, 92]. The head movement facilitates subsequent gaze shifts toward the future position of the hand to guide object manipulations, thus leading to a strong correlation between head and hand movement parameters [102, 123, 124]. The purpose of this study is to ensure that the robot end-effector will be available for intermittent contact in complete safety when the human hand is close to the surface to touch. An industrial robot can perform powerful and fast movements that can be dangerous for the humans around it. Involuntary contact between the robot and humans is a threat. This is particularly important in a virtual reality context where humans equipped with an HMD will not be able to anticipate the robot's movements.

Today, more than ever, humans work closely with robots. In the case of intermittent contact interface ICI, contact is inevitable between humans and robots. Cobots are best suited to such a scenario, but in terms of human safety, accident prevention can always be improved [25]. These robots are designed to work at limited speeds during potential contacts. Moreover, it must be ensured that the desired contact with the robot during interaction will not result in a necessary restart of the robot after a safety stop [82].

The main contributions presented in this chapter can be summarized as follows:

— A set of strategies to predict the user intention.

— The modulation of the robot's speed according to its location in relation to human.

— A user-based analysis of the proposed strategies.

A motion capture system based on HTC Vive-trackers is used to know the position of the body and especially the hand used for interaction as well as the position of the chair and the

robot [129] (Figure 5.1). The prop can carry six different materials. The robot is fixed on a 80 cm high table and the user sits on a seat 60 cm above the floor. The placement of the robot in the scene has been chosen to be able to reach all the places where the user's hand will want to have haptic interaction with the robot's probe[50].

Figure 5.1 – The complete system setup for human-robot interaction.

A virtual model under the Unity 18.4 LTS software represents the fixed objects in the environment. The position of moving objects is known, the user position thanks to HTC trackers located on the hands, and its seat and robot location thanks to encoders on the motors.

The outline of this chapter is as follows. Section 5.2 introduces the proposed model and methodology. Section 5.3 presents different regions of robot motion. Section 5.4 explains all the proposed strategies. Section 5.5 presents the user study carried out to evaluate the introduced strategies. Results and insights from the study are also discussed. Finally, general remarks and conclusions are commented in Section 5.6.

5.2 Human intention prediction

5.2.1 Detection of target

The robot needs to anticipate human's future actions and act accordingly while performing collaborative tasks. In most human-robot collaboration systems, the motion of robots is based on some predefined programs, which are task-based. However, most tasks are highly complex and it is difficult to redefine a complete set of instructions for such situations. In such tasks, the role of the robot should be changed from purely automated machines to autonomous companions. Previous works relied on supervised learning methods to build models of human motion, which

relied on understanding the environment, offline training or manual labeling, adapt to new people, and motion styles.

Human intention is mainly expressed through the behavior of humans and the objects they interact with. Most of the current research on human intention prediction just focuses on action classification, in which the human action is classified into several categories, such as running, walking, jumping [41] which is inadequate for accurate inference of human intention in human-robot collaboration.

We propose an HRI framework that combines hand motion with gaze direction to build models on the fly, which predict human intention in virtual reality and move the robot to the required position in a virtual space without offline training.

5.2.2 Proposed model

The aim of the work is for the human to make contact with different parts of the car, in a design phase where only a virtual model exists, to be able to assess the quality of the materials. The areas to be explored are limited, driver's door, passenger seat, dashboard, touchpad. Depending on where the human wants to touch, the robot must position itself so that the human can touch the appropriate material placed on the probe. The probe has a certain surface area, so a limited number of Regions of Interest (ROI) in the car have been defined that the robot will have to reach to allow contact with the human. The set of 18 ROI considered is described in Section 5.2.3. The objective is therefore to determine as soon as possible the ROI that the user wants to reach and even more so that the robot's probe is positioned as soon as possible on this ROI. If the probe arrives before the human, the human will be able to make contact without being aware that he/she is in a virtual world, otherwise the waiting time before making contact should be as short as possible.

The major elements involved in our approach are summarized in Figure 5.2. Measurements of the pose of the hand and the gaze direction via the orientation of the HMD are used to select an ROI where the human hand will touch the prop of the robot.

It should be noted that as the objective is that the robot arrives at the target as soon as possible, several strategies are possible and can be combined:

— Detect the target at the earliest,
— Move the robot as soon as possible in the right direction, even if the final target is not yet known,
— Move the robot as quickly as possible.

As we are in a cobotic context with a human locked in a virtual world that does not see the robot (the robot can also be visualized in the virtual mode but the immersion will be less), safety is a priority. A description of the methods implemented to have a fast speed of movement of the robot and ensure safety will be discussed in Section 5.3.

5.2.3 Scene information

From the model of the car in Unity virtual reality software, we defined the ROI the user is to interact with. Each ROI is represented as a capsule placed at the center of the surface. For each surface, the desired orientation of the probe is defined. We have defined 18 ROI to be studied in the car (Figure 5.3). They are located as follows:

- Four capsules on the door,
- Four capsules on the chair,
- Four capsules on the dash board,

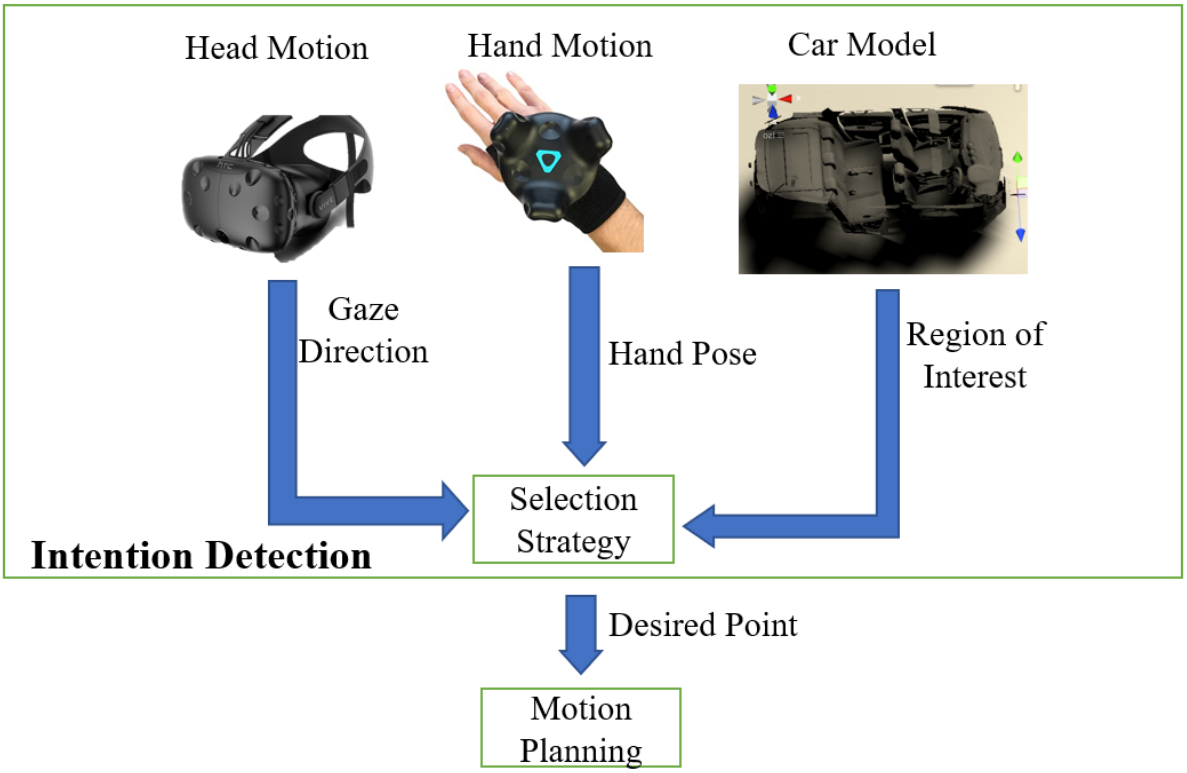


Figure 5.2 – Diagram of the inputs used to choose a robot movement strategy.

- One capsule on steering wheel,
- One capsule on touch pad,
- Three capsules on glove compartment,
- One capsule on speedometer.

5.3 Safe and fast motion

When the target is defined, the robot must be moved. For this, a series of trajectories that avoid obstacles have been defined (see section 5.3.1) between the point of interest and/or safe point. While the robot is moving, a new point of interest can be defined. One could stop the robot's movement and recalculate an obstacle-free trajectory online. However, in order to avoid wasting time in this calculation, predefined trajectories have no calculation time, so the robot is let to perform its movement. And it will take into account the new target at the end of its movement.

5.3.1 Cobot motion

The robot will navigate between a finite number of points which are our ROI. However, the movements must ensure that collisions with humans are avoided. To do this, we will generate offline robot movements that ensure that no part of the robot enters an area encompassing the human at rest in the driver's seat. The area to be avoided is composed of a sphere and is illustrated in Figure 5.4. The dimension of the sphere covers the human head and torso and

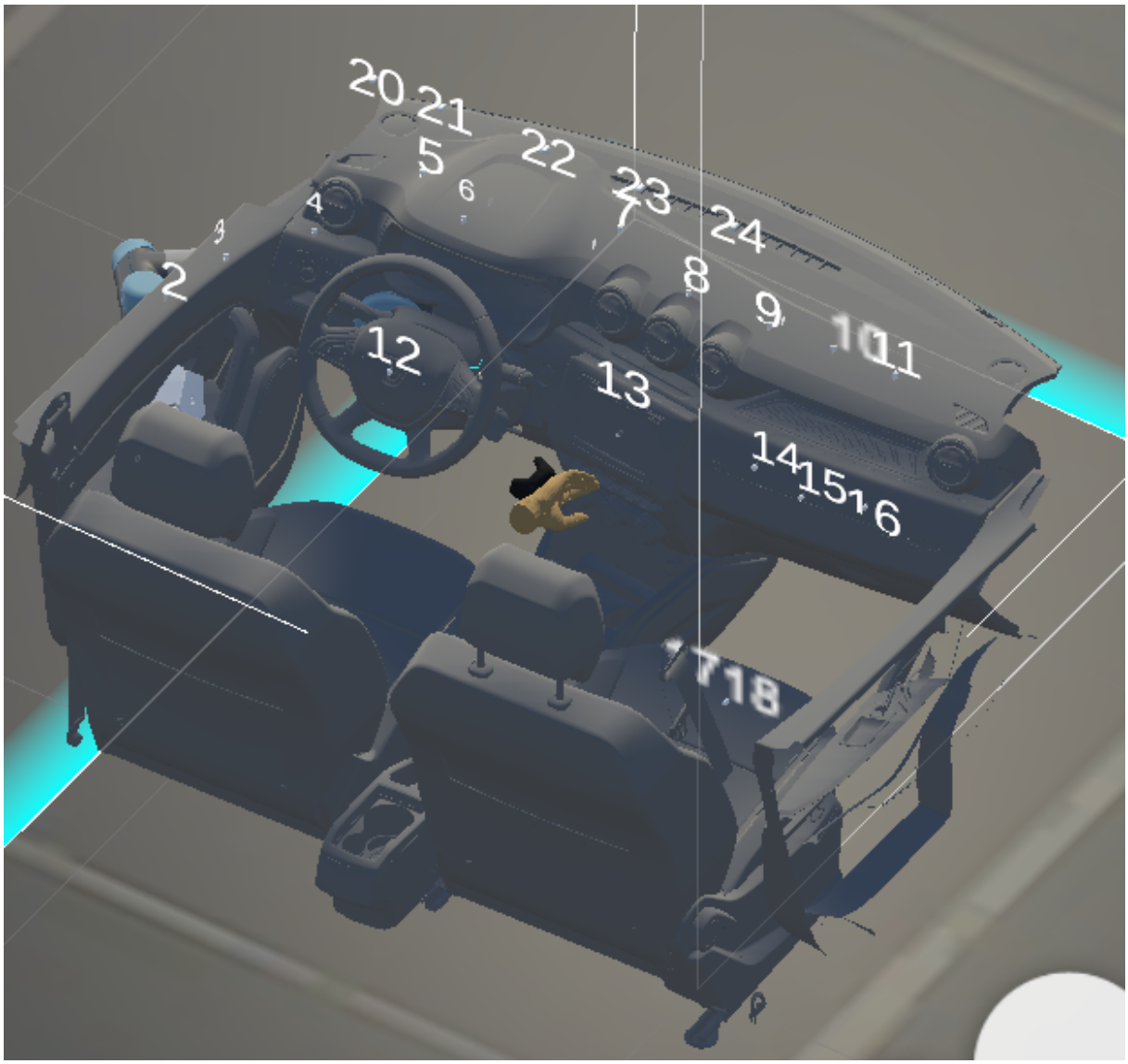


Figure 5.3 – Location of the ROI 1 to 18 inside the car and safe-points 20 to 24.

part of the arm, but the hands can be outside since they must be able to reach the ROI. For this, we need to construct 18×17 off-line trajectories that we will assemble in line according to the ROI detected to accomplish the task. These trajectories are close to the human, they are realized with a maximum speed of 0.25m/s to ensure the use of the UR-5 cobot according to the ISO standard for human-robot collaboration [125]. This condition guarantees that a possible collision with the human will not hurt him.

5.3.2 Velocity zones

The robot must be moved closer to the target point to prepare for the interaction. The movement must be fast so that the robot has arrived before the human and thus avoid unpleasant waiting, but the maximum speed of the robot must be limited for safety reasons. Figure 5.5 shows the scene of the VR environment, it consists of the car interior and user model.

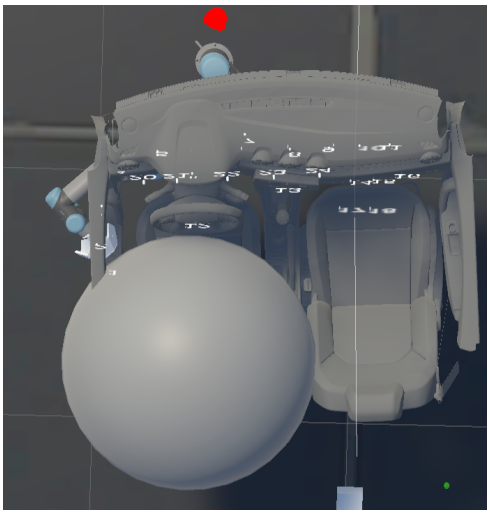


Figure 5.4 – In the definition of the robot motion to joint the ROI, the sphere that represents the user occupancy zone is avoided.

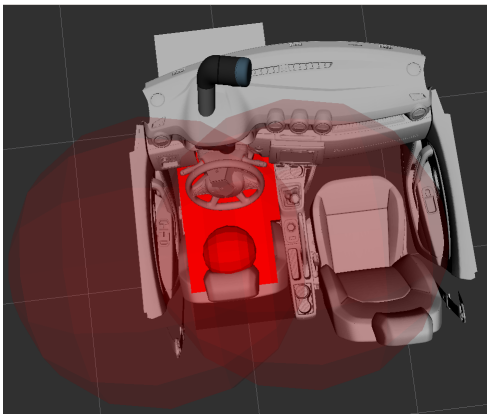


Figure 5.5 – Car interior and user workspace in Rviz.

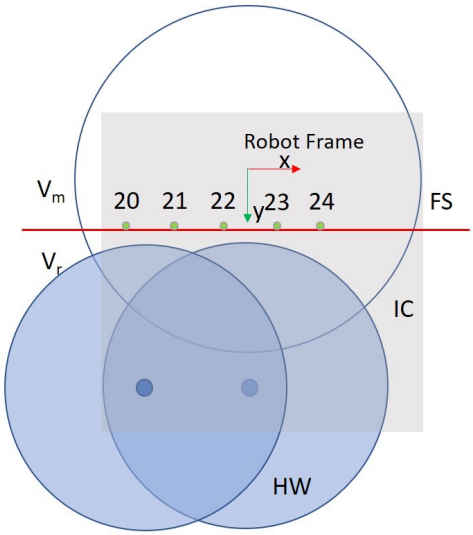


Figure 5.6 – The spaces defining the robot velocity. The two blue spheres described the human workspace when seated. The grey part shows the car model. The transparent sphere is the robot's working area. The red line delimits an area where the speed of the robot can be higher because there is no risk of collision with the human.

Based on this, we distinguish three velocity zones:

- The human workspace (HW), is defined as two spheres whose radius is the size of the arm centered on the shoulders of the mannequin. This workspace will evolve according to the movements of the human. We could also consider a constant space if we limit the realistic movements of the torso. This space is represented by blue circles in Figure 5.6.
- The inside of the car (IC): this space delimits the area where we know the human must move. Even if the Unity model is complex, this zone can be approximated by a larger simple region that includes the real interior of the car. The gray rectangle, in general, represents the entire Unity model and we define a plane, depicted by the red line in Figure 5.6, that separates the region that can be reached by the user.
- The free space (FS) cannot contain points that are in the HW. We can have a certain safety margin to define this zone. In our example, this zone is simply limited by a plane represented in red in Figure 5.6.

The limit on the robot velocity is chosen according to the space:

- When the robot moves in FS, it can do so at maximum speed V_m (all parts of the robot are in FS),
- When the robot moves outside of FS, it must move at reduced speed V_r,

The speeds are chosen such as $V_m \geq V_r \geq 0$. The different spaces are shown in Figure 5.6. The blue transparent circle is the robot's workspace, two blue-filled circles are the workspace of the user's hands. The grey rectangle is the complete interior model of the car and the red line is the plane that we use to differentiate the reachable and unreachable parts of the car by the user.

5.3.3 Velocity profiles

To ensure safety and also have better response time we defined two velocity profiles of $V_r = 0.25\ m/s$ and $V_m = 4\ m/s$ based on the zones defined above. To illustrate the idea, we

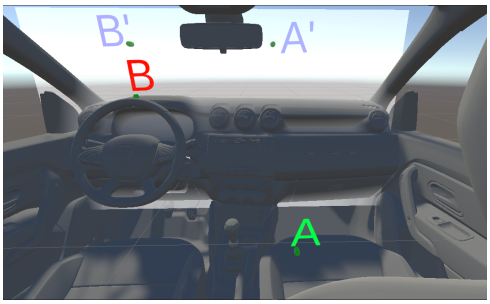


Figure 5.7 – Illustration for the comparison of motion using safe-points.

devise the scene as shown in Figure 5.7. We define four points:

- Two points A' and B' are on the plane boundary, these are in the FS, and fast motion between these points can be produced. For the application studied in the paper, the capsule denoted 20 to 25 are in the FS.
- Two points B and A are inside the plane boundary, one point on the dashboard and another on the passenger's seat. These points play the same role as the ROI 1 to 18.

We analyze two different scenarios based on different velocity combinations.

- *The shortest way*: Knowing the target point, the robot moves towards it, and adapts its speed according to the spaces it crosses. In the studied example, it goes directly from A to B with a velocity of less than $0.25m/s$.
- *Safe-points*: We use of safe-points to keep the robot's speed high. In the studied example, the points A' and B' belongs on the plane that limits FS. The robot goes from A to A' with a maximal velocity less than $0.25m/s$, then from A' to B' with a maximal velocity less than $4m/s$, B' to B with a maximal velocity less than $0.25m/s$.

The movement of the robot inside the car should be performed at reduced speeds for safety reasons. In some cases, when the desired point is far away from the user space, it takes longer to reach it due to the low speed. In such situations, we use the via-points on plane boundary in the FS, called safe-points, between which the robot can move at high speed. The path is longer but its execution can be faster.

New trajectories are calculated off-line to connect the ROIs and the safe points with the two motion speeds.

5.3.4 Safe-points

As we have shown in the previous section, the interest to move the robot in FS is to speed up the robot movements. Five points on the plane boundary of FS have been defined SP_{20}, SP_{21}, SP_{22}, SP_{23}, SP_{24} to be used for this purpose.

The interest of moving the robot in the FS is also to increase the safety of the operator by moving the robot away from the human. To quantify this notion of safety, we will define an average distance between the robot's end-effector and the sphere encompassing the human's torso shown in Figure 5.4. The higher this distance, the safer the human/robot interaction will be. This distance is calculated in an approximate way from the points of passage of the robot (SP_1,...SP_{24}), by making the hypothesis of a straight line displacement at constant speed between the points (defined as the distance between the points divided by the duration of the displacement).

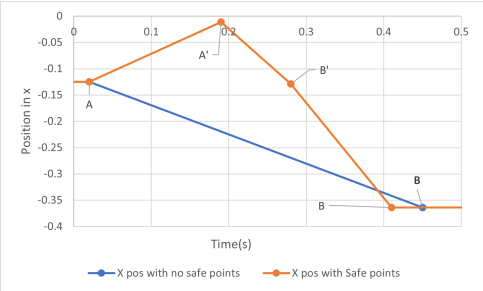


Figure 5.8 – Comparison of motion through safe-points and without safe-points.

$$d_s = \frac{\sum_{i=1}^{N}(\|p_i - C\| - R)}{N} \tag{5.1}$$

Where the robot motion is sampled in N instant, p_i is the coordinate vector of the end-effector of the robot for the sample i, C is the coordinates vector of the center of the sphere shown in Figure 5.4 and R is its radius.

Considering the user safety and robot velocity it is of interest to pass through points on the plane boundary in FS, when we have long robot trajectories to make. In the next part of the study we will show that this can also be useful when a movement is initiated by the human by hand and gaze but without knowing yet where the human will stop. Placing the robot on one of the safe points SP_{20}, SP_{21}, SP_{22}, SP_{23}, SP_{24} will allow the robot to get closer to the goal more quickly.

5.3.5 Comparison of motion with or without safe-points

An example is illustrated in Figure 5.8 where $A = P_{17}$ and $B = P_5$, $A' = SP_{24}$ and $B' = SP_{21}$. The points A' and B' are positioned on the plane that divides the two different velocity zones.

We compared the time taken by the robot to move between two points, taking into account the presence of safe points and without them.

The results in Figure 5.8 show the position of points in X coordinate with respect to time. The Figure 5.8 is a representation to show the point and at what time the robot reaches that point. From the recorded trajectories of the robot to reach the points, the X-coordinates of the robot are plotted at the beginning and end of the trajectory (and connected by a straight line) against time in Figure 5.8. The direct motion is shown in blue, when the motion with intermediate safe point is shown in red. It proves that by passing through safe-points A' and B', the robot takes less time than going directly.

By traveling through safe points, the robot starts from point A and moves through A', B' at a higher velocity and then to the final point B. The total time taken to reach the final point was 0.41s. For the motion inside the car, the robot arrives at the final point after 0.45s.

5.4 Proposed strategies

We will study and compare four strategies that integrate the position of the hand, the direction of the gaze, and the use of safe-point to efficiently move the robot to one of the ROI that the human wants to reach.

5.4.1 Strategy A: Hand position

As the user's objective is to touch with his/her hand the ROI, the first and the simplest strategy proposed is to consider that the point to be reached is the closest to the hand position. The strategy is presented in Figure 5.9, in the example the selected point is P_2. The main advantage is the simplicity of the approach. For the search for the nearest point, the k-d tree structure is used to find the nearest point of the hand. An implementation of this algorithm for VR environment is done in [96]. The strategy is summarized in Algorithm 9.

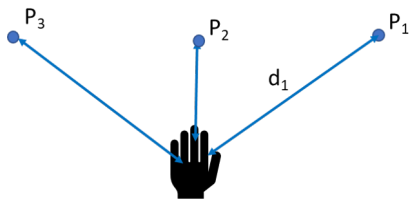


Figure 5.9 – Pictorial representation of Strategy A.

Algorithm 9 Strategy A: Predictions with hand.

Input: Hand position $P_h \in R^3$
Output: Nearest Point P in the set of P_i, $i = 1...18$.

```
Build a k-d tree for all points P_i in the scene.

function STA(P_h)
    Using hand pose as a query point q, return nearest point P from the k-d tree.
    return P
end function
```

The main characteristics of the approach are:

1. The point is detected only when the human has almost reached the points;
2. If two points are equidistant, a prediction fluctuation can occur with small changes in hand motion;
3. During the movement of the hand, intermediate points can be detected, which will allow the robot to start its movement before the desired end point is detected.

We used a proximity search, which is an optimization problem of finding a point in a closed set that is closest to a given point. Closeness is defined by a dissimilarity function such that the dissimilar the objects, the larger the function values.

Problem definition

Given a set P of our interest points in a 3D space D, and a query point q which represents the user's dominant hand, we find the closest point P to q. This problem can be generalized as a k-nearest neighbour (kNN) query where we have to find the k closest points where k $\in Z^+$. Implementing the kNN is normally done by computing the distances from q to all elements in P. However, this method is computationally intensive for a large number of data points. We used a k-d tree for the nearest neighbor search proposed by [8, 44].

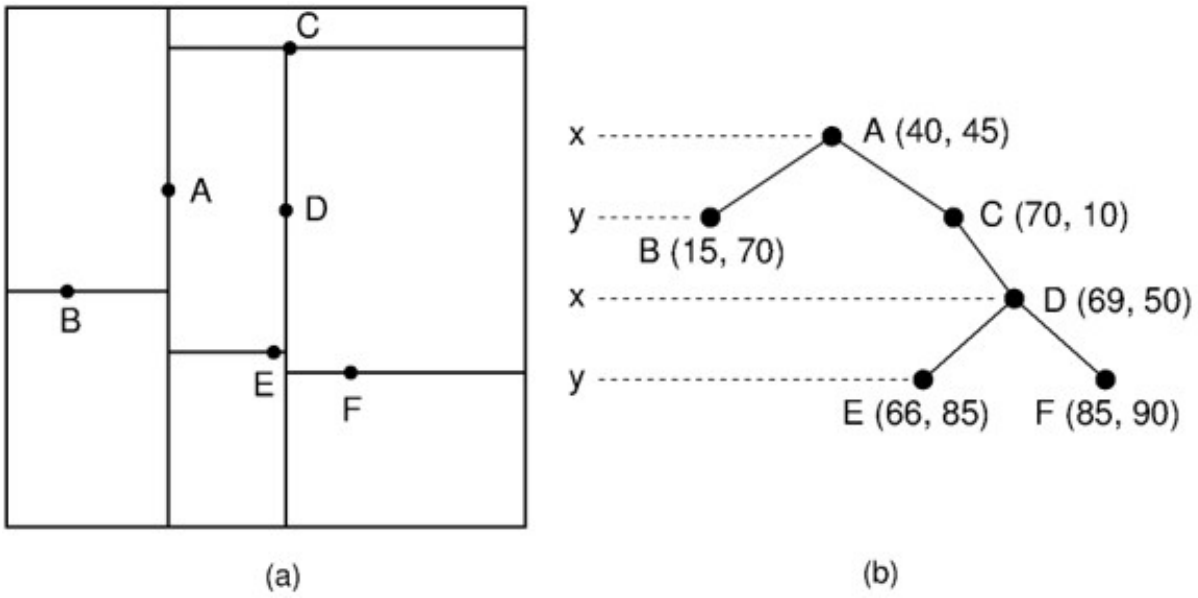


Figure 5.10 – k-d tree (a) k-d tree decomposition for point set and (b) The resulting k-d tree.

Formally a k-d tree is a balanced binary tree for a set of data points $p_1, p_2, ..., p_n \in P$ where the root corresponds to all points and its two children represent almost equal-sized subsets of P. Every leaf corresponds to a k-dimensional point and every non-leaf node is a splitting point generating a hyperplane that splits points into subsets in a level-wise manner. Points to the left of this hyperplane are represented by the left sub-tree of the node and points to the right are represented by the right sub-tree. For a given node p at level i, the points associated with p are split into two halves by resorting to the median in dimension i mod k. Such that the point inserted in the tree at each step is the one, which has the median coordinate in the direction considered [8]. However, splitting rules may vary. The recursive construction ends as soon as a node p corresponds to a singleton or to a set of predefined sizes.

Nearest neighbor search using the k-d tree

Given a point q, find the point p in the data set P that is closest to q. This can be done with the three following steps :

— Step 1: We defined the points representing the position and orientation of the surface for each of the ROI in the car model.

— Step 2: Build a k-d tree to store the positions for all the cubes.

— Step 3: Using the hand position as a query, we find the closest point to the hand from the k-d tree.

The algorithm 10 describes precisely these above steps.

Algorithm 10 Nearest neighbor search.

Input: k-d tree root node, query point q .
Output: Nearest node p.

1: Start from the root node.
2: Move down the tree recursively following the same procedure as it would be for the insertion of point q in the tree.
3: Once a leaf node is reached, save the leaf as the current best point p.
4: Rewind the recursion tree, and
5: For each node v, if $d(q, p) > d(q, v)$, then the current node is $v = p$, where d is a distance metric.
6: Check if there could be yet better points on the other side of the subtree by checking if neighboring boxes potentially contain points that are closer to q as the current best candidate (using the median values).
7: In case a point might be closer, recurse to the sub-tree that has not yet been visited.
8: If there could be, move down again on the other side of the sub-tree. Otherwise, go up another level.

Drawbacks of the above approach

The above approach has the following shortfalls:

1. The accuracy decreases when the hand is close to the midpoint of any two points. In this case, the difference in the distance between each point and the hand is very small and such that a slight displacement of the hand results in the inappropriate motion of the robot to any of the nearest points.
2. Unnecessary and Involuntary hand movements. Due to human nature, the user can move their hands involuntarily without the intention to interact with any objects in the space. In this case, the algorithm would still move the robot to the best point according to the minimum distance.

To overcome the above shortfalls, we decided to include the head gaze in the model such that a predicted point is considered valid and intentional. If the user was gazing in the direction of the object the hand interacts with. This is because when humans reach to grasp an object, they look at the target first, then bring the hand to the center of gaze as the object [88, 114, 102, 123].

Improved approach

The improved approach involves information extraction and definition of ROI from car model, k-d tree construction for each ROI and Hand pose, nearest neighbor search, and Hand - head gaze coordination to determine the best point. A 5 step process :

— Step 1: We defined the points representing the position and orientation of the surface for each of the ROI in the car model.

— Step 2: Build a k-d tree to store the positions for all the cubes.

— Step 3: Using the hand position as a query, we find the n_p points closest to the hand from the k-d tree. Contrary to section 5.4.1, we do not define directly the closest point

but we select n_p candidate points. Depending on experiments n_p can be two or four.

— Step 4 : From the n_p candidate points, we select only the n_{pg} points. belonging to the view frustum of the HMD.

— Step 5: Among these n_{pg} points, the closest to the gaze direction is selected as predicted contact point between the user and the robot end effector.

Theses steps are commented in the Algorithm 11.

Algorithm 11 Predictions with head gaze.

Input: Scene information, head gaze direction.
Output: position and orientation of desired point.

1: Cubes representing the surface and orientation of regions of interest.
2: Build a k-d tree for all points in the scene.
3: Using the hand pose as a query point q, return the nearest n_p-points from the k-d tree.
4: For each of the n_p points, we select only the n_{pg} points which lie within the view frustum of the HMD.
5: find the gaze direction as a unit vector from the central point of the eyes and draw a ray in the gaze direction.
6: calculate the distance of each point in n_{pg} from the ray and return the position of the nearest point.

Experimentation

We conducted experiments to analyse the motion of the robot in response to hand motion in two approaches by moving the hand between the two points, with the following objectives:

1. Midpoint test: To show the response and error in robot motion when the hand is close to the midpoint. For this, we consider two points, which are 1.3 meters apart. Placing the hand tracker at the midpoint and displacing it by 2 cm results in the robot going to extreme points
2. Involuntary and Unintended Hand Motion Test: To verify that the robot moves only when the hand and head gaze are in the same direction. A hand motion to areas outside human vision is not sufficient to move the robot.

Results

In the first approach, we used the model without data from VR HMD, we observed that by moving the hand approximately 1.5 mm from the midpoint, the robot responds by moving to the other closest point. Figure 5.11 shows the displacement of the hand. The corresponding motion of the robot is shown in Figure 5.12. It can be seen that Robot TCP moves to the extreme points ever-time there is a small displacement in the hand tracker. To avoid this noise, motion we introduce Head Gaze.

In the second approach, by including the head gaze in the model, it is observed that the robot only moves in the direction of the object where the head is currently facing. While Figure 5.13 shows displacement of the hand tracker about the midpoint, it can be noticed that there is no significant change in the displacement of the tracker. However, for the robot TCP, it changes

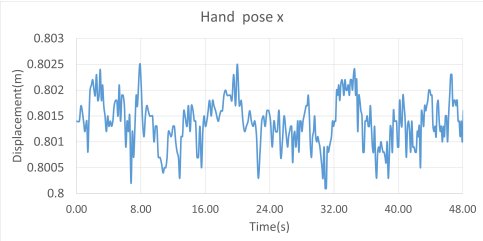


Figure 5.11 – Hand Tracker Motion.

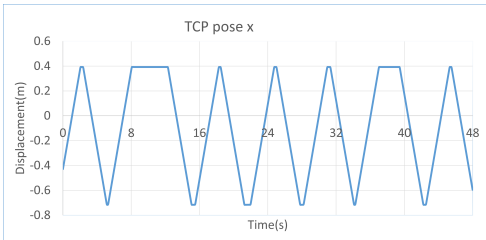


Figure 5.12 – Robot TCP motion without head gaze.

only when the gaze shifts. Figure 5.14 shows the motion of the robot TCP, and when comparing it with Figure 5.12, we can see a reduced noise in the displacement of the robot.

From Figures 5.14 and 5.13, it was observed the movement of the hand does not affect the motion of the robot as long as the head is not facing in the direction of the hand movement. The robot only changes the direction of motion at instances corresponding to the rotation of the head.

The fact that gaze direction is taken into account limits the inappropriate variations of the target point and allows a smoother movement of the robot. This is based on the assumption that the operator looks in the direction where he/she wants to go.

5.4.2 Strategy B: Hand position and gaze direction

Head gazes direction is introduced to limit the detection of points to only what the user can see. The detection of the interesting point is only possible if the point is in the field of view. The strategy is presented in Figure 5.15. The pose of the hand is used to select 2 points, the closest to the hand, in the example P_1 and P_2. From these 2 points, the closest to the gaze direction is selected, by comparison of the angle between the line connecting the point and the line of view. In the example, the closest point is P_2. The strategy is summarized in Algorithm 12.

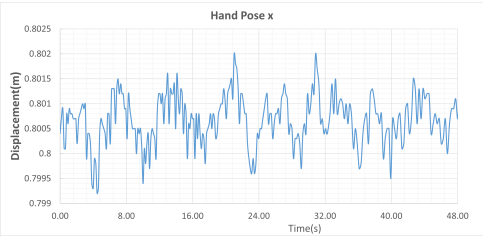


Figure 5.13 – Hand Tracker Motion.

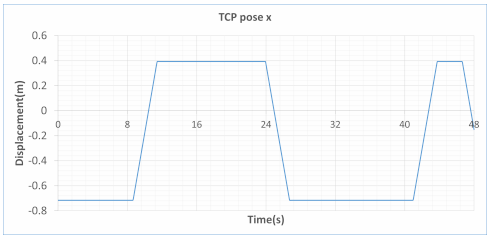


Figure 5.14 – Robot TCP motion with head gaze.

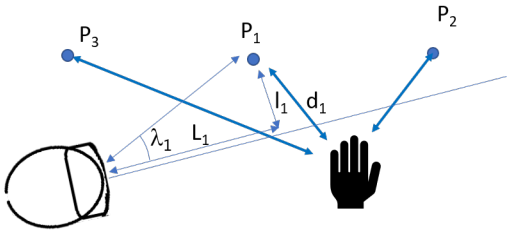


Figure 5.15 – Pictorial representation of Strategy B.

Algorithm 12 Strategy B: Predictions with head gaze.

Input: Hand position $P_h \in R^3$, HMD position $P_s \in R^3$, Head Orientation $O_s \in R^4$.
Output: Nearest Point P.

1: Build a k-d tree for all points in the scene.

2: **function** STB(P_h, P_s, O_s)
3: Using hand pose as a query point q, return nearest 2-points from the k-d tree.
4: Find the gaze direction as a unit vector from the central point of the eyes and draw a ray in the gaze direction.
5: Calculate the distance l_1 of each point in n_p from the ray.
6: Find the angle λ_i for each point such that $\lambda_i = l_i/L_i$ where L_i is the distance from the HMD to the projection of the point on the line. (Shown in Figure 5.15).
7: The point with $\min(\lambda_i)$ is the closest point P.
8: **return** P
9: **end function**

The aim of this approach is to try to find the point of interest with a little anticipation compared to the previous method, by being able to choose a point that may be a little further from the hand but directed according to the direction of the gaze.

5.4.3 Strategy C: Addition of safe points

If the hand is far from the point of interest, it is probably on the way, but still far from the goal, so it may be appropriate to move the robot to a safe point to prepare for a higher speed movement. This strategy is an extension of strategy B, but with added extra safe points. This strategy is designed such that the robot will always go to the safe point if the distance between the hand and the closest point is above a threshold, here 0.2 m. The strategy is presented in

Figure 5.16. In the example, the point P_2, the closest to the view line among the two closest to the hand, is at more than 0.2 m from the hand, thus the robot will go to the safe point which is the closest to P_2 among SP_{20}, SP_{21}, SP_{22}, SP_{23}, SP_{24}. The strategy is summarized in Algorithm 13.

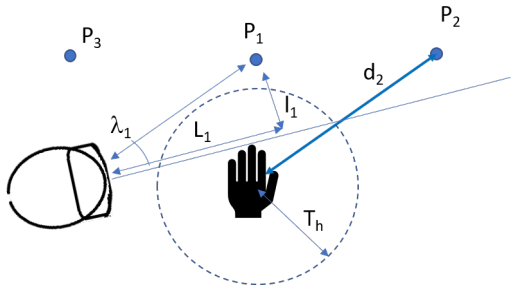


Figure 5.16 – Pictorial representation of Strategy C.

Algorithm 13 Strategy C: Addition of safe-point.

Input: Hand position $P_h \in R^3$, HMD position $P_s \in R^3$, Head Orientation $O_s \in R^4$, Hand Threshold T_h (Shown in Figure 5.16).
Output: Nearest Point P.

Build a k-d tree for all points in the scene.
function STC(P_h, P_s, O_s, T_h)
  function STB(P_h, P_s, O_s)
    return P
  end function
  if distance(P,P_h) $< T_h$ **then**
    return P
  else
    for all $SP_i \in SP$ **do**,
      d_i = distance(P, **SP**$_i$).
      min(d_i);
    end for
    return SP_i
  end if
end function

The main characteristics of this strategy are as follows:

— The difference between this approach and strategy B can only be seen for long displacements (of more than 0.6 m between the points) for which the hand displacement can be quite far from the points P_i, $i = 1, ...18$.
— There is a risk that the robot will move to a safe point in an inefficient way with a longer path that will not allow the robot to arrive faster
— The results obtained can vary with the choice of the threshold T_h, an analysis of the influence of this parameters on the prediction can be found in Section 5.5.6.

5.4.4 Strategy D: Head gaze and safe points

All the strategies presented so far are based on a selection or pre-selection of the point of interest-based on the hand position. Here the approach is different and the selection is based on the direction of the gaze which can greatly anticipate the movement of the hand. As in strategy C, safe points will be used if the point of interest is more than $T_h = 0.2\ m$ away from the hand. The strategy is presented in Figure 5.17. In the example, the point P_2, is the closest to the view line among the points in the frustum of the HMD. As the point P_2 is more than 0.2 m from the hand, thus the robot will go to the safe-point which is the closest to P_2 among SP_{20}, SP_{21}, SP_{22}, SP_{23}, SP_{24}. The strategy is summarized in Algorithm 14.

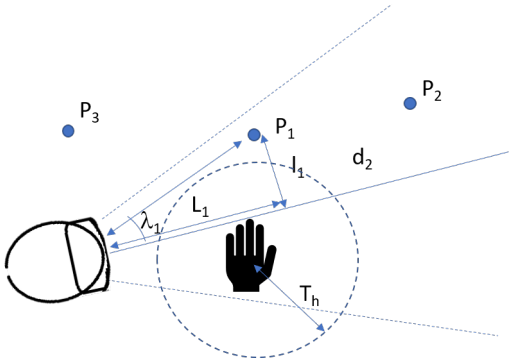


Figure 5.17 – Pictorial representation of Strategy D.

This strategy is based on the assumption that the task will be carried out with the coordination of gaze and movement. In general, it is reasonable to think that the gaze anticipates the movement. In the context of the study, where the human is enclosed in a virtual world, it is likely that he/she will not be disturbed by external elements and that he/she will remain focused with his/her gaze directed towards the point of interest. The context should therefore be favorable to this approach.

5.5 Experiments and analysis

5.5.1 Criterion

The main research question was to find a selection strategy that maximizes user safety while minimizing robot response time. Four strategies explained in the previous section were tested against each criterion. The strategy selected needs to minimize robot time to reach the desired target pose while ensuring maximum user safety.

To evaluate the strategies, the following criteria were considered for strategy.

1. Efficacy:
   - Q_1: If the strategy detects the final point or not. A value of 1 or 0 was assigned if final end point was detect or not.
2. Time for detection:
   - Q_2: Time taken by the strategy to detect the desired/final point the user want to reach.
   - $Q_{2_{norm}}$: In order to be able to compare the results for several strategy, we defined a normalized criterion. It a ratio of each value of Q_2 for a trajectory divide by the min-

Algorithm 14 Strategy D: Predictions with head gaze and safe-points.

Input: Hand position $P_h \in R^3$, HMD position $P_s \in R^3$, head orientation $O_s \in R^4$, hand threshold T_h.
Output: Nearest Point P.

function STD(P_h, P_s, O_s, T_h)
    n_i = points in the view frustum of HMD.
    Find the gaze direction as a unit vector from the central point of the eyes and draw a ray in the gaze direction.
    Calculate the distance of each point in n_i from the ray.
    Find the angle λ_i for each point such that $\lambda_i = l_i/L_i$ where l_i represents the distance between a point and it's projection on the line as calculated in previous step and L_i the distance from the HMD to the projection of the point on the line (Shown in Figure 5.17).
    The point with min(λ_i) is the closest point P.
    if distance(P,P_h) $< T_h$ **then**
        return P
    else
        for all $SP_i \in SP$ **do**,
            d_i = distance(P, $\mathbf{SP}_i$).
            min(d_i);
        end for
        return SP_i
    end if
end function

imum value of Q_2 for this trajectory and the four strategies analyzed. $[Q_2/\min(Q_2)]$. For the best strategy with respect to this criterion the $Q_{2_{norm}} = 1$.

3. Time for robot:
   - Q_3: Time taken by the robot to reach the final point (to move from start to desired point including all via points). It is the sum of the duration of all the pre-computed trajectories according to the strategy of motion of the robot according to section 5.3 and the time that the robot waited to have new point where to go.
   - $Q_{3_{norm}}$: As for the criterion Q_2, we define a normalized criterion, to be able to compare the strategy for several trajectories. Its a ratio of each value of Q_3 for a trajectory divide by the minimum value of Q_3 for this trajectory and the four strategies analyzed. $[Q_3/\min(Q_3)]$. For the best strategy with respect to this criterion the $Q_{3_{norm}} = 1$.
4. Robot distance:
   - Q_4: The distance travelled by the robot from start to end point for all via points the robot travels through.
   - $Q_{4_{norm}}$: The ratio of distance travelled by the robot (from start to end point for all via points the robot travels through) with distance between start and end point.
5. User distance:
   - Q_5: The mean distance between the sphere centered on the driver's seat with a radius of 0.5m and all points on the trajectory. This distance is evaluated via the equation (5.1) and characterizes the safety of the user. The further the robot is from this sphere, the safer it is.

5.5.2 Experimental setup

We used the hand motion sensor as a proximity sensor. The objective is to find the closest ROI with respect to the hand. This is an optimization problem of finding a point in a closed set that is closest to a given point. Using the Head-mounted display, we find the points in the user view and if the gaze is directed towards a point P_i. We classify the distance of the points in the direction of the gaze as a function of l_1/L_1. The user will direct his/her hand towards a capsule (discrete set of N points). The goal is to detect as soon as possible, which point is to be reached by the human and to move the robot to that point:

1. If the direction is known, the robot can be moved to intermediate points to facilitate the task.
2. Safe-points SP_i $(i = 20 : 24)$, located outside the car's interior space defined. Between this points the robot can move quickly.
3. At each moment, from the sensor data, we define the target point among the set of N points P_i $(i = 1 : 18)$.

Seven trajectories were considered in the experimental design: two consisted of long-distance trajectories (from points 2 to 11 and 5 to 18), three medium distances (from points 5 to 11, 5 to 15, 12 to 15), and two shorter distances (from points 3 to 4 and 17 to 16). The seven trajectories have been done by three different participants.

The participant was seated in the car seat $0.6m$ above the ground and at a position of $0.9m$ in y and $-0.1m$ in x from the robot base frame. The sphere used for the obstacle avoidance of the user is centered at this reference point. An HTC Viva HMD was worn by the user and Vive sensors were attached to the user's dominant hand as shown in Figure 5.1. Then for each trajectory, the user was instructed to move his/her hand from a start point to a defined endpoint.

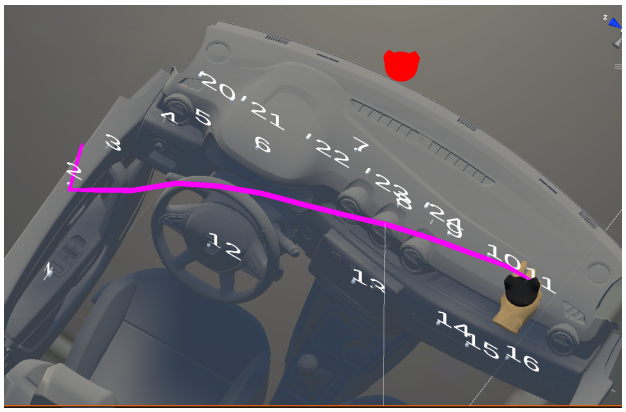


Figure 5.18 – User hand trail for motion from point 2 to 11.

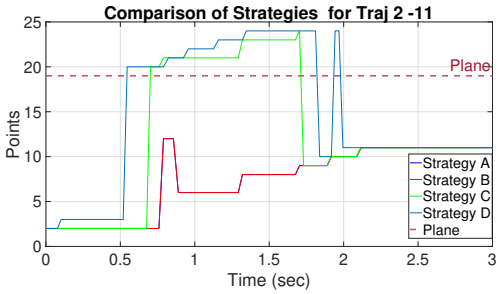


Figure 5.19 – Points of interest detected for different strategies for a hand motion from 2 to 11.

For each trajectory performed by the user, data were recorded. It comprised of the position of the hand tracker, the head position, and orientation. The user can do the task at the speed he/she wants. During this experiment, the robot was not moved. The data recorded was used to perform the analysis in parallel with the four strategies in order to compare them on the same data set.

5.5.3 Analysis of one experiment

The four strategies are described and illustrated on one example, a trajectory done by one subject for moving his/her hand from P_2 to P_{11} was recorded. A visual trail of the user hand is shown in Figure 5.18. This recorded motion was used to analyze the four proposed strategies.

Detection of points of interest

Figure 5.19 shows the sequence of points that are detected for the four studied strategies. It can be visualized that different strategies have different intermediate points selected except for strategies A and B that produced the same sequence of points detected.

Based on the results from the different strategies, it can be observed that strategies A, B select intermediate points which are inside the car while strategy C and D select the safe-points as some of the intermediate points. For this example, all the strategies allows to find the desired final point P_{11}. However the strategy D success to detect this point earlier that the strategies A, B, C that detect the final desired point at the same time (as it can be seen in Table 5.1).

The obtained sequence of points of interest is now detailed:

— Strategy A: P_2, P_{12}, P_6, P_8, P_9, P_{10}, P_{11}.
The points are selected based on the least distance to the hand, the points selected can be easily explained by the hand trail described in figure 5.18.
— Strategy B: P_2, P_{12}, P_6, P_8, P_9, P_{10}, P_{11}.
The selection of this strategy is similar to strategy A for this example because all points were all the time the point closest to the hand is also closest to the direction of gaze. Since the set of points detected is the same as for the strategy A. The robot motion will be similar and analyzed simultaneously.
— Strategy C: P_2, SP_{20}, SP_{21}, SP_{23}, P_9, P_{10}, P_{11}.
For strategy C, a threshold is introduced around the detected points to choose whether the robot should go to the point or to a safe-point. It is observed on Figure 5.18 that the hand passes at a distance $> 0.3m$ from the points P_{12}, P_6, P_8. Consequently, the selected points will be the associated safe-points SP_{20} SP_{21} and SP_{23}. Then points P_9, P_{10}, P_{11} were detected and found to be within the required threshold of the distance from the hand.
— Strategy D: P_2, P_3, SP_{20}, SP_{21}, SP_{22}, SP_{23}, SP_{24}, P_{10}, SP_{24}, P_{11}
Strategy D uses the direction of gaze as the primary criterion and it is therefore more difficult to predict the sequence of points detected based on Figure 5.18. In this strategy P_3 is selected, it is done based on the user gaze and since the hand moved not far from this point, P_3 is selected. For the following points selected by the gaze, the hand is farther from the points so the associated safe-points were selected. Then the gaze is directed toward the point P_{10}, since the distance from the hand was below a threshold T_h, the point P_{10} was selected. Then the gaze is probably oriented to point P_{11} since it's distance from the hand is above the threshold, the robot has to return to the safe-point P_{24} and finally when the the hand is close to the final point P_{11}, the point is detected. This happens at a moment when the point P_{11} is not yet the closest point to the hand and is therefore not yet detected by strategy A. The points detected in this example show that the gaze does not go directly to the goal but sweeps along the path accompanying the hand. The results also show a certain sensitivity of the point sequence to the value chosen for the threshold.

The robot motion for the four strategies

For the three different selections of points (strategies A-B, strategy C, strategy D), the robot motion and the safe distance is now commented. Figures 5.20, 5.21, 5.22 illustrate, starting from the sequence of points detected represented in black as function of time, the corresponding robot motion represented in red as function of time. On the same figure, the distance between the end-effector of the robot and the sphere encompassing the human, is shown in blue as function of time and expressed in meter. This curve is directly calculated based on the robot motion and will be used for the evaluation criterion in Table 5.1.

The sequence of progression of robot motion with time from start to end point is described below:

Strategy A and B:

The progression of the robot motion is indicated by the red line as shown in Figure 5.20. Starting from point P_2, the robot waits for a new point. When P_{12} is detected, the robot starts moving and before it arrives, P_6 is detected. However the robot has to continue and complete

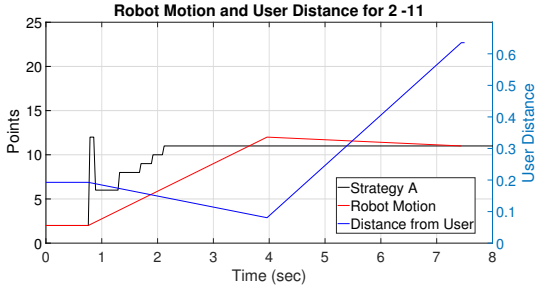


Figure 5.20 – Robot motion, user distance and time for detection, for strategies A and B for trajectory 2 to 11.

the motion so then arriving at P_{12} the robot now detects P_{11} as new desired point, so the robot moves directly to P_{11}. The robot avoids all the points that are detected during the motion towards P_{12}. This pattern continues until the robot reaches the last point detected.

Strategy C:

A graph of robot motion is indicated by the red line as shown in Figure 5.21. The robot starts at P_2 and waits for a new point. When point SP_{20} is detected, it starts moving but along the way, a new point SP_{21} is detected, so it has to complete the motion to SP_{20} first. By the time it has reached the point SP_{20}, points SP_{21}, SP_{23}, SP_{24} have been detected and passed by the prediction algorithm. All the points that were detected and passed during the motion of the robot have been ignored by the robot. Once a new point has been detected the goal state of the robots updates and neglects the previous points that have not been reached. After reaching SP_{20} the prediction now shows point P_{11} as the desired destination. So the robot moves to P_{11}.

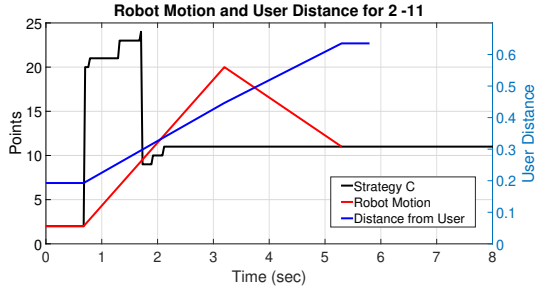


Figure 5.21 – Robot motion, user distance and time for detection, for strategy C using trajectory 2 to 11.

Strategy D:

The motion of the robot is indicated by a red line as shown in Figure 5.22. The robot starts from P_2 and when P_3 is detected, it moves to point P_3. When P_{20} is detected, the robot is still in motion to P_3, so it ignores the point. Further SP_{21}, SP_{22}, SP_{23} and SP_{24} are detected, but when it the robot reaches P_3, the prediction system still predicts SP_{24} as it moves to SP_{24} without waiting. Again during the motion P_{10} and SP_{24} are detected, but before arriving, P_{11} is detected so on arrival at SP_{24}, it does not wait but continues to P_{11}, where it finally stops.

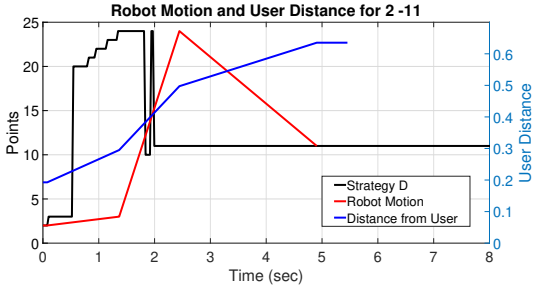


Figure 5.22 – Robot motion, user distance and time for detection, for strategy D using trajectory 2 to 11.

Criterion analysis for single trajectory

Table. 5.1 shows the complete data for all criteria proposed, for the single trajectory 2-11. Q_1 is the success of the strategy detecting the final goal state. It can been seen that all strategies are able to detect the goal point. From the table it can be seen that $Q_{2_{norm}}$ and $Q_{3_{norm}}$ has its best value for strategy D. These strategies have a very fast detection and robot travel time. However the distance traveled is not the best as this strategy allows the selection of safe-points which increase the robot travel distance. When considering safety, strategy D is second best. The best safety Q_5 is provided by strategy C.

Table 5.1 – Strategy analysis for the trajectory 2-11

Trajectory 2-11								
Strategy	Efficacy	Time for Detection		Time for Robot		Robot Dist.		User Dist.
	Q_1	Q_2	$Q_{2_{norm}}$	Q_3	$Q_{3_{norm}}$	Q_4	$Q_{4_{norm}}$	Q_5
St. A	1	2.0868	1.0600	7.4413	1.5158	1.4648	1.1959	0.4370
St. B	1	2.0867	1.0600	7.4413	1.5158	1.4648	1.1959	0.4370
St. C	1	2.0867	1.0600	5.2969	1.0790	1.5690	1.2810	**0.4804**
St. D	1	1.9686	**1**	4.9092	**1**	2.1931	1.7903	0.4428

After this detailed analysis for one trajectory, a discussion of the criterion evaluation for seven trajectories recorded is presented in the following sections.

5.5.4 Analysis of all recorded experiments

The results obtained are summarized in Table. 5.2. For all the tests, the final desired position is detected, thus criterion efficacy Q_1 is one and the criterion is not summarized in the Table 5.2. The analysis of the results will be separated into two parts. Firstly, an observation of the results obtained for the different trials will enable us to arrive at certain conclusions about less variations of the results as function of the trajectories. Then, in a second part, we will use the average values of the different criteria, taking into account the seven trajectories, to highlight the particularities of each strategy by comparing the criteria two by two.

Table 5.2 – Complete analysis for seven trajectories

	Criteria	Strategy	Long Trajectory		Medium Trajectory			Short Trajectory		Analysis	
			2-11	5-18	5-11	5-15	12-15	3-4	17-16	Mean	St. Dev
Time for Detection	Q_2	A	2.0868	2.3029	2.5878	2.1767	1.9842	0.6418	1.0381	1.8312	0.6597
		B	2.0868	2.3029	2.5878	2.1767	2.3366	0.6418	1.0381	1.8815	0.6825
		C	2.0868	2.5370	2.5878	2.1767	2.3366	0.6418	1.0381	1.9150	0.7076
		D	1.9687	2.5370	2.2383	2.0428	2.3366	0.3258	0.8709	1.7600	0.7687
	$Q_{2_{norm}}$	A	1.0600	**1**	1.1561	1.0655	**1**	1.9701	1.1919	1.2062	0.3190
		B	1.0600	**1**	1.1561	1.0655	1.1776	1.9701	1.1919	1.2316	0.3085
		C	1.0600	1.1016	1.1561	1.0655	1.1776	1.9701	1.1919	1.2461	0.2995
		D	**1**	1.1016	**1**	**1**	1.1776	**1**	**1**	**1.0399**	0.0663
Time for Robot	Q_3	A	7.4413	7.2735	7.5526	8.0443	5.4879	2.4640	6.9719	6.4622	1.7930
		B	7.4413	7.2735	7.3964	8.0443	5.4879	2.4640	6.9719	6.4399	1.7802
		C	5.2969	7.2735	6.6089	8.0443	4.3095	2.4640	7.9156	5.9875	1.9159
		D	4.9092	9.8848	4.9460	5.8196	6.0033	2.1480	5.7774	5.6412	2.1183
	$Q_{3_{norm}}$	A	1.5158	**1**	1.5270	1.3823	1.2734	1.1471	1.2067	1.1605	0.1802
		B	1.5158	**1**	1.4954	1.3823	1.2734	1.1471	1.2067	1.1848	0.1746
		C	1.0790	**1**	1.3362	1.3823	**1**	1.1471	1.1701	1.2003	0.1591
		D	**1**	1.3590	**1**	**1**	1.3930	**1**	**1**	**1.0398**	0.1701
Robot Dist.	$Q_{4_{norm}}$	A	**1.1959**	**1.4943**	**1.0948**	**1.1582**	**1.0225**	**1.0000**	**1.3042**	**1.1814**	0.1601
		B	**1.1959**	**1.4943**	**1.0948**	**1.1582**	**1.0225**	**1.0000**	**1.3042**	**1.1814**	0.1601
		C	1.2810	1.6766	3.9707	1.5271	3.0161	1.0000	2.9585	2.2043	1.0273
		D	1.7904	3.9039	7.4400	2.8315	4.2133	1.0000	2.6207	3.4000	1.9464
User Dist.	Q_5	A	0.4370	0.3301	0.4846	0.4143	0.3285	0.3535	0.4429	0.3987	0.0570
		B	0.4370	0.3301	0.4846	0.4143	0.3285	0.3535	0.4429	0.3987	0.0570
		C	**0.4804**	**0.4134**	**0.4932**	0.4149	0.3165	0.3535	0.4684	**0.4200**	0.0616
		D	0.4428	0.3836	0.4199	**0.4327**	**0.3510**	0.3535	**0.5044**	0.4126	0.0506

Variations of the results for the different trajectories

For almost all trajectories, the strategy B and A produce the same result, but for one test, the direction of the gaze is not well directed at the end of motion and a delay is observed with strategy B, contrary to what is expected. This delay affects the time of detection and also the time for the motion of the robot (trajectory 12-15).

It can be observed that in all the trials, the method that allowed the fastest detection of the desired point of contact was also the one that allowed the robot to reach this point as quickly as possible.

From the point of view of the efficiency of detection of the contact point, strategies A and D were the most efficient: two times for strategy A and five times for strategy D. There is no clear correlation between the efficiency of the strategy and the length of the trajectory. However for short trajectories, the strategy D is the most efficient.

For all trajectories, the distance traveled by the robot is always smaller for strategies A and B, which was expected as the robot does not move to safe-points. Consequently, in contrary strategies C and D have higher user distance.

User distance versus time for detection

For the purpose of analyzing, $-Q_5$ has been used so as the goal would be to minimize all the selected criteria. A comparison of $-Q_5$ and $Q_{2_{norm}}$ shows that strategies C and D belong to the Pareto front for these two criteria. Strategy D takes the least time to detect a desired point the user would like to reach and a slightly higher mean distance from the sphere as shown

in Figure 5.23. Strategy C has the largest user distance. However, it takes the longest time to detect a point than all the strategies.

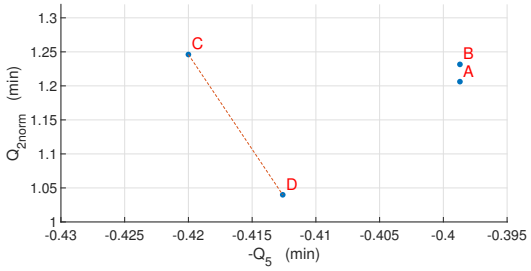


Figure 5.23 – Comparison of time for detection ($Q_{2_{norm}}$) vs user distance ($-Q_5$) for the four strategies, all trajectories.

User distance versus robot distance

A comparison of $-Q_5$ and $Q_{4_{norm}}$ as presented in Figure 5.24 shows that strategies A-B and C belong to the Pareto front for these two criteria. Since strategies A and B don't use any safe points they have smallest user distance (max $-Q_5$) and always take the minimal robot distance to arrive to a desired point (min $Q_{4_{norm}}$). Strategies C and D use safe points so, have higher user distance (min $-Q_5$) and robot distance $Q_{4_{norm}}$. While strategy D uses head gaze as primary selection, it gives a value of Q_5 better than A and B but the worst robot distance.

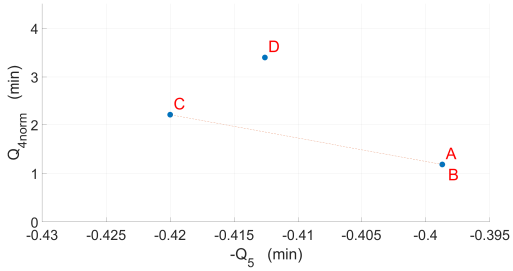


Figure 5.24 – Comparison of robot distance ($Q_{4_{norm}}$) vs user distance ($-Q_5$) for the four strategies, all trajectories.

User distance versus time for robot

A comparison of $-Q_5$ and $Q_{3_{norm}}$ as presented in Figure 5.25 shows that strategy C and D belong to the Pareto front for both criteria. As known Strategy C gives a higher value for safety (min $-Q_5$) followed by strategy D. However strategy D far outperforms strategy C in terms of time for the robot.

Time for robot versus time for detection

Visualization of $Q_{3_{norm}}$ vs $Q_{2_{norm}}$ is shown in Figure 5.26. It shows that the faster the strategy detects the point, faster the robot reaches the point. The best being strategy D. It uses the head gaze and an added use of safe-points helps it in reaching the desired point faster, as the

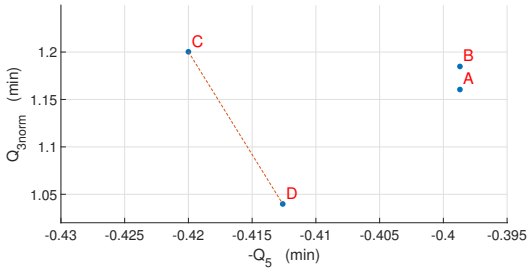


Figure 5.25 – Comparison of time for robot ($Q_{3_{norm}}$) vs user distance ($-Q_5$) for the four strategies, all trajectories.

robot travels at higher velocity than in strategies A and B. Strategy A has lots of intermediate point, but as the hand moves closer to the desired point the robot gradually moves closer. This is one of the reason why this strategy is the second best. Even thought strategies B and C have a head gaze, the primary selection is still based on the hand. Unless the hand is closer to the point the robot does not move to the desired point. Strategy C takes the longest time in both the axis, because the selection process used in strategy C is based primarily on hand location. For strategy D it is the user gaze that help in primary selection of points.

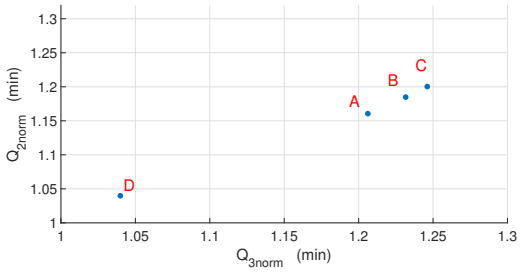


Figure 5.26 – Comparison of time for robot ($Q_{3_{norm}}$) vs time for detection ($Q_{2_{norm}}$) for the four strategies, all trajectories.

Robot distance versus time for robot

Visualization of $Q_{4_{norm}}$ vs $Q_{3_{norm}}$ is shown in Figure 5.27 strategies A and D belong to the Pareto front for these two criteria. In contrary to the assumption that longer robot distance implies longer time for robot, the results show that strategy D has minimum time for robot but has longer robot distance. This result achieved is due to combination of fast time for detection and use of safe-point as intermediate points. Strategy C uses also safe-points, but it alone does not guarantee a fast response time.

Time for detection versus robot distance

Visualization of $Q_{2_{norm}}$ vs $Q_{4_{norm}}$ is shown in Figure 5.28 strategies A and D belong to the Pareto front for these two criteria. From Figure 5.28,it can be see that Strategy A and B have least robot distance, but its detects the goal later than Strategy D. Strategy C is not ideal in both criteria.

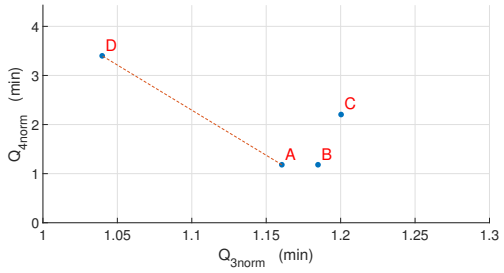


Figure 5.27 – Comparison of robot distance ($Q_{4_{norm}}$) vs time for robot ($Q_{3_{norm}}$) for the four strategies, all trajectories.

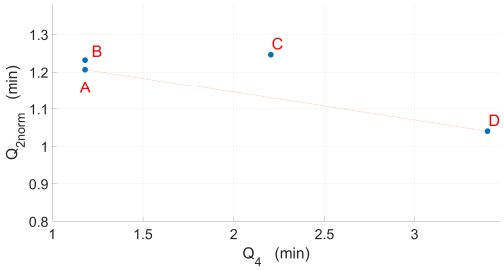


Figure 5.28 – Comparison of time for detection ($Q_{2_{norm}}$) vs robot distance ($Q_{4_{norm}}$) for the four strategies, all trajectories.

5.5.5 Discussion

A comparative analysis of data from all the trajectories (in Table. 5.3) shows that, if the objective is to maximize safety strategy C and D would be better. Both the strategies C and D ensure safety by selecting safe-points when the hand is far away from the desired point. The safe-points are located outside the user reach, such that the robot can travel fast and does not collide with the user. The selection of the safe-points mean that the robot will have to travel a longer distance to reach the desired point. While for strategies A and B, they select intermediate points which are inside the car and no safe-points. So since the points are all inside the car, and the robot does not a travel longer distance but has reduced velocity.

Table 5.3 – Analysis of strategies for all trajectories.

Strategy	Time for Detection	Time for Robot	Robot Dist.	User Dist.
St. A	+	+	++	-
St. B	-	-	++	-
St. C	-	-	-	++
St. D	++	++	-	+

Strategy D gives second best safety and at the same time minimize the time to detect/reach a desired point. Therefore it can be seen as the best strategy. The detection time for strategy D is the smallest because we used the gaze of the user to pre-select the points. This plays a big role in giving priority to vision information over information from the hand position. Fastest detection time allows the robot to start moving to the desired point at the earliest time and

reach the desired point the fastest.

5.5.6 Analysis on hand threshold

In strategy C, the primary selection is based on user hand unlike in strategy D which uses the gaze direction. Since the selection in strategy C depends on the values of hand threshold T_h. In this subsections, Strategy C is selected to investigate the effect of each parameter on the time to detect a point, robot time, user distance and Robot distance. The same 7 trajectories recorded by same 3 users were used for this analysis.

The effect is analyzed by adjusting the threshold value T_h using $T_1 = 0.10$ m, $T_2 = 0.15$ m, $T_3 = 0.2$ m, $T_4 = 0.25$ m and $T_5 = 0.3$ m.

Table 5.4 – Analysis for various thresholds

T_h	$Q_{2_{norm}}$		$Q_{3_{norm}}$		$Q_{4_{norm}}$		Q_5	
	Mean	St. Dev	Mean	St. Dev	Mean	St. Dev	Mean	St. Dev
0.1	1.6084	0.4919	0.9796	0.1861	2.1135	1.0912	0.4467	0.0871
0.15	1.4882	0.4075	0.9197	0.2484	2.0835	1.0461	0.4271	0.0381
0.2	1.2461	0.2995	1.2003	0.2214	2.2043	1.0273	0.4200	0.0616
0.25	0.9862	0.3312	1.3204	0.2246	2.2857	0.9892	0.3975	0.2583
0.3	0.7660	0.3865	1.5808	0.2206	2.3175	1.0561	0.3848	0.2378

User distance versus time for detection

A comparison of $-Q_5$ and $Q_{2_{norm}}$ as presented in Figure 5.29 shows that all thresholds from T_1 to T_5 belong to the Pareto front for these two criteria. T_1 takes the highest time to detect a desired point the user would like to reach and it gives the best mean distance from the sphere because the threshold is low so the robot is most of the time in safe points. In case of T_5 since the threshold is large the robot is closer to the user.

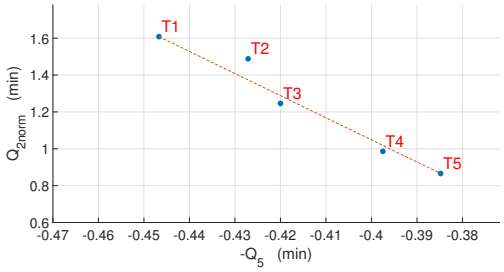


Figure 5.29 – Comparison of time for detection ($Q_{2_{norm}}$) vs user distance ($-Q_5$) for 5 thresholds.

User distance versus robot distance

A comparison of $-Q_5$ and $Q_{4_{norm}}$ as presented in Figure 5.30 shows that T_1 does not belong to the Pareto front for these two criteria. During the experiments it was realized that with small threshold like T_1 the robot sometimes fails to stop after it has detected the point. When the

user has reached the desired point and interacting with the surface. It was noticed that a small variation in hand position causes the robot to return to the safe points. This can be seen in Figure 5.30. T_1 despite detecting the point earlier (concluded from Figure 5.29) fails to have the best robot distance.

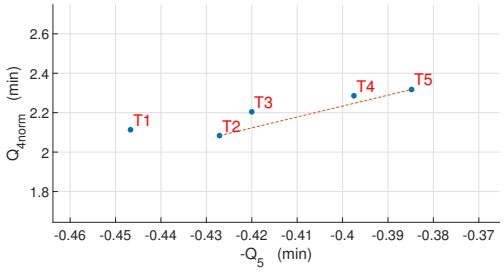


Figure 5.30 – Comparison of robot distance ($Q_{4_{norm}}$) vs user distance ($-Q_5$) for 5 thresholds.

User distance versus time for robot

A comparison of $-Q_5$ and $Q_{3_{norm}}$ as presented in Figure 5.31 shows that T_2, T_3, T_4 and T_5 belong to the Pareto front for both criteria. As known T_2 gives a higher value for safety (min $-Q_5$) and time for robot but fails to outperform T_3 in detection time (concluded from Figure 5.29).

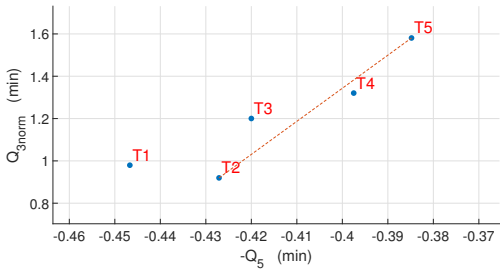


Figure 5.31 – Comparison of time for robot ($Q_{3_{norm}}$) vs user distance ($-Q_5$) for 5 thresholds.

Time for robot versus time for detection

A comparison of $Q_{3_{norm}}$ vs $Q_{2_{norm}}$ as presented in Figure 5.32 shows that T_1 does not belong to the Pareto front for these two criteria. Lower thresholds give a slow response of detection but fast robot time due to use of safe points. Lower thresholds have fewer intermediate point inside the car, but as the hand moves closer to the desired point the robot gradually moves closer. This is one of the reason why lower thresholds are good in robot response time. Even thought Strategy C has a head gaze, the primary selection is still based on the hand. Unless the hand is closer to the point the robot does not move to the desired point. If the threshold is big the robot continues to move inside the low velocity region and not pass through the safe-points.

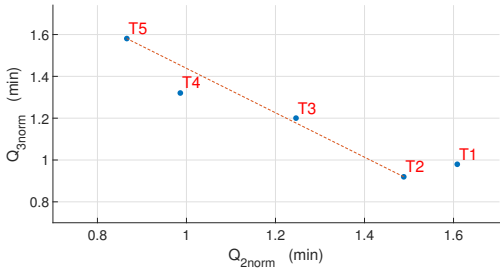


Figure 5.32 – Comparison of time for robot ($Q_{3_{norm}}$) vs time for detection ($Q_{2_{norm}}$) for 5 thresholds.

Robot distance versus time for robot

A comparison of of $Q_{4_{norm}}$ vs $Q_{3_{norm}}$ is shown in Figure 5.33. The results are consistent with the assumption that longer robot distance implies longer time for robot, the results show that lower thresholds have minimum robot distance and time for robot. The explanation for such behavior is similar to analysis seen before for $Q_{3_{norm}}$ vs $Q_{2_{norm}}$. Larger thresholds cause the detection of points inside the user space and cause the robot to move in low velocities.

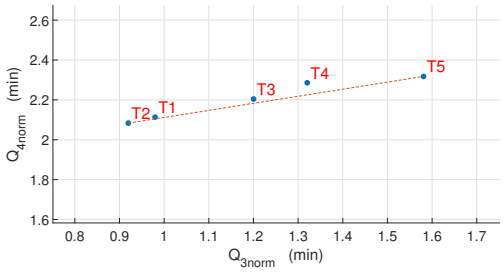


Figure 5.33 – Comparison of robot distance ($Q_{4_{norm}}$) vs time for robot ($Q_{3_{norm}}$) for 5 thresholds.

Robot distance versus time for detection

A comparison of $Q_{2_{norm}}$ vs $Q_{4_{norm}}$ is shown in Figure 5.34. T_1 does not belong to the Pareto front for both criteria. It can be see that T_2 and T_1 have least robot distance, but its detects the goal later than T_3 and T_4.

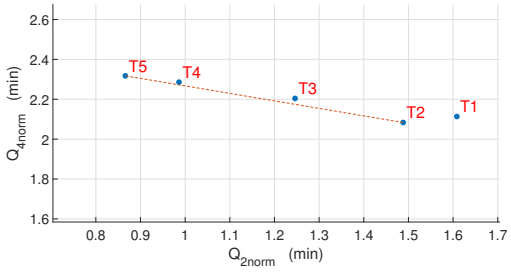


Figure 5.34 – Comparison of robot distance ($Q_{4_{norm}}$) vs time for detection ($Q_{2_{norm}}$) for 5 thresholds.

Overall, there was no significant difference between $T_h = 0.10$ m, and 0.15 m for all the objectives $Q_{2_{norm}}$, $Q_{3_{norm}}$, $Q_{4_{norm}}$ and Q_5. For this study, the best value to be selected is $T_h = 0.20$ m. Its based on several reasons. First, the task needs user to hold/interact with a surface whose width is 10cm. Secondly, the hand threshold is used to detect the intention of user to move his/her hand to the next point. A sufficient distance between the hand and the surface points was necessary to notice that the human had let it go after touching. Lastly, the tracker is placed on the top of hand, a distance of approximately 4cm from the palm. So the total distance from the center of the prop to the tracker is about about 15cm.

5.6 Conclusions

Four motion prediction strategies are used to select the areas with which the user intends to interact and to move the robot as fast as possible while ensuring user safety.

We introduce two speed profiles for the user's safety. The robot moves at a higher speed when it is outside the user's workspace. In situations where there is a large distance between two points within the workspace, we introduce via points to reduce travel time. The time needed to go through via points can be less than the time needed to go directly inside the car while being much safer.

Seven trajectories done by three users were analyzed thank to five criteria. A compromise must be made between user safety and speed for the robot to reach its target.

Conclusions and future work

Synopsis

Force feedback interfaces are robotic systems allowing natural motion interactions with virtual or remote environments. They are employed in several domains such as remote handling, manufacturing, entertainment, education, medicine and rehabilitation, just to mention the most popular. In virtual reality applications, the user typically holds a handle that is mechanically linked to the end-effector of the robot. This link has a non-negligible influence since the presence of the robot can be felt even in free space, decreasing the realism of the interaction.

Intermittent contact haptic interfaces are a promising technological development aiming to cope with this issue. These interfaces track and closely follow (without contact) the user movements in free space and come to his/her contact only when force feedback is required. This way ICI haptic interfaces aim to provide more realistic interactions with virtual environments.

Their design and objective evaluation are however particularly complex and there are still challenges to be met.

The focus of this doctoral thesis was on creating a robot control system for haptic-tactile sensation using such ICI systems. A system that is both safe and immersive for the user. The research is guided in 2 major directions:

— Improvement of the performances of ICI interfaces while ensuring high user safety and security.
— Improving the immersive experience of the ICI model by providing improved prediction and robot trajectory planning.

A literature review about ICIs was presented in Chapter 1. This chapter proposed a definition of the term Intermittent Contact Interface and its fundamental notions. Later, the history of haptic device was presented to showcase how the field has evolved over the years. This chapter also contributed with a classification for haptic devices according to their haptic feedback. Then, a hardware review was presented where the research works were classified according to the feedback types. The research works were reviewed according to the type of haptic feedback, interaction, and application. This chapter concluded by raising the research questions to be answered in this thesis based on the field's limitations and research opportunities.

The final and important way forward is to integrate this research with other collaborators. This research dealt with giving a trajectory planning of the robot which is safe for user and gives him independence on selection of interactive points. INRIA team had worked on designing different prop prototypes and rendering immersive interactive haptic displays. CLARTE works to integrate these contributions and also to develop better immersive VR environment.

The major contributions of this doctoral thesis are summarized as follows.

Contributions

1. Improving user safety and security by defining a robot base location.

Chapter 2 presented the choice of a robot base placement to increase the safety when interacting with an ICI systems. The chapter introduced a design dedicated to increasing user safety based on goal positions (interaction points) to estimate the best robot base location. A set of tool orientations based on the TCP design were considered, and different robot workspace were created to explore the different possible base locations for avoiding collisions with the user occluded in haptic display. Later a preliminary simulation was carried with a group of interaction points to reflect the safety precautions such as position of robot far from the user, single base location to reach all the desired intersection points. Then the chapter presented the results of the evaluation made under the criteria previously proposed. Overall, results from this evaluation suggest that a single robot base location with a new table design of the robot was possible to reach all the points of interests in the car, without compromising user safety.

2. Improving user safety by building user model, for collision detection.

Chapter 3 introduces a user model designed in ROS based on the vive trackers. The model is designed to actively track the user motion for collision avoidance during robot motion planning. The model tracks both user hands and also his/her torso movement, the accuracy of the model was also analyzed in this chapter. This technique uses four vive trackers, two on each hands of the user. These trackers are used to estimate the position and orientation of the user hand and torso. Key features of the model were portrayed in illustrative scenarios that allowed user model to track the user for obstacle avoidance in Rviz, ROS.

3. Motion planning

Chapter 4 presented motion planning and velocity control strategies for the robot, considering collision avoidance based on the user model designed in Chapter 3. This approach uses the idea of restricting the max robot velocity in different regions based on the possible user location and interaction points in the VR. This approach was envisioned to optimize the robot trajectory time to increase the immersion of the user by reducing the wait time for the user to interact with the robot. This was achieved by introducing via points in the region with higher velocity. So the robot passes faster in between points to reach the goal.

4. Improving the security and immersion of user by prediction of intention

Chapter 5 studied the design and evaluation of intention prediction techniques for ICIs. First, a design for interaction techniques was presented along with its main features: input, movement control, and contact. Later, the chapter presented a set of interaction techniques conceived from the aforementioned designs. A use-case scenario was created to contextualize the use of the techniques. Later, the techniques were evaluated in a user study to assess user experience and their performance in helping users to accomplish the task. Results suggested that the techniques using the head-gaze from HMD and vive tracker in hand are useful for scenarios that requires detection of early user intention. The designed techniques could be used in other application scenarios and contexts where other types of applications might be required.

Future work

The presented contributions in this thesis could be further extended as future work by focusing on key areas.

1. Improve user safety

In the case of addressing user safety presented in Chapter 4, the motion planning can be improved by considering the motion of the user mannequin for real-time planning scenarios. At present the strategy plans the trajectory offline and uses the appropriate one based on the user prediction. The full potential of the user mannequin has not been used in this case, during the trajectory execution additional features can be added to the planning strategy used to avoid collision with the user.

The single scalable user model can be used to create various sizes of user models and have different set so pre-computed trajectories. These trajectories can be accessed based on the dimension of the user. In the present research a fixed sphere is used to have simple occupancy of the user workspace. But the user sphere can be adjusted based on the user height/user model dimensions to have better user safety. By changing the sphere based on the user, the velocity boundary limits also changes.

Due to the complexity of the system, a complete characterization of real-time dynamic motion planning based on the user model was not possible and is still an open problem. The present approach only does a pre-computed (off-line) trajectory planning, a improved planning system where the trajectories are computed directly and re-planning is possible to have better response time.

2. Improve robot response

When addressing the response time of the robot, at present we have introduced velocity limits in different zones (Chapter 5). As mentioned in Chapter 4, the trajectories are computed offline and stored to be accessed later based on the prediction system. Now to increase the response time of the robot one way is to introduce more number of via points for the robot to pass. In present scenario only 23 points are used, 18 for task interaction and 5 safe points (via-points). By introducing multiple points and using the prediction system from Chapter 5, a better response can be achieved. As the robot will start its motion earlier (as points will be detected earlier). A study to analyze the effect of the number of point and relative position or distance would be an interesting direction to explore.

3. Improve haptic rendering for complex surfaces

The present research has only considered rendering of small flat surfaces. The research can be extend to have a continues rendering of a surface. Further developments can be made to render complex planes (not just flat). How to virtually have the sense of infinite surface has been discussed in [88] using a different kind of prop. But the further challenge will be to plan the robot and improve the prediction and trajectory planning to achieve the same.

Another development can be made to render the stiffness of the material the user interacts. The information of different stiffness of the sample materials can be accessed and used to control the robot force impedance. This would improve the immersion and result in better tactile feedback.

Coda

This thesis introduces a technology that seems to be specific to a specific context, but such technologies have a huge potential. This technology can also be used in entertainment industry and improved to augmented reality. Research and development in such technology have a effect on humanity and I hope my contribution to this has helped in developing towards the betterment of it.

Personal publications

Journal publications

1. Guda, V. K., Mugisha, S. Chevallereau, C., Zoppi, M., Molfino, R., Chablat, D. (2022). Motion Strategies for a Cobot in a Context of Intermittent Haptic Interface. Journal of Mechanisms and Robotics.
2. Mugisha, S., Guda, V. K., Chevallereau, C., Zoppi, M., Molfino, R., Chablat, D. (2022). Improving Haptic Response for Contextual Human Robot Interaction. Sensors, 22(5), 2040.

Conference publications

1. Guda, V. K., Chablat, D., Chevallereau, C. (2020, July). Safety in a human robot interactive: Application to haptic perception. In International Conference on Human-Computer Interaction (pp. 562-574). Springer, Cham.
2. Mugisha, S., Zoppi, M., Molfino, R., Guda, V., Chevallereau, C., Chablat, D. (2021, August). Safe collaboration between human and robot in a context of intermittent haptique interface. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 85451, p. V08BT08A007). American Society of Mechanical Engineers.
3. Gutierrez, A., Guda, V., Mugisha, S., Chevallereau, C., Chablat, D. (2022, June). Trajectory planning in Dynamics Environment: Application for Haptic Perception in Safe Human-Robot Interaction. In 24TH International Conference on Human-Computer Interaction.

Appendix A

UR-5 kinematics

Introduction

For the remainder of the document, we use the notation $c_i = \cos\theta_i$, $s_i = \sin\theta_i$, and for angle sums, $c_{ij} = \cos(\theta_i + \theta_j)$.

Forward Kinematics

We first begin by giving the forward kinematics, describing the position of the end effector as a function of joint angles. The transformation matrix is defined as:

$$ {}^0T_6\left(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6\right) = \begin{bmatrix} {}^0R_6 & {}^0p_6 \\ 0 & 1 \end{bmatrix} \tag{A.1} $$

$$ {}^0T_6\left(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6\right) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

We can split the transformation matrix is given as a chain of transformations, one for each joint:

$$ {}^0T_6\left(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6\right) = {}^0\,T_1\left(\theta_1\right)^1T_2\left(\theta_2\right)^2T_3\left(\theta_3\right)^3T_4\left(\theta_4\right)^4T_5\left(\theta_5\right)^5T_6\left(\theta_6\right) \tag{A.2} $$

The kinematic structure of the UR-5 robot in zero position is shown in Figure A.1:
The DH-parameters shown are specified as:
— a_i = distance from z_i to z_{i+1} measured along x_i
— α_i = angle from z_i to z_{i+1} measured about x_i
— d_i = distance from x_{i-1} to x_i measured along z_i
— θ_i = angle from x_{i-1} to x_i measured about z_i
For the UR-5, the DH parameters are shown in Table A.1. The DH-parameters can be used to write the transformations for each link. The general transformation between link i-1 and i is given by :

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos(\alpha_{i-1}) & \cos\theta_i\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin\theta_i\sin(\alpha_{i-1}) & \cos\theta_i\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3} $$

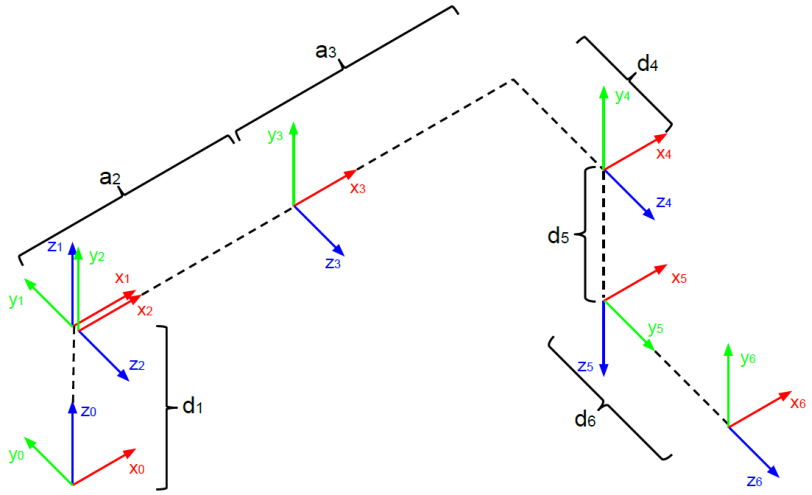


Figure A.1 – Coordinate frames for UR5 manipulator. Joints rotate around the z-axis and are pictured at $\theta_i = 0$ for $1 \leq \mathrm{i} \leq 6$

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	0	$\pi/2$	0	θ_2
3	a_2	0	0	θ_3
4	a_3	0	d_4	θ_4
5	0	$\pi/2$	d_5	θ_5
6	0	$-\pi/2$	d_6	θ_6

Table A.1 – Modified Denavit-Harteberg parameters (DH-parameters) of a UR5 robot, corresponding to the frames in Figure A.1. The parameters θ_i are variables and the remaining parameters are constants.

It is straightforward to write the transformation matrix for each link of the UR-5 robot by inserting the DH-parameters from Table A.1 in general transformation equation . The complete transformation from base to end-effector can then be derived by multiplication of all 6 transformation matrices. The result is analytical expressions for all 12 parameters in the transformation matrix ${}^0T_6$. The complete analytic equations can be found below:

$$n_x = c_6(s_1s_5 + ((c_1c_{234} - s_1s_{234})c_5)/2.0 + ((c_1c_{234} - s_1s_{234})c_5)/2.0) - \Big(s_6((s_1c_{234} + c_1s_{234}) - (s_1c_{234} - c_1s_{234}))\Big)/2.0$$

$$n_y = c_6(((s_1c_{234} + c_1s_{234})c_5)/2.0 - c_1s_5 + ((s_1c_{234} - c_1s_{234})c_5)/2.0) + s_6\Big((c_1c_{234} - s_1s_{234})/2.0 - (c_1c_{234} - s_1s_{234})/2.0\Big)$$

$$n_z = (s_{234}c_6 + c_{234}s_6)/2.0 + s_{234}c_5c_6 - (s_{234}c_6 - c_{234}s_6)/2.0$$

$$o_x = -(c_6((s_1c_{234} + c_1s_{234}) - (s_1c_{234} - c_1s_{234})))/2.0 - s_6\Big(s_1s_5 + ((c_1c_{234} - s_1s_{234})c_5)/2.0 + ((c_1c_{234} + s_1s_{234})c_5)/2.0\Big)$$

$$o_y = c_6((c_1c_{234} - s_1s_{234})/2.0 - (c_1c_{234} + s_1s_{234})/2.0) - s_6\Big(((s_1c_{234} + c_1s_{234})c_5)/2.0 - c_1s_5 + ((s_1c_{234} - c_1s_{234})c_5)/2.0\Big)$$

$$o_z = (c_{234}c_6 + s_{234}s_6)/2.0 + (c_{234}c_6 - s_{234}s_6)/2.0 - s_{234}c_5s_6$$

$$a_x = c_5s_1 - ((c_1c_{234} - s_1s_{234})s_5)/2.0 - ((c_1c_{234} + s_1s_{234})s_5)/2.0$$

$$a_y = -c_1c_5 - ((s_1c_{234} + c_1s_{234})s_5)/2.0 + ((c_1s_{234} - s_1c_{234})s_5)/2.0$$

$$a_z = (c_{234}c_5 - s_{234}s_5)/2.0 - (c_{234}c_5 + s_{234}s_5)/2.0$$

$$p_x = -(d_5(s_1c_{234} - c_1s_{234}))/2.0 + (d_5(s_1c_{234} + c_1s_{234}))/2.0 + d_4s_1 - (d_6(c_1c_{234} - s_1s_{234})s_5)/2.0 - (d_6(c_1c_{234} + s_1s_{234})s_5)/2.0 + a_2c_1c_2 + d_6c_5s_1 + a_3c_1c_2c_3 - a_3c_1s_2s_3$$

$$p_y = -(d_5(c_1c_{234} - s_1s_{234}))/2.0 + (d_5(c_1c_{234} + s_1s_{234}))/2.0 - d_4c_1 - (d_6(s_1c_{234} + c_1s_{234})s_5)/2.0 - (d_6(s_1c_{234} - c_1s_{234})s_5)/2.0 - d_6c_1c_5 + a_2c_2s_1 + a_3c_2c_3s_1 - a_3s_1s_2s_3$$

$$p_z = d_1 + (d_6(c_{234}c_5 - s_{234}s_5))/2.0 + a_3(s_2c_3 + c_2s_3) + a_2s_2 - (d_6(c_{234}c_5 + s_{234}s_5))/2.0 - d_5c_{234}$$

Inverse kinematic solution for UR-5, 6-DoF arm

The inverse kinematic (IK) equations calculates the joint angles θ_{1-6} based on a desired position and orientation of the final frame, specified as the transformation ${}^0T_6$.

The analytic inverse kinematics problem is to find the set of joint configurations Q = q_i where $q_i = ({\theta_1}^i, {\theta_2}^i, {\theta_3}^i, {\theta_4}^i, {\theta_5}^i, {\theta_6}^i) \in [0, 2\pi)^6$ that satisfies

$${}^0T_6(\theta_1^i, \theta_2^i, \theta_3^i, \theta_4^i, \theta_5^i, \theta_6^i) = ({}^0T_6^d) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

where ${}_0{T_6}^6$ describes the desired position and orientation of the final link.

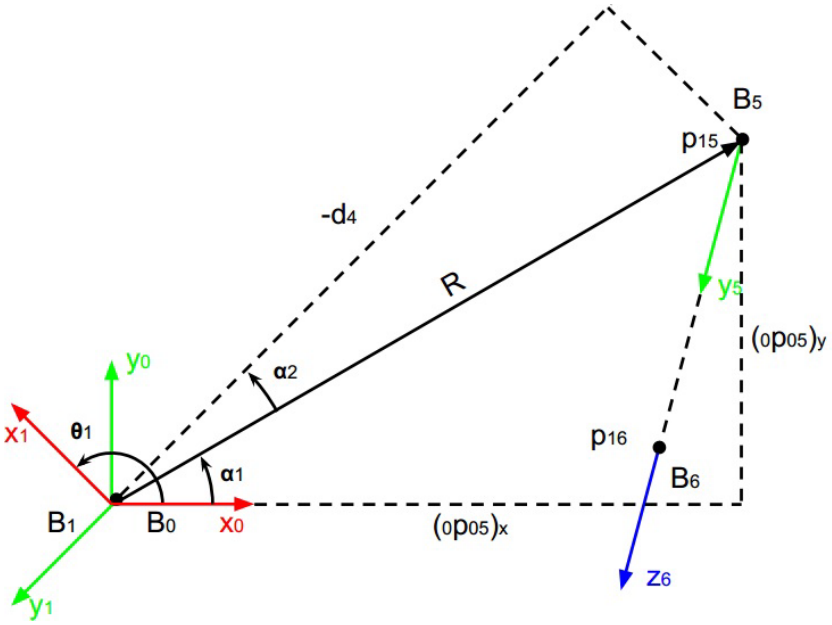


Figure A.2 – Illustration of the geometry of finding θ_1. This is essentially an overhead view of the robot, looking down on the x-y plane.

Finding θ_1

To find θ_1, we first determine the location of frame 5 (the wrist frame) in relation to the base frame; ${}^0p_{05}$. As illustrated in Figure A.2, ${}^0p_{05}$ can be found by translating backwards from frame 6 to frame 5 along z_6. Remember that both ${}^0T_6$ and d_6 are known. Analyzing the transformation from frame 1 to frame 5 using equations (1) and (2), we can form the equality

$$\left[{}^1T_5\right]_{LHS} = \left[{}^1T_5\right]_{RHS} \tag{A.5}$$

$$[({}^0T_1)^{-1}({}^0T_6^d)({}^5T_6)^{-1}]_{LHS} = [({}^1T_2)({}^2T_3)({}^3T_4)({}^4T_5)]_{RHS} \tag{A.6}$$

We can then see that the y-coordinate of the position of this frame is

$$[({}^1p_{15})_y]_{LHS} = ({}^0y_1)^T({}^0p_{05}) \tag{A.7}$$

$$= \begin{bmatrix} -s_1 & c_1 & 0 \end{bmatrix} \begin{bmatrix} p_x - d_6 z_x \\ p_y - d_6 z_y \\ p_z - d_6 z_z \end{bmatrix} \tag{A.8}$$

$$= (p_x - d_6 z_x)(-s_1) + (p_y - d_6 z_y)(c_1) \tag{A.9}$$

$$[({}^1p_{15})_y]_{RHS} = -d_4 \tag{A.10}$$

The equation

$$(p_x - d_6 z_x)(-s_1) + (p_y - d_6 z_y)(c_1) = -d_4 \tag{A.11}$$

is known as a phase-shift equation whose solution can be found as

$$\tan \alpha_1 = \frac{({}^0p_{05})y}{({}^0p_{05})x} \tag{A.12}$$

$$\cos\alpha_2 = \frac{d_4}{R} = \frac{d_4}{\sqrt{({}^0p_{05})_x^2 + ({}^0p_{05})_y^2}} \tag{A.13}$$

$$\theta_1 = \alpha_1 + \alpha_2 + \pi/2 \tag{A.14}$$

$$= \arctan 2(({}^0p_{05})_y, ({}^0p_{05})_x) \pm \cos^{-1}\frac{d_4}{R} + \pi/2 \tag{A.15}$$

We can see that there are generally 2 solutions for θ_1, which correspond to configurations where the shoulder is "left" or "right". Using the function atan2 is essential for insuring correct signs and behavior when $({}^0p_{05})x = 0$. By looking at the Figure A.2, it is easy to see that physically, no configuration is possible which makes $\sqrt{({}^0p_{05})_x{}^2 + ({}^0p_{05})_y{}^2} \leq |d_4| \leq 0$. Thus, both α_1 and α_2 always exist if an inverse solution exists.

Finding θ_5

Given a particular θ_1, we can solve for θ_5. Using the tranformation from frame 1 to frame 6, we can form the equality

$$[({}^0T_1)^{-1}({}^0T_6^d)]_{LHS} = [({}^1T_2)({}^2T_3)({}^3T_4)({}^4T_5)({}^5T_6)]_{RHS} \tag{A.16}$$

We can then see that the y-coordinate of the position of this frame is

$$[({}^1p_{16})_y]_{LHS} = ({}^0y_1)^T({}^0p_{06}) \tag{A.17}$$

$$= \begin{bmatrix} -s_1 & c_1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{A.18}$$

$$= p_x(-s_1) + p_y(c_1) \tag{A.19}$$

$$[({}^1p_{16})_y]_{RHS} = -d_4 - c_5 d_6 \tag{A.20}$$

$$\theta_5 = \pm\cos^{-1}\frac{p_x s_1 - p_y c_1 - d_4}{d_6} \tag{A.21}$$

Again, we find that there are 2 solutions for θ_5, which correspond to configurations where the wrist is "in/down" or out/up". This occurs due to the fact that the joint sum θ_{234} can allow the ${}^1T_5$ to achieve orientations where ${}^1y_5$ is pointing in the same direction, but that ${}^1z_5$ is pointing in the opposite direction.

This flip can then be reversed very simply by the 6^{th} joint.

Finding θ_6

This joint has a solution so long as the argument of $\cos^{-1}$ has magnitude not greater than 1, or $|({}^1p_{16})_y - d_4| \leq |d_6|$. To solve for the 6th joint, we look at the ${}^6y_1$ coordinate axis:

$$[{}^6y_1]_{LHS} = \begin{bmatrix} -x_x s_1 + x_y c_1 \\ -y_x s_1 + y_y c_1 \\ -z_x s_1 + z_y c_1 \end{bmatrix} \tag{A.22}$$

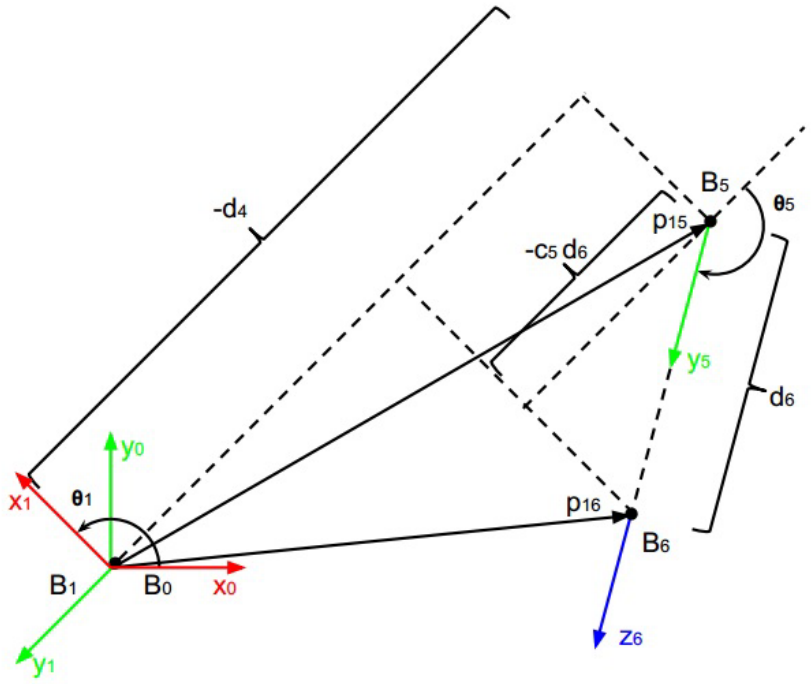


Figure A.3 – Illustration of the geometry of finding θ_5. This is essentially an overhead view of the robot, looking down on the x-y plane.

$$[{}^6y_1]_{RHS} = \begin{bmatrix} -c_6 s_5 \\ s_6 s_5 \\ -c_5 \end{bmatrix} \tag{A.23}$$

As shown in Figure A.4, this equality forms a spherical coordinate expression for the vector $-{}^6y_1$ where θ_6 is the azimuthal angle and θ_5 is the polar angle. The x and y coordinates of this vector form a system which can be easily solved as

$$\theta_6 = \arctan(\frac{-y_x s_1 + y_y c_1}{s_5}, \frac{-x_x s_1 + x_y c_1}{s_5}) \tag{A.24}$$

The denominators of each argument can be replaced by $\text{sign}(s_5)$. This solution is undefined in two circumstances, when both of the numerators are 0 or $s_5 = 0$. Inspection of equations (15) and (16) shows that these conditions actually imply each other. When $s_5 = 0$, we know $c_5 = \pm 1$, which indicates that the joints 2, 3, 4, and 6 are all parallel and the solution is under determined. When this occurs, a desired θ_6 can be supplied to fully determine the system.

Finding θ_3

The final 3 joints can be found easily, understanding that they together form a classical 3R planar arm. Since we have the other 3 joints, we solve for the location of the base and end effector of the 3R arm, and use those equations to solve. The solution has two possible configurations, where the arm is elbow "up" or "down". No solutions exist when the distance to the 4th joint exceeds the sum $|a_2 + a_3|$ or is less than the difference $|a_2 - a_3|$. If $a_2 = a_3$, a displacement singularity exists when $\theta_3 = \pi$, making θ_2 arbitrary.

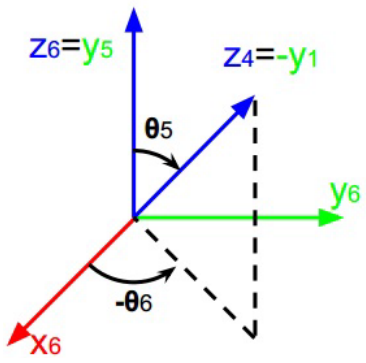


Figure A.4 – Overhead view of the kinematics problem, for computing θ_6.

We can constrict ourselves to look at ${}^1T_4$ (frame 4 in relation to frame 1) because ${}^1T_0$, ${}^4T_5$ and ${}^5T_6$ at this point are known. This transformation is illustrated in the x; z-plane of frame 1 in Figure A.5. From the figure it is clear that the length of the translation $|{}^1p_{14xz}|$ is determined only by θ_3, or similarly by α_3. The angle α_3 can be found by using the law of cosine:

$$\cos\alpha_3 = \frac{(-a_2)^2 + (-a_3)^2 - |{}^1p_{14xz}|^2}{2(-a_2)(-a_3)} \tag{A.25}$$

The relationship between $\cos\alpha_3$ and $\cos\theta_3$ is:

$$cos\theta_3 = cos(\pi - \alpha_3) = -cos(\alpha_3) \tag{A.26}$$

Combining above 2 equations:

$$\theta_3 = \pm \arccos \frac{|{}^1p_{14xz}|^2 - {a_2}^2 - {a_3}^2}{2a_2a_3} \tag{A.27}$$

Finding θ_2

The angle θ_2 can be found as $\alpha_1 - \alpha_2$. Each of these can be found by inspecting Figure A.5 and using atan2 and sine relations:

$$\alpha_1 = \arctan(-{}^1p_{14z}, -{}^1p_{14x}) \tag{A.28}$$

$$\alpha_2 = \arcsin \frac{-a_3 \sin\alpha_3}{|{}^1p_{4xz}|} \tag{A.29}$$

We can replace α_3 with θ_3 by noticing that $\sin\alpha_3 = \sin(\pi - \theta_3) = sin\theta_3$. Combining the equations now give:

$$\theta_2 = \alpha_1 - \alpha_2 = \arctan(-{}^1p_{14z}, -{}^1p_{14x}) - \arcsin\left(\frac{-a_3 \sin\alpha_3}{|{}^1p_{14xz}|}\right) \tag{A.30}$$

Finding θ_4

The last remaining angle θ_4 is defined as the angle from x_3 to x_4 measured about z_4. It can thus easily be derived from the last remaining transformation matrix, ${}^3T_4$, using its first column ${}^3x_4$:

$$\theta_4 = \arctan({}^3x_{4y}, {}^3x_{4x}) \tag{A.31}$$

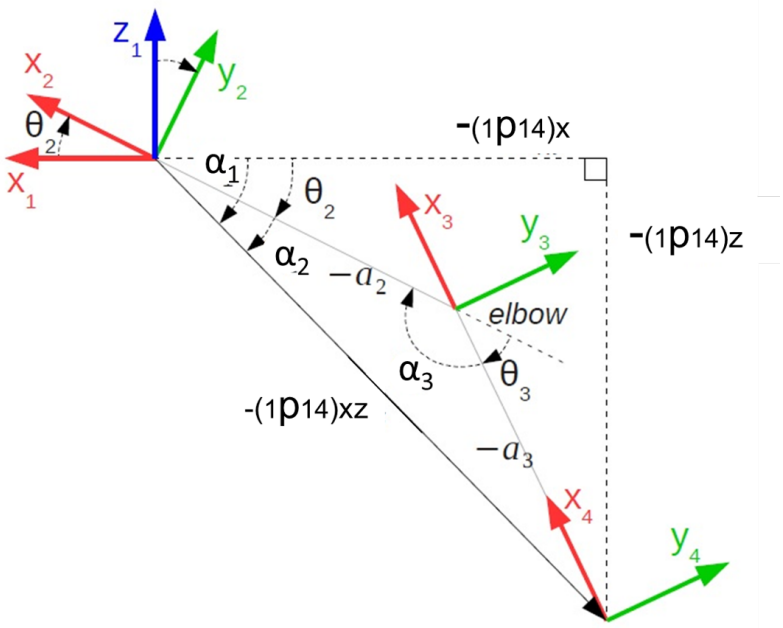


Figure A.5 – Overhead view of the 3R planar manipulator constituted together by joints 2,3 and 4. Kinematics problem, for computing θ_3, θ_2 and θ_4.

Summary

To sum up, a total of 8 solutions exist in general for the general inverse kinematic problem of the UR5: $2\theta_1 \times 2\theta_5 \times \theta_6 \times 2\theta_3 \times \theta_2 \times \theta_4$.

Appendix B

Error analysis for user model

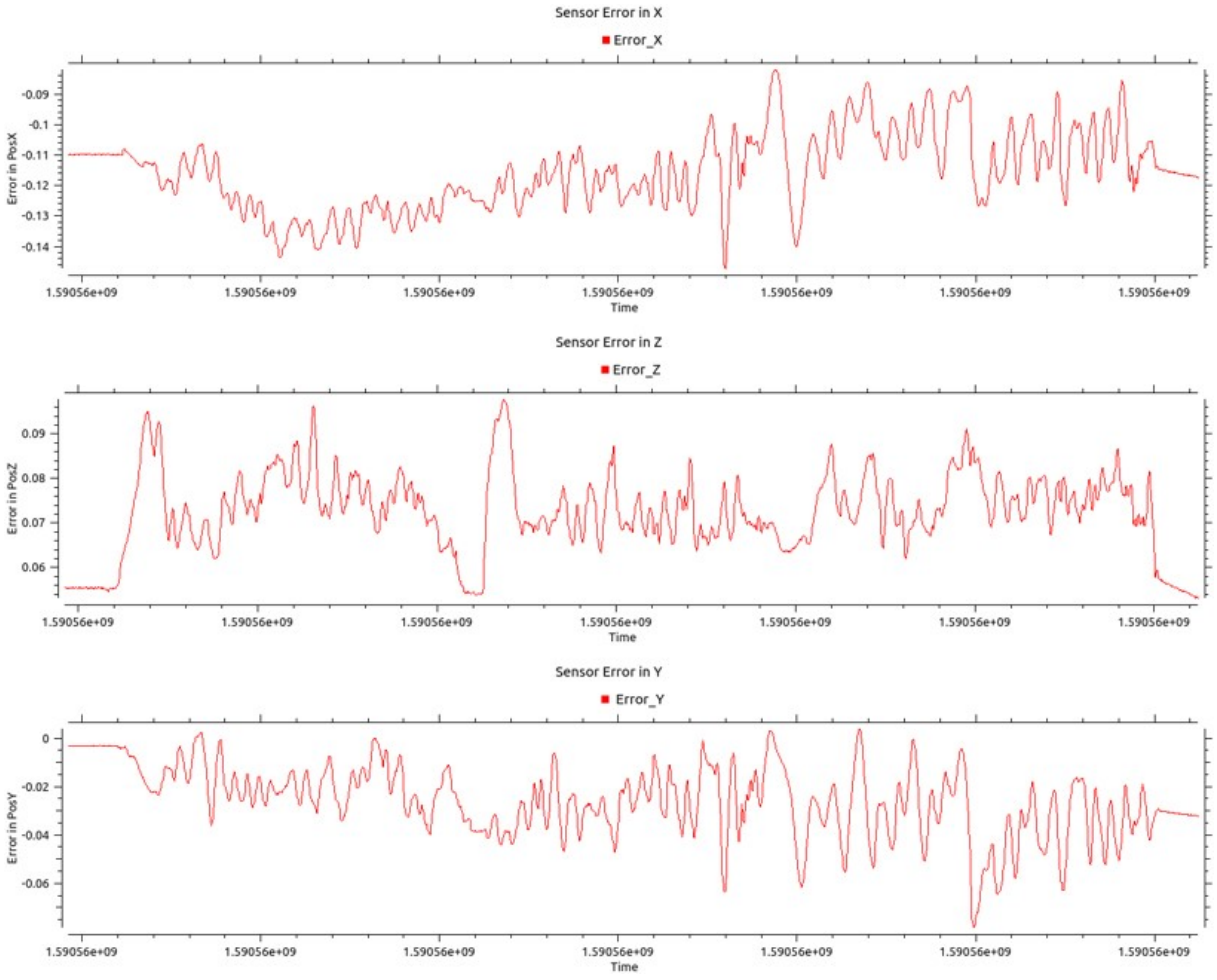


Figure B.1 – Error for position in shoulder sensor in right hand (Y-axis in cm).

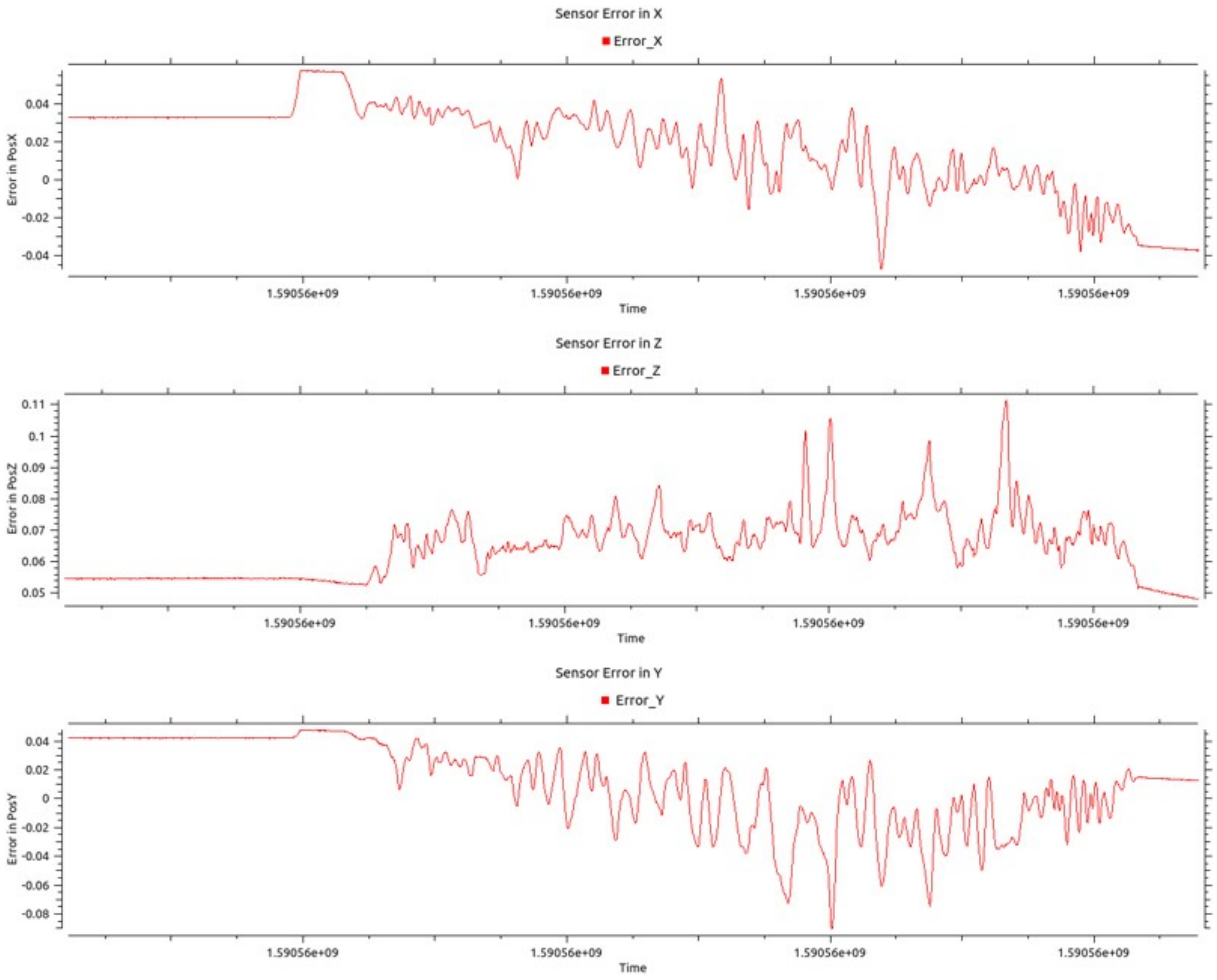


Figure B.2 – Error for position in shoulder sensor in left hand (Y-axis in cm).

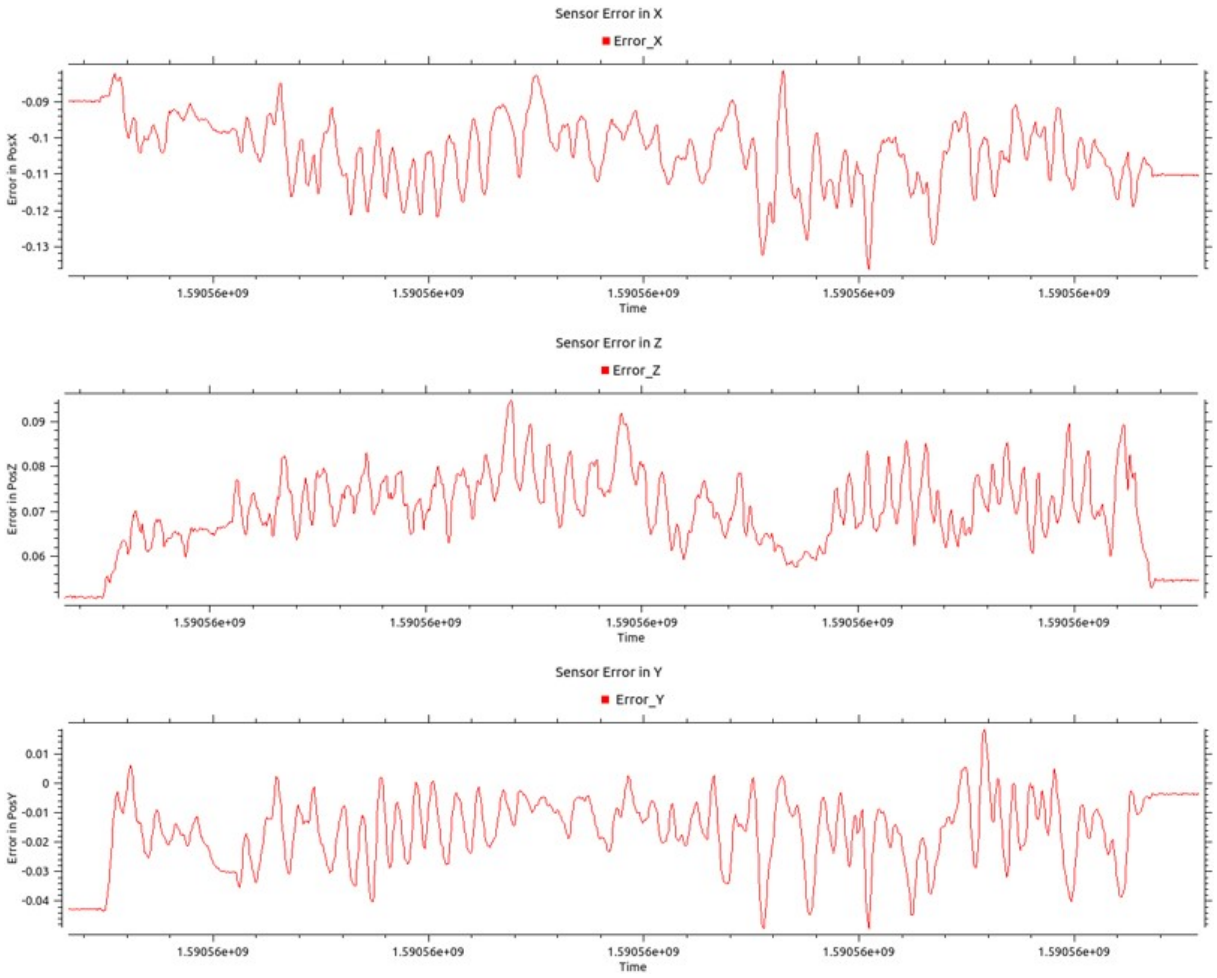


Figure B.3 – Error for position in wrist sensor in right hand (Y-axis in cm).

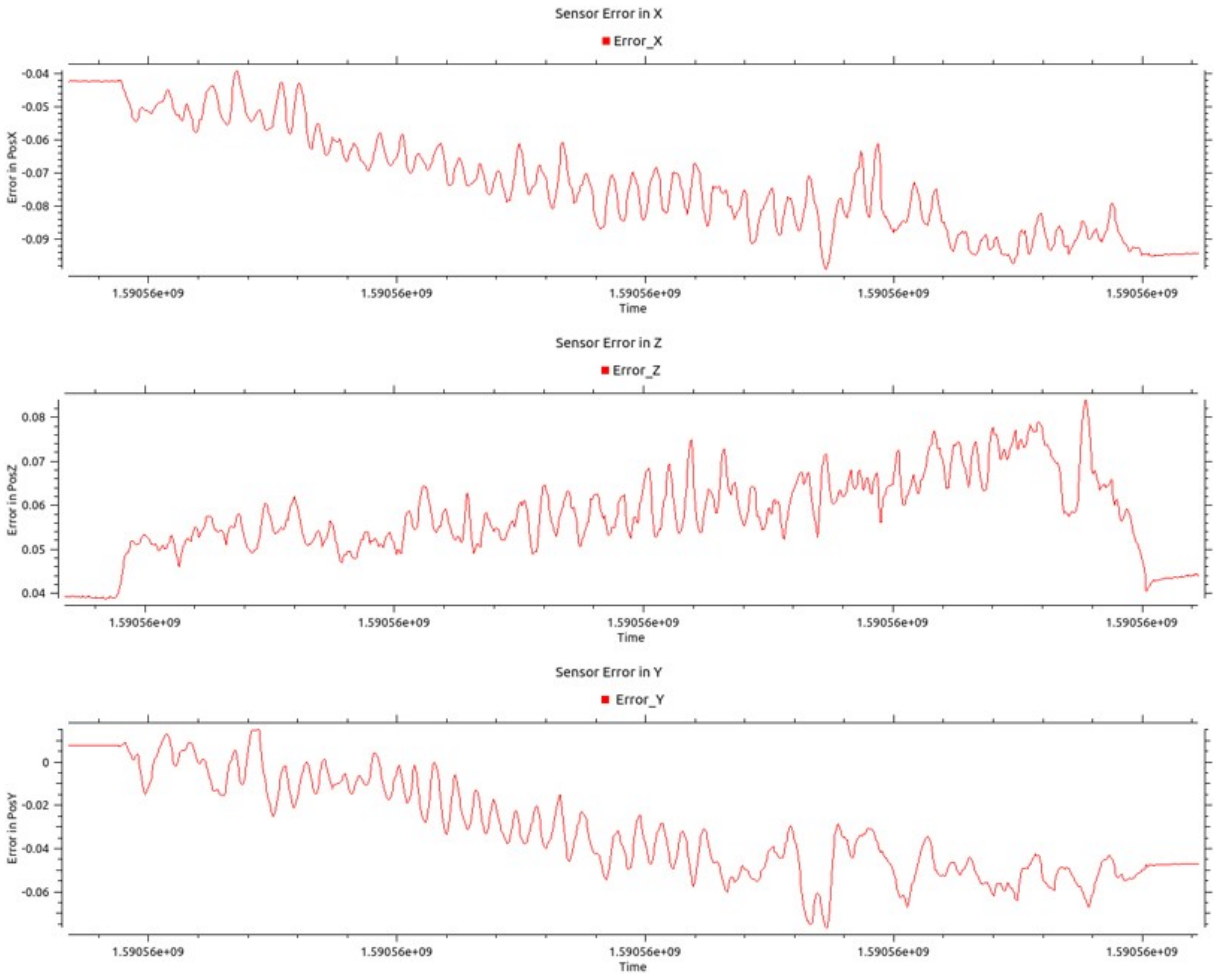


Figure B.4 – Error for position wrist sensor in left hand (Y-axis in cm).

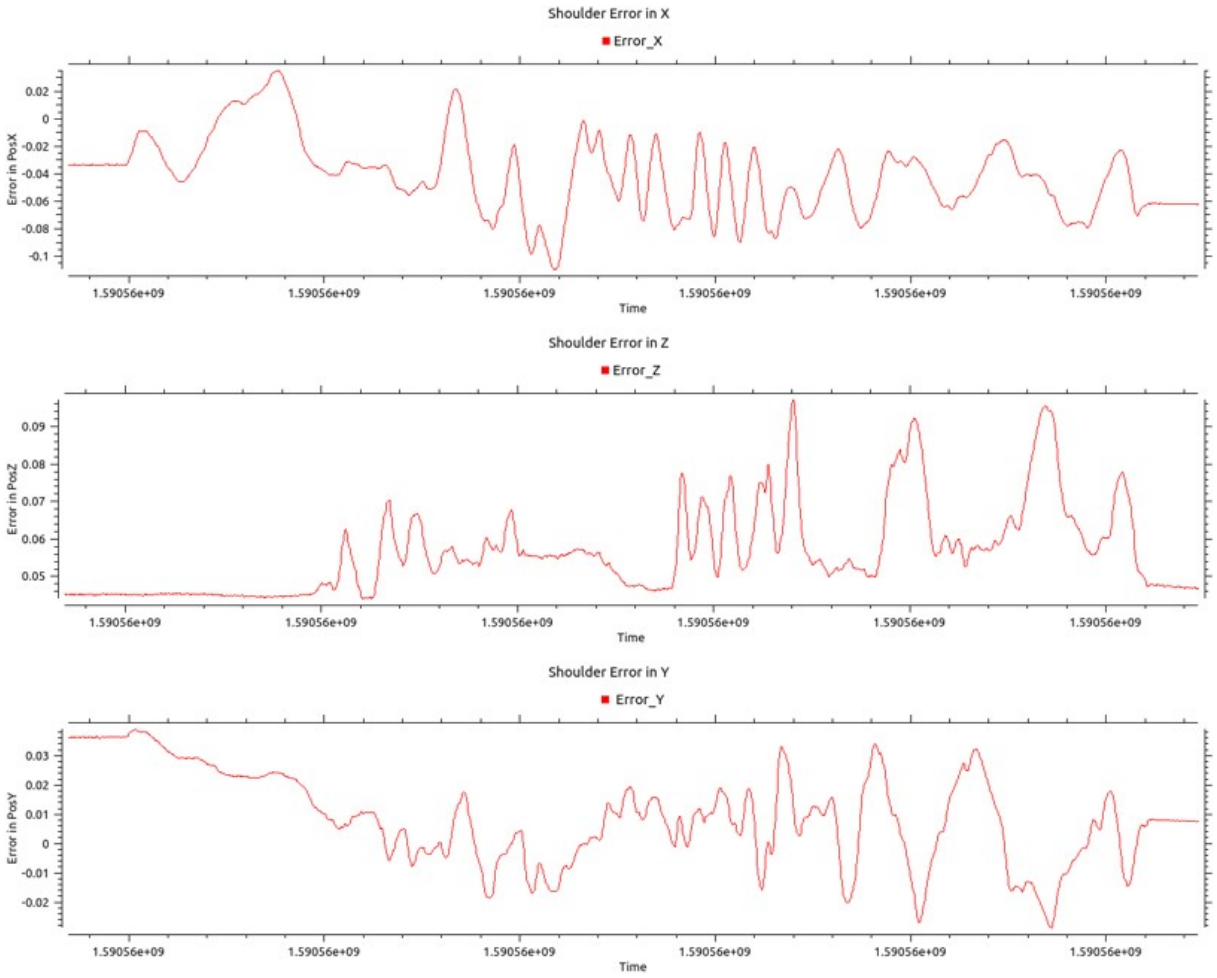


Figure B.5 – Error between actual and estimated value of right shoulder frame.

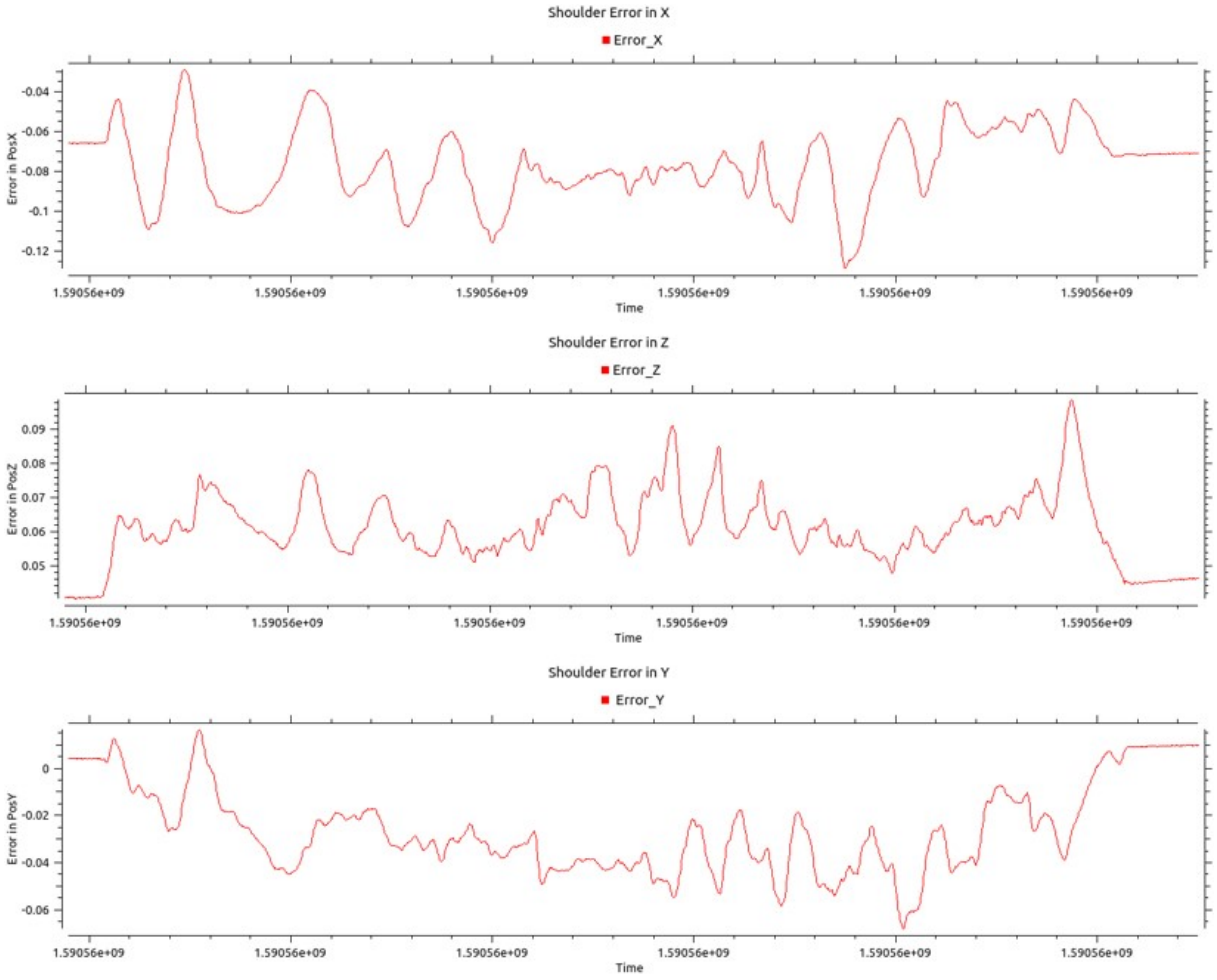


Figure B.6 – Error between actual and estimated value of left shoulder frame.

Appendix C

What is MoveIt ?

Introduction

The MoveIt architecture is based on two main nodes, the *move_group* node, and the *planning_scene* (Figure C.1). The move_group takes care of obtaining the parameters, the setup, and the individual components of the robot model being used, so it can provide to the user *ROS* services and actions for the users to use on the robot. The move_group node can be divided into three parts (Figure C.2):

1. The user interface: This takes care of handling the inputs given by the user in order to perform actions on the robot, this can be done by:
   — GUI through MoveIt RVIz plugin.
   — APIs for C++ or python.
   Based on the actions given, the move_group node is able to solve the robot kinematics and is able to plan a trajectory that takes into account the defined scene, obstacles, and constraints of the system.
2. The ROS Param Server: Which obtains the robot description through *URDF*, *SRDF* and configuration files. Obtaining specifications such as joint limits, velocity limits, or other constraints for the system.
3. The Robot Interface: This provides information obtained from the robot sensors (*2D* or *3D*) or the robot joint states provided by the controllers.

The second element which is the planning_scene uses the *planning scene monitor* to handle the scene in which the robot will be included. This is what will constitute the obstacles or elements the robot has to interact with. Providing information about where they are located, and any updates that happen in the robot's surroundings. It uses as well the robot interface data to make the connection between the data being processed and the objects defined in the scene.

Motion Planning

MoveIt works with motion planners through a plugin interface. This allows MoveIt to communicate with and use different motion planners from multiple libraries, making MoveIt easily extensible. The interface to the motion planners is through a *ROS* Action or service (offered by the move_group node). The default motion planners for move_group are configured using

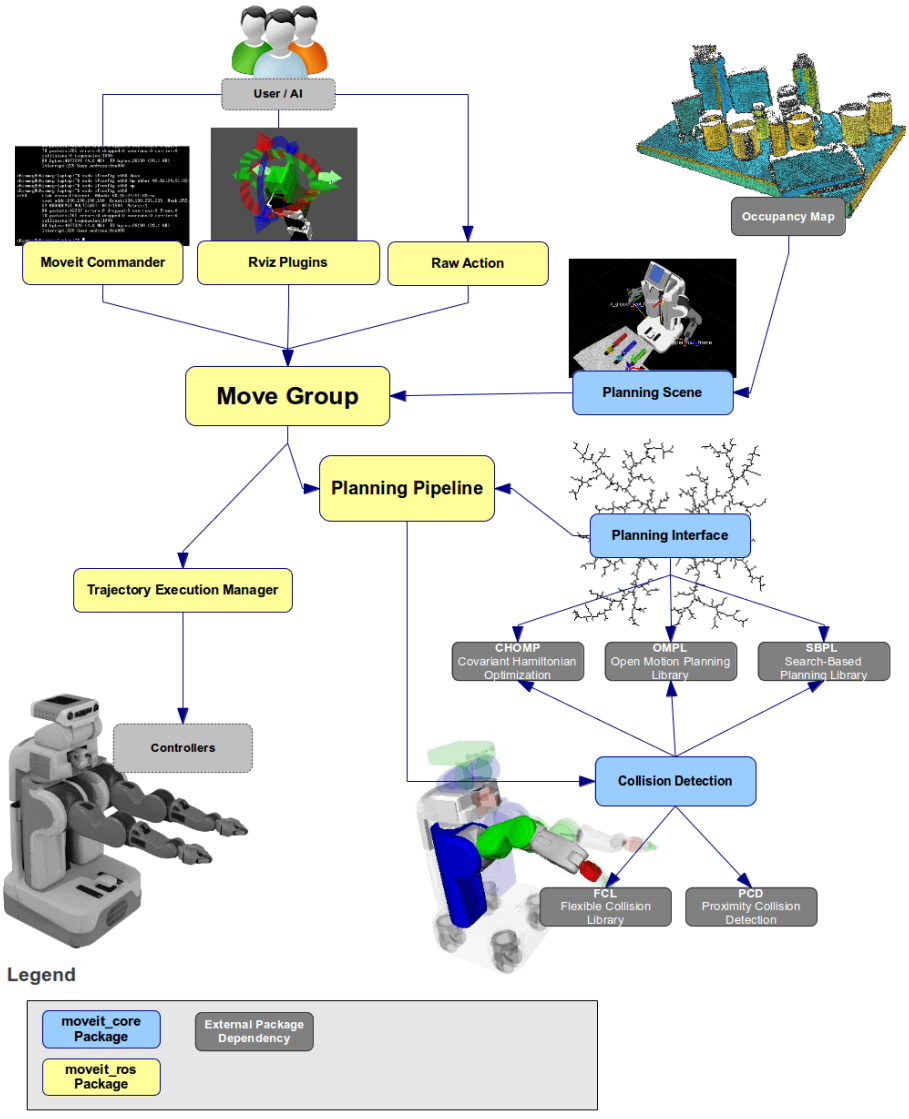


Figure C.1 – MoveIt system architecture.

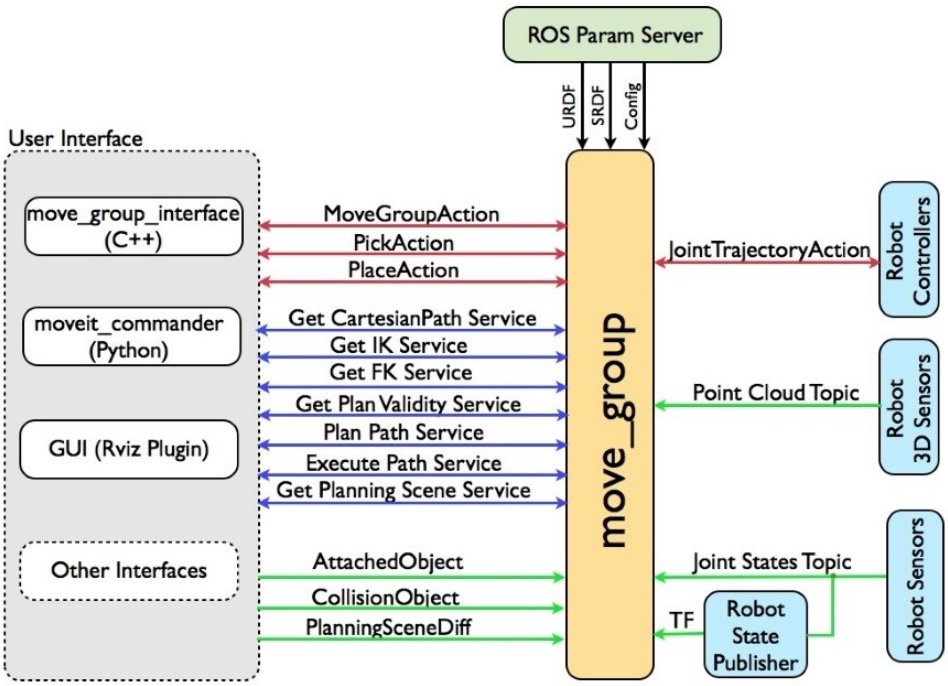


Figure C.2 – Move group architecture.

OMPL [126] and the MoveIt interface to *OMPL* by the MoveIt Setup Assistant. Other planners that are available by default are the *Pilz* industrial motion planner and *CHOMP*.

Within the planners available in the *OMPL* library there are:

1. PRM methods:
   — PRM [69]
   — PRM* [68]
   — LazyPRM [12]
   — LazyPRM* [12] [68]
2. RRT methods:
   — RRT [80]
   — RRT* [68]
   — TRRT [63]
   — BiTRRT [32]
   — LBTRRT [115]
   — RRTConnect [78]
3. Expansive Spacial Trees (EST) methods:
   — EST [60]
   — BiEST [60]

Collision detection

Collision checking in MoveIt is configured inside a Planning Scene using the CollisionWorld object. Fortunately, MoveIt is set up so that users never really have to worry about how

collision checking is happening. Collision checking in MoveIt is mainly carried out using the Flexible Collision Library (*FCL*) [39] package - MoveIt's primary collision checking library.

MoveIt supports collision checking for different types of objects including:

— Meshes

— Primitive shapes (e.g. boxes, cylinders, cones, spheres, and planes)

— Octomap: The Octomap object can be directly used for collision checking (normally built from 3D sensor's data)

Allowed Collision Matrix (ACM): Given that collision, checking is a very expensive operation often accounting for close to 90% of the computational expense during motion planning. The Allowed Collision Matrix encodes a binary value corresponding to the need to check for collision between pairs of bodies (which could be on the robot or in the world). If the value corresponding to two bodies is set to 1 in the ACM, this specifies that a collision check between the two bodies is not needed. This would happen if, e.g., the two bodies are always so far away that they would never collide with each other.

Kinematics

MoveIt uses a plugin infrastructure, especially targeted toward allowing users to write their own inverse kinematics algorithms. Forward kinematics and finding Jacobians are integrated within the RobotState class itself. The default inverse kinematics plugin for MoveIt is configured using the KDL [70] numerical Jacobian-based solver. This plugin is automatically configured by the MoveIt Setup Assistant.

Planning Scene

For the environment and planning scene definition (Figure C.3), MoveIt counts with instances that allow the manipulation and monitoring of the scene to keep it up-to-date. These instances are:

— PlanningSceneInterface: Is responsible for adding and removing objects in the scene.

— PlanningSceneMonitor: Takes care of keeping track of the planning_scene to keep it updated.

The last one of these instances is absolutely necessary to perform collision checking, as we need to ensure that the scene which is being processed is the latest one available.

ROS-Industrial

ROS-Industrial is an open-source project that extends the advanced capabilities of ROS software to industrial relevant hardware and applications. For this project, we used the ROS-Industrial-Universal-Robots [89] meta-package, which provides and facilitates the main configuration files for the usage of the Universal Robots co-bots within the ROS environment, providing the different descriptions of the robot, configuration files such as joint limits, UR kinematics, etc.

This package also facilitates the usage of the robot within MoveIt, providing the setup for its usage in simulation or in real-world implementations.

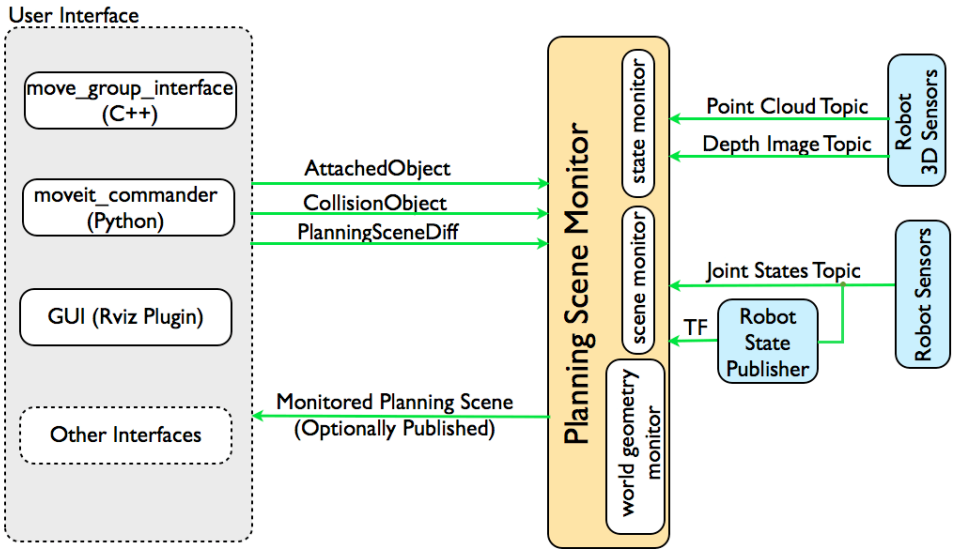


Figure C.3 – Planning scene architecture.

Bibliography

[1] *3D Systems (2018),Touch haptic device*. Feb. 2022. URL: https://www.3dsystems.com/haptics-devices/touch..

[2] Iina Aaltonen, Timo Salmi, and Ilari Marstio. "Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry". In: *Procedia CIRP* 72 (2018), pp. 93–98.

[3] A. Abbadi and V. Pernosil. "Collided Path Replanning in Dynamic Environments UsingRRT and Cell Decomposition Algorithms". In: *International Workshop on Modelling and Simulation for Autonomous Systems* (2015).

[4] Rajesh Aggarwal and Ara Darzi. "From scalpel to simulator: a surgical journey". In: *Surgery* 145.1 (2009), pp. 1–4.

[5] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[6] KN An, LJ Askew, and EY Chao. "Biomechanics and functional assessment of upper extremities". In: *Trends in ergonomics/human factors III*. Vol. 1986. Elsevier Science Publishers North-Holland. 1986, pp. 573–580.

[7] Bruno Araujo et al. "Snake Charmer: Physically Enabling Virtual Objects". In: *Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. ACM. 2016, pp. 218–226.

[8] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: https://doi.org/10.1145/361002.361007.

[9] Massimo Bergamasco. "Haptic interfaces: the study of force and tactile feedback systems". In: *Proceedings 4th IEEE International Workshop on Robot and Human Communication*. IEEE. 1995, pp. 15–20.

[10] Tiziano Bernard et al. "Haptic feedback astronaut suit for mitigating extra-vehicular activity spatial disorientation". In: *AIAA SPACE and Astronautics Forum and Exposition*. 2017, p. 5113.

[11] S James Biggs and Mandayam A Srinivasan. "Haptic interfaces". In: *Handbook of virtual environments* (2002), pp. 93–116.

[12] R. Bohlin and L.E. Kavraki. "Path Planning Using Lazy PRM". In: *IEEE International Conference on Robotics and Automation* (2000), pp. 521–528. DOI: 10.1109/ROBOT.2000.844107.

[13] Aude Bolopion and Stéphane Régnier. "A review of haptic feedback teleoperation systems for micromanipulation and microassembly". In: *IEEE Transactions on automation science and engineering* 10.3 (2013), pp. 496–502.

[14] Diego Borro et al. "A large haptic device for aircraft engine maintainability". In: *IEEE Computer Graphics and Applications* 24.6 (2004), pp. 70–74.

[15] Stuart A Bowyer, Brian L Davies, and Ferdinando Rodriguez y Baena. "Active constraints/virtual fixtures: A survey". In: *IEEE Transactions on Robotics* 30.1 (2013), pp. 138–157.

[16] Thurston L Brooks. "Telerobotic response requirements". In: *1990 IEEE international conference on systems, man, and cybernetics conference proceedings*. IEEE. 1990, pp. 113–120.

[17] Frederick P Brooks Jr et al. "Project GROPEHaptic displays for scientific visualization". In: *ACM SIGGraph computer graphics* 24.4 (1990), pp. 177–185.

[18] Grigore Burdea. "The role of haptics in physical rehabilitation". In: *Haptic Rendering* (2008), pp. 517–529.

[19] Grigore Burdea and Philippe Coiffet. *Virtual reality technology*. John Wiley & Sons, 2003.

[20] Grigore C Burdea. *Force and touch feedback for virtual reality*. John Wiley & Sons, Inc., 1996.

[21] Gianni Campion. *The Synthesis of Three Dimensional Haptic Textures: Geometry, Control, and Psychophysics*. Springer Science & Business Media, 2011.

[22] Anthony Chabrier et al. "Design and experimental evaluation of an infrared instrumentation for haptic interfaces". In: *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE. 2017, pp. 1–6.

[23] Alan Chalmers, David Howard, and Christopher Moir. "Real virtuality: A step change from virtual reality". In: *Proceedings of the 25th Spring Conference on Computer Graphics*. 2009, pp. 9–16.

[24] Adrian David Cheok and Kasun Karunanayaka. *Virtual taste and smell technologies for multisensory internet and virtual reality*. Springer, 2018.

[25] Andrea Cherubini et al. "Collaborative manufacturing with physical human–robot interaction". In: *Robotics and Computer-Integrated Manufacturing* 40 (2016), pp. 1–13.

[26] Michael Cohen and Elizabeth M. Wenzel. "The Design of Multidimensional Sound Interfaces". In: *Virtual Environments and Advanced Interface Design*. USA: Oxford University Press, Inc., 1995, pp. 291–346. isbn: 0195075552.

[27] D. Coleman et al. "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study". In: *Journal of Software Engineering for Robotics* 5(1) (2014), pp. 3–16. url: https://moveit.ros.org/.

[28] C.I. Conolly, J.B. Burns, and R. Weiss. "Path Planning Using Laplaces Equation." In: *IEEE International Conference on Robotics and Automation* (1990).

[29] Damien Couroussé. *Mechanical impedance*. 2007.

[30] *CyberGlove Systems*. Jan. 2022. url: http://www.cyberglovesystems.com/cybertouch.

[31] *Desktop haptic interface*. Feb. 2022. URL: http://www.moog.com/products/haptics-robotics/..

[32] D. Devaurs, T. Simeon, and J. Cortes. "Enhancing the Transition-based RRT to Deal with Complex Cost Spaces". In: *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)* (2013), pp. 4120–4125. DOI: 10.1109/ICRA.2013.6631158.

[33] E. Dijkstra. "A Note on Two Problems in Connexion with Graphs Numerische Mathematik". In: *Journal of Computer and System Sciences - JCSS* (1959).

[34] Adam Drewnowski. "Taste preferences and food intake". In: *Annual review of nutrition* 17.1 (1997), pp. 237–253.

[35] Mark Wm Dubin. *How the brain works*. John Wiley & Sons, 2013.

[36] Michael Eisenstein. "Taste: More than meets the mouth". In: *Nature* 468.7327 (2010), S18–S19.

[37] Abdulmotaleb El Saddik. *Haptics rendering and applications*. BoD–Books on Demand, 2012.

[38] Abdulmotaleb El Saddik. "The Potential of Haptics Technologies". In: *IEEE Instrumentation Measurement Magazine* 10.1 (2007), pp. 10–17. DOI: 10.1109/MIM.2007.339540.

[39] *FCL flexible collision library*. June 2019. URL: https://github.com/flexible-collision-library/fcl.

[40] *FeelReal*. Jan. 2022. URL: https://feelreal.com.

[41] C. Feichtenhofer, A. Pinz, and A. Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 1933–1941. DOI: 10.1109/CVPR.2016.213.

[42] Alessandro Filippeschi et al. "Encountered-type haptic interface for virtual interaction with real objects based on implicit surface haptic rendering for remote palpation". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5904–5909.

[43] *Force Dimension (2018). sigma.7*. Feb. 2022. URL: http://www.forcedimension.com/products/sigma-7/overview..

[44] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. "An Algorithm for Finding Best Matches in Logarithmic Expected Time". In: *ACM Trans. Math. Softw.* 3.3 (Sept. 1977), pp. 209–226. ISSN: 0098-3500. DOI: 10.1145/355744.355745. URL: https://doi.org/10.1145/355744.355745.

[45] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), pp. 2997–3004. DOI: 10.1109/IROS.2014.6942976.

[46] A. Gasparetto et al. "Path Planning and Trajectory Planning Algorithms: A General Overview". In: *Motion and Operation Planning of Robotic Systems* 29 (2015).

[47] G. Ghinea, F. Andres, and S. Gulliver. *Multiple Sensorial Media Advances and Applications: New Developments in MulSeMedia*. Premier Reference Source. Information Science Reference, 2012. ISBN: 9781609608217. URL: https://books.google.fr/books?id=PjyOngEACAAJ.

[48] Franck Gonzalez, Wael Bachta, and Florian Gosselin. "Smooth transition-based control of encounter-type haptic devices". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 291–297.

[49] Florian Gosselin et al. "Haptic systems for training sensorimotor skills: a use case in surgery". In: *Robotics and Autonomous Systems* 61.4 (2013), pp. 380–389.

[50] V.K. Guda, D. Chablat, and Chevallereau C. "Safety in a Human Robot Interactive: Application to Haptic Perception". In: *International Conference on Human-Computer Interaction HCII 2000*. 2020.

[51] Blake Hannaford and Allison M Okamura. "Haptics". In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1063–1084.

[52] *HaptX*. Jan. 2022. URL: https://haptx.com.

[53] P.E. Hart, N.J. Nilson, and Raphael B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4(2) (1968), pp. 100–107.

[54] Paul Havig, John McIntire, and Eric Geiselman. "Virtual reality in a cave: limitations and the need for HMDs?" In: *Head-and helmet-mounted displays XVI: Design and applications*. Vol. 8041. SPIE. 2011, pp. 58–63.

[55] Mary Hayhoe et al. "Visual short-term memory and motor planning." eng. In: *Progress in brain research* 140 (2002), pp. 349–363. ISSN: 0079-6123 (Print). DOI: 10.1016/S0079-6123(02)40062-3.

[56] Vincent Hayward and Karon E MacLean. "Do it yourself haptics: part I". In: *IEEE Robotics & Automation Magazine* 14.4 (2007), pp. 88–104.

[57] Mort Heilig. "Enter the Experimental Revolution". In: *Proceeding of Cyberarts Conference*. 1992, pp. 292–305.

[58] Evert Helms, Rolf Dieter Schraft, and M Hagele. "rob@ work: Robot assistant in industrial environments". In: *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*. IEEE. 2002, pp. 399–404.

[59] Koichi Hirota and Michitaka Hirose. "Development of surface display". In: *Proceedings of IEEE Virtual Reality Annual International Symposium*. IEEE. 1993, pp. 256–262.

[60] David. Hsu, Jean-Claude. Latombe, and Rajeev. Motwani. "Path planning in expansive configuration spaces". In: *International Journal of Computational Geometry & Applications* 9.4-5 (1999), pp. 495–512. DOI: 10.1142/S0218195999000285.

[61] Hsin-Yu Huang et al. "Haptic-Go-Round: A Surrounding Platform for Encounter-Type Haptics in Virtual Reality Experiences". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–10. ISBN: 9781450367080. DOI: 10.1145/3313831.3376476. URL: https://doi.org/10.1145/3313831.3376476.

[62] TS ISO. "15066: Robots and robotic devices-Collaborative robots". In: *Geneva: ISO copyright office* (2016).

[63] L. Jaillet, J. Cortes, and T. Simeon. "Sampling-Based Path Planning on Configuration-Space Costmaps". In: *IEEE Transactions on Robotics* 26.4 (Aug. 2010). DOI: 10.1109/TRO.2010.2049527.

[64] Jason Jerald. *The VR book: Human-centered design for virtual reality*. Morgan & Claypool, 2015.

[65] K. Jin-Oh and P.K. Khosla. "Real-Time Obstacle Avoidance Using Harmonic Potential Functions". In: *IEEE Transactions on Robotics and Automation* 8(3) (1992), pp. 338–349.

[66] Roland S. Johansson et al. "Eye–Hand Coordination in Object Manipulation". In: *Journal of Neuroscience* 21.17 (2001), pp. 6917–6932. doi: 10.1523/jneurosci.21-17-06917.2001. url: https://app.dimensions.ai/details/publication/pub.1074868523andhttp://www.jneurosci.org/content/21/17/6917.full.pdf.

[67] M. Jordan and A. Perez. "Optimal Bidirectional Rapidly-Exploring Random Trees". In: *MIT-CSAIL-TR-2013-021* (2013).

[68] S. Karaman and E. Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. doi: 10.1177/0278364911406761.

[69] L.E. Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." In: *IEEE Transactions on Robotics and Automation* 12(4) (1996), pp. 566–80.

[70] *KDL kinematics and dynamics library (kdl)*. June 2019. url: http://wiki.ros.org/kdl.

[71] Wisama Khalil and Etienne Dombre. *Modeling identification and control of robots*. CRC Press, 2002.

[72] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *IEEE International Conference on Robotics and Automation* (1985).

[73] Yaesol Kim, Hyun Jung Kim, and Young J Kim. "Encountered-type haptic display for large VR environment using per-plane reachability maps". In: *Computer Animation and Virtual Worlds* 29.3-4 (2018), e1814.

[74] Yaesol Kim et al. "Synthesizing the roughness of textured surfaces for an encountered-type haptic display using spatiotemporal encoding". In: *IEEE Transactions on Haptics* 14.1 (2020), pp. 32–43.

[75] B. Kolb and I.Q. Whishaw. *An Introduction to Brain and Behavior, Third Edition*. Worth Publishers, 2005. isbn: 9781429281478.

[76] Jörg Krüger, Terje K Lien, and Alexander Verl. "Cooperation of human and machines in assembly lines". In: *CIRP annals* 58.2 (2009), pp. 628–646.

[77] Ernst Kruijff, Dieter Schmalstieg, and Steffi Beckhaus. "Using Neuromuscular Electrical Stimulation for Pseudo-Haptic Feedback". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 316–319. isbn: 1595933212. doi: 10.1145/1180495.1180558. url: https://doi.org/10.1145/1180495.1180558.

[78] James. Kuffner and Steven.M. Lavalle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings of the 2000 IEEE International Conference on Robotics & Automation* (Apr. 2000), pp. 995–1001. doi: 10.1109/ROBOT.2000.844730.

[79] G. Santhosh Kumar and G. Sony Bhavani. "A new dimension of immersiveness into virtual reality through haptic technology". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. 2017, pp. 1651–1655. doi: 10.1109/ICPCSI.2017.8391994.

[80] S. LaValle and J.J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* (2001).

[81] Susan J Lederman and Roberta Klatzky. "Haptic Exploration and Object". In: *Vision and action: The control of grasping* 2 (1990), p. 98.

[82] Philip Long et al. "An industrial security system for human-robot coexistence". In: *Industrial Robot: An International Journal* 45.2 (2018), pp. 220–226.

[83] T. Lozano-Perez. "Spatial planning: a configuration space approach". In: *IEEE Transactions on Computers* (1983).

[84] Anderson Maciel et al. "Multi-Finger Haptic Rendering of Deformable Objects, In. Tenth Eurographics Symposium on Virtual Environments". In: *Eurographics*. CONF. 2004, pp. 105–111.

[85] A. Makhal and A. K. Goins. "Reuleaux: Robot Base Placement by Reachability Analysis". In: *ArXiv e-prints* (Oct. 2017). arXiv: 1710.01328 [cs.RO].

[86] Dan Maynes-Aminzade. "Edible bits: Seamless interfaces between people, data and food". In: *Conference on Human Factors in Computing Systems (CHI'05)-Extended Abstracts*. Citeseer. 2005, pp. 2207–2210.

[87] William A McNeely. "Robotic graphics: a new approach to force feedback for virtual reality". In: *Proceedings of IEEE Virtual Reality Annual International Symposium*. IEEE. 1993, pp. 336–341.

[88] Víctor Rodrigo Mercado, Maud Marchal, and Anatole Lécuyer. "ENTROPiA: Towards Infinite Surface Haptic Displays in Virtual Reality Using Encountered-Type Rotating Props". In: *IEEE Transactions on Visualization and Computer Graphics* (2019).

[89] F. Messmer et al. *ROS-Indutrial-Universal-Robots*. https://github.com/ros-industrial/universal_robot. 2022.

[90] Daniel Mestre et al. "Immersion et présence". In: *Le traité de la réalité virtuelle. Paris: Ecole des Mines de Paris* (2006), pp. 309–38.

[91] Daniel R Mestre. "CAVE versus head-mounted displays: ongoing thoughts". In: *Electronic Imaging* 2017.3 (2017), pp. 31–35.

[92] Land Michael, Mennie Neil, and Rusted Jennifer. "The Roles of Vision and Eye Movements in the Control of Activities of Daily Living". In: *Perception* 28.11 (1999). PMID: 10755142, pp. 1311–1328. DOI: 10.1068/p2935. URL: https://doi.org/10.1068/p2935.

[93] Matjaž Mihelj, Domen Novak, and Samo Beguš. "Virtual reality technology and applications". In: (2014).

[94] A. MONTAGU. "Touching, The human significance of the skin". In: *Perennial Library* (1972), pp. 98–99. URL: https://ci.nii.ac.jp/naid/10007744975/en/.

[95] *MPB Technologies Inc. (2018b). How do you Choose a Haptic Device?* Feb. 2022. URL: http://www.mpb-technologies.ca/mpbt/mpbt_web_2009/_en/resources/articles/Howdoyouchooseahapticdevice.pdf.

[96] Stanley Mugisha et al. "Safe collaboration between human and robot in a context of intermittent haptique interface". In: *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. 2021.

[97] K. Naderi and J. Rajamaki. "RT-RRT*: a real-time path planning algorithm based on RRT*". In: *Conference: the 8th ACM SIGGRAPH Conference* (2015).

[98] Allison M Okamura. "Haptic feedback in robot-assisted minimally invasive surgery". In: *Current opinion in urology* 19.1 (2009), p. 102.

[99] Esben H Ostergaard. "Lightweight robot for everybody [industrial activities]". In: *IEEE robotics & automation magazine* 19.4 (2012), pp. 17–18.

[100] Ezgi Özcan. "Analysis of Immersive Virtual Reality through Senses". In: (2020).

[101] Richard Palluel-Germain et al. "A visuo-haptic device-telemaque-increases kindergarten children's handwriting acquisition". In: *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*. IEEE. 2007, pp. 72–77.

[102] Jeff Pelz, Mary Hayhoe, and Russ Loeber. "The coordination of eye, head, and hand movements in a natural task". In: *Experimental Brain Research* 139.3 (2001), pp. 266–277. issn: 1432-1106. doi: 10.1007/s002210100745. url: https://doi.org/10.1007/s002210100745.

[103] Jérôme Perret and Emmanuel Vander Poorten. "Touching virtual reality: a review of haptic gloves". In: *ACTUATOR 2018; 16th International Conference on New Actuators*. VDE. 2018, pp. 1–5.

[104] Javier Posselt et al. "Toward virtual touch: investigating encounter-type haptics for perceived quality assessment in the automotive industry". In: *Proceedings of the 14th annual EuroVR conference*. Laval, France, 2017, pp. 11–13.

[105] Pinyo Puangmali et al. "State-of-the-art in force and tactile sensing for minimally invasive surgery". In: *IEEE Sensors Journal* 8.4 (2008), pp. 371–381.

[106] Eleanora P Westebring-van der Putten et al. "Haptics in minimally invasive surgery–a review". In: *Minimally Invasive Therapy & Allied Technologies* 17.1 (2008), pp. 3–16.

[107] *Quanser Inc. (2018). High definition haptic device.* Feb. 2022. url: https://www.quanser.com/products/hd2-high-definition-haptic-device/..

[108] Nimesha Ranasinghe and Ellen Yi-Luen Do. "Digital lollipop: Studying electrical stimulation on the human tongue to simulate taste sensations". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 13.1 (2016), pp. 1–22.

[109] Miriam Reiner. "The role of haptics in immersive telecommunication environments". In: *IEEE Transactions on Circuits and Systems for Video Technology* 14.3 (2004), pp. 392–401.

[110] D.E. Rimon E. Koditschek. "Exact robot navigation using artificial potential functions." In: *IEEE Transactions on Robotics and Automation* 8(5) (1992), pp. 501–518.

[111] Gabriel Robles-De-La-Torre. "The importance of the sense of touch in virtual and real environments". In: *Ieee Multimedia* 13.3 (2006), pp. 24–30.

[112] John W. Ruffner. "Multimedia and Virtual Reality: Designing Multisensory User Interfaces by Alistair Sutcliffe". In: *Ergonomics in Design* 13.1 (2005), pp. 29–29. doi: 10.1177/106480460501300108. eprint: https://doi.org/10.1177/106480460501300108. url: https://doi.org/10.1177/106480460501300108.

[113] Mikel Sagardia et al. “VR-OOS: The DLR’s virtual reality simulator for telerobotic on-orbit servicing with haptic feedback”. In: *2015 IEEE Aerospace Conference*. IEEE. 2015, pp. 1–17.

[114] Steeven Villa Salazar et al. “Altering the Stiffness, Friction, and Shape Perception of Tangible Objects in Virtual Reality Using Wearable Haptics”. In: *IEEE Transactions on Haptics (ToH)* (2020).

[115] Oren Salzman and Dan. Halperin. “Asymptotically near-optimal RRT for fast, high-quality motion planning”. In: *IEEE Transactions on Robotics* 32.3 (Apr. 2016), pp. 473–483.

[116] Ganesh Sankaranarayanan et al. “Role of haptics in teaching structural molecular biology”. In: *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.* IEEE. 2003, pp. 363–366.

[117] M. Seda. “Roadmap Methods vs. Cell Decomposition in Robot Motion Planning”. In: *Proceedings of the 6th WSEAS International Conference on Signal Processing* (2007).

[118] JEE Sharpe. “Technical and human operational requirements for skill transfer in teleoperations”. In: *Proceedings of the International Symposium on Teleoperation and Control*. 1988, pp. 175–187.

[119] J. Sheng et al. “An Improved Artificial Potential Field Algorithm for Virtual Human Path Planning”. In: *Edutainment 2010, Lecture Notes in Computer Science (LNCS)* 6249 (2010), pp. 592–601.

[120] William R Sherman and Alan B Craig. *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann, 2018.

[121] Karun B Shimoga. “A survey of perceptual feedback issues in dexterous telemanipulation. II. Finger touch feedback”. In: *Proceedings of IEEE Virtual Reality Annual International Symposium*. IEEE. 1993, pp. 271–279.

[122] N. Sleumer and N. Tschichold-gurman. “Exact Cell Decomposition of Arrangements used for Path Planning in Robotics”. In: *Technical Report* (2000).

[123] Jeroen B. J. Smeets, Mary M. Hayhoe, and Dana H. Ballard. “Goal-directed arm movements change eye-head coordination”. In: *Experimental Brain Research* 109.3 (1996), pp. 434–440. issn: 1432-1106. doi: 10.1007/BF00229627. url: https://doi.org/10.1007/BF00229627.

[124] Alexander Stamenkovic. “Do postural constraints affect eye, head, and arm coordination?” In: *Journal of neurophysiology* 120.4 (2018), pp. 2066–2082. doi: 10.1152/jn.00200.2018.

[125] Standard International Organization for Standardization. “Robots and robotic devices - safety requirements for industrial robots - part I: Robots, DIN EN ISO 10218-1”. en. In: (2021).

[126] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. “The Open Motion Planning Library”. In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012), pp. 72–82. doi: 10.1109/MRA.2012.2205651. url: https://ompl.kavrakilab.org.

[127] Susumu Tachi. “A construction method of virtual haptics space”. In: *Proc. of the ICAT’94 (4th International Conference on Artificial Reality and Tele-Existence)*. 1994, pp. 131–138.

[128] Russell M Taylor. "Haptics for scientific visualization". In: *ACM SIGGRAPH 2005 Courses*. 2005, 174–es.

[129] Marija Tomić et al. "Human to humanoid motion conversion for dual-arm manipulation tasks". In: *Robotica* 36.8 (2018), pp. 1167–1187.

[130] *Universal Robot*. https://www.universal-robots.com/fr/. Accessed: 2022-03-23. 2022. URL: https://www.universal-robots.com.

[131] *UR modern driver*. 2022. URL: https://github.com/ros-industrial/ur_modern_driver.

[132] ÅB Vallbo et al. "Microstimulation of single tactile afferents from the human hand: Sensory attributes related to unit type and properties of receptive fields". In: *Brain* 107.3 (1984), pp. 727–749.

[133] *Vaqso*. Jan. 2022. URL: https://vaqso.com.

[134] *Virtuose 6d*. Feb. 2022. URL: https://www.haption.com/fr/products-fr/virtuose-6d-fr.html..

[135] R.A. Volpe. "Real-time obstacle avoidance for manipulators and mobile robots". PhD thesis. 1990.

[136] R.A. Volpe and P. Khosla. "Manipulator control with superquadric artificial potential functions: theory and experiments". In: *IEEE Transactions on Systems, Man, and Cybernetics* 20(6) (1990), pp. 1423–1436.

[137] Xi Vincent Wang, A Seira, and Lihui Wang. "Classification, personalised safety framework and strategy for human-robot collaboration". In: *Proceedings of the International Conference on Computers and Industrial Engineering., Auckland, New Zealand*. 2018.

[138] Richard M. Warren. *Auditory Perception: An Analysis and Synthesis*. 3rd ed. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511754777.

[139] Pingjun Xia. "Haptics for product design and manufacturing simulation". In: *IEEE transactions on haptics* 9.3 (2016), pp. 358–375.

[140] Y. Yokokohji, R.L. Hollis, and T. Kanade. "What you can see is what you can feel-development of a visual/haptic interface to virtual environment". In: *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*. IEEE. 1996, pp. 46–53. DOI: 10.1109/VRAIS.1996.490509.

[141] Yasuyoshi Yokokohji, Ralph L Hollis, and Takeo Kanade. "WYSIWYF display: A visual/haptic interface to virtual environment". In: *Presence* 8.4 (1999), pp. 412–434.

[142] Yasuyoshi Yokokohji, Junji Kinoshita, and Tsuneo Yoshikawa. "Path planning for encountered-type haptic devices that render multiple objects in 3d space". In: *Proceedings IEEE Virtual Reality 2001*. IEEE. 2001, pp. 271–278.

[143] Yasuyoshi Yokokohji et al. "Design and path planning of an encountered-type haptic display for multiple fingertip contacts based on the observation of human grasping behavior". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 2. IEEE. 2004, pp. 1986–1991.

[144] Tsuneo Yoshikawa and Akihiro Nagura. "A touch and force display system for haptic interface". In: *Proceedings of international conference on robotics and automation*. Vol. 4. IEEE. 1997, pp. 3018–3024.

Titre : Contributions à l'utilisation de cobots comme interfaces haptiques à contact intermittent en réalité virtuelle.

Mots clés : sécurité, interface à contact intermittent, prédiction de l'intention humaine, planification de trajectoires, collaboration homme-robot, réalité virtuelle.

Résumé : La réalité virtuelle (RV) est de plus en plus utilisée dans des simulations industrielles mais la possibilité de toucher les objets manque rapidement par exemple pour juger de la qualité perçue dans la conception de véhicule automobile. Les interfaces haptiques actuels ne permettent de restituer aisément la notion de texture, l'approche envisagée est donc une interface à contact intermittent. Un cobot vient positionner une surface mobile à l'endroit du contact avec un objet virtuel pour permettre un contact physique avec la main de l'opérateur.

Les contributions de cette thèse portent sur plusieurs aspects : le placement du robot, la modélisation de l'opérateur, la gestion du déplacement et de la vitesse du robot et la détection des intentions de l'opérateur.

Le placement du robot est choisi pour permettre d'atteindre les différentes zones de travail et pour assurer une sécurité passive en rendant impossible au robot de heurter la tête et le buste de l'opérateur en position normale de travail, i.e. assis dans un fauteuil. Un modèle de l'utilisateur, incluant un torse et des bras, est conçu et testé pour suivre les mouvements de l'utilisateur en temps réel.

L'interaction est possible sur un ensemble de pose prédéfinies que l'utilisateur enchaine comme il le désire. Différentes stratégies sont proposées pour prédire les intentions de l'utilisateur. Les aspects clés de la prédiction sont basés sur la direction du regard et la position de la main de l'utilisateur. Une étude expérimentale ainsi que l'analyse qui en découle montrent l'apport de la prise en compte de la direction du regard. L'intérêt d'introduire des points dit « de sécurité » pour éloigner le robot de l'opérateur et permettre des déplacements rapides du robot est mis en évidence.

Title: Contributions to utilize a Cobot as intermittent contact haptic interfaces in virtual reality.

Keywords: safety, intermittent contact interface, human intention prediction, trajectory planning, human robot collaboration, virtual reality.

Abstract: Virtual reality (VR) is evolving and being used in industrial simulations but the possibility to touch objects is missing, for example to judge the perceived quality in the design of a car. The current haptic interfaces do not allow to easily restore the notion of texture, therefore an approach is considered "intermittent contact interface" to achieve this. A cobot positions a mobile surface at the point of contact with a virtual object to allow physical contact with the operator's hand.

The contributions of this thesis concern several aspects: the placement of the robot, the modeling of the operator, the management of the displacement and the speed of the robot and the detection of the operator's intentions.

The placement of the robot is chosen to allow reaching the different working areas and to ensure passive safety by making it impossible for the robot to hit the head and chest of the operator in a normal working position, i.e. sitting in a chair. A model of the user, including a torso and arms, is designed and tested to follow the user's movements in real time

Interaction is possible on a set of predefined poses that the user chains together as desired. Different strategies are proposed to predict the user's intentions. The key aspects of the prediction are based on the gaze direction and the hand position of the user. An experimental study as well as the resulting analysis show the contribution of taking into account the gaze direction. The interest of introducing "safety" points to move the robot away from the operator and allow fast robot movements is highlighted.