Année 2024

MASTER Ingénierie des Systèmes Complexes

Parcours Microsystèmes – Instrumentation Embarquée Robotique

Mémoire de stage master

Design and control of a training device for needle insertion

Luis MALDONADO

Laboratoire ICUBE
1 Pl. de l'Hôpital, 67000 Strasbourg

École Nationale Supérieur de Mécanique et des Microtechniques

26 Rue de l'Epitaphe - 25030 Besançon - France - Tél. 03 81 40 29 03 www.ens2m.fr

Stagiaire :	*Nom : MALDONADO.* *Prénom : Luis*
Entreprise :	*Nom : Laboratoire ICUBE.* *Adresse : 1 Pl. de l'Hôpital,* *.67000.Strasbourg……* *……* *Téléphone : +33 (0)3 68 85 46 16...* *Site internet : https://icube.unistra.fr/*
Sujet du stage :	*Design and control of a training device for needle insertion.* *……* *……*
Mots clés :	*Robotics, Interventional radiology, training, haptics*

Clauses de diffusion

Classement du mémoire	***- Soutenance -***	***Signature du responsable dans l'entreprise***
Confidentiel ☐ Public ☒ Si **public**, le rapport sera conservé pendant **3 ans** à compter de la date de soutenance. Une sélection de mémoires pourra être mise à disposition à titre d'exemple à la bibliothèque pour consultation sur place. Dans le cas où ce mémoire serait sélectionné, en autorisez-vous la consultation ? Oui ☒ Non ☐ ***L'autorisation de consultation donnée pourra à tout moment, être retirée. A charge pour l'auteur ou le tuteur d'en aviser la Cellule stages => stages@ens2m.fr***	Publique ☒ Huis –clos ☐ Date de la soutenance : ……	NOM : Hassan OMRAN Signature : Signature du stagiaire :

(*) Le rapport de stage sera accepté par l'ENSMM uniquement s'il contient ce feuillet validé par le responsable du stagiaire dans l'entreprise.

Cadre réservé à l'administration

Dépôt service des stages le :	Réception bibliothèque le :

Chapter 1: Acknowledgments

First of all I would like to express my sincere gratitude to M. Omran HASSAN, my internship supervisor, for his invaluable guidance, support, and encouragement throughout the duration of this internship.

My heartfelt thanks also go to M. Bernard BAYLE, director of the RDH team at the ICube laboratory, for providing me with the opportunity to be part of this dynamic team and for his insightful advice.

I would like to extend my deepest gratitude to M. Thibault POIGNONEC, research engineer at the ICube lab, for his constant support with the ROS2 component of this project. His expertise was fundamental in helping me learn how to build a proper robot application.

I am also particularly grateful to M. Florent NAGEOTTE for his assistance with the development of the stereo-vision system. I would also like to thank M. Baptiste GOMES for his help with the mechanical design of the system.

Finally, I would like to thank all my colleagues of the RDH team for their kindness, helpfulness, and for creating a welcoming and collaborative environment that made this internship a truly enriching experience.

Quito-Ecuador, 19 August 2024
Luis Maldonado

Contents

List of Figures

Glossary

ANR Agence Nationale de la Recherche.

CAD Computer Aided Design.

CT Computed Tomography.

DICOM Digital Imaging and Communications in Medicine.

DKM Differential Kinematics Model.

DLT Direct Linear Transforms.

DoF Degrees of Freedom.

DoFF Degrees of Force Feedback.

FKM Forward Kinematics Model.

FOV Field of Vliew.

GUI Graphical user interface.

HCI Human Computer Interface.

HSV Hue, Saturation, Value.

IKM Inverse Kinematics Model.

IMU Inertial Measurement Unit.

IR Interventional radiology.

PTFE Polytetrafluoroethylene.

RCM Remote Center of Motion.

ROS2 Robot Operating System 2.

URDF Unified Robot Description Format.

US Ultrasound.

Chapter 2: Introduction

2.1 Presentation of the ICUBE Lab :

The *ICUBE* laboratory, established in 2013 through a collaboration between the *CNRS*, the *Strasbourg University*, the *ENGEES*, and the *INSA Strasbourg*, uniquely integrates different scientific communities to bridge the digital and physical worlds. By 2024, the lab has grown to around 650 members, making it a significant research hub in Strasbourg. The *ICUBE* lab focuses primarily on health, the environment and sustainable development. The lab has a close relationship with the *Télécom Physique Strasbourg* engineering school and is a member of the *Institut Carnot Télécom & Société Numérique*. It also maintains numerous industrial partnerships with national and international companies and collaborates with over 270 scientific organizations all over the world.

Figure 1: *Principal ICUBE partners*

The lab is composed of 17 teams distributed in 4 departments according to the fundamental domains of the lab (computer science, imaging and robotics, electronics and photonics, mechanics). The *ICUBE* lab also possesses 7 leading technology platforms in medical imaging and robotics, remote sensing, Internet of Things, photovoltaic materials, bioinformatics, biomechanics and water treatment, that have their own organisation and dedicated personnel. These platforms support the research activities in the lab but also have the capacity to provide services to outside actors. A complete organizational chart of the lab can be found on annex **A**.

In my 4 months internship, I was assigned to the Imaging, Robotics, Teledection and Health (D-IRTS), in which I worked in the Robotics, Data science and Healtcare technologies (RDH) team. This interdisciplinary team, as the name implies, actively participates in research in the following fields :

- **Medical Robotics and Interventional Imaging :** encompasses the team's historical work in robotic assistance for minimally invasive medical and surgical procedures, as well as advancements in methodologies and clinical developments in Interventional Radiology.
- **Learning, Modelling and Data Science :** gathers the activities of the team around artificial intelligence, biomechanical simulation and measurement and evaluation methods, pursued both independently and in synergy, as simulation can be used to generate data for learning.
- **Complex systems and parsimony :** gathers activities around the control of complex systems, with an evolution over the period aiming at taking into account parsimony as an issue for the control but also for the mechatronics design of robots.

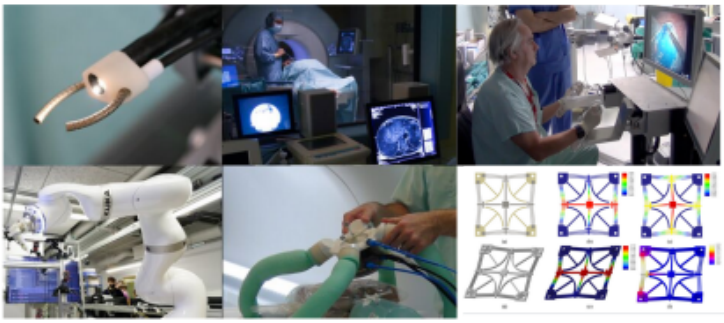

Figure 2: *Robotics projects of the RDH team at ICUBE*

The material aspect of the project I worked on was financed by the *ANR* (Agence Nationale de la Recherche), a public establishment placed under the supervision of the Ministry of Higher Education, Research and Innovation. Created in 2005, the *ANR* has the following missions :

- **Finance and promote** the development of fundamental and finalized research, technical innovation and technology transfer as well as partnership between the public and private sectors.
- **Implement the programming** decreed by the Minister responsible for Research who gathers the opinions of the ministers supervising research organizations or public higher education establishments.
- **Manage major State investment programs** in the field of higher education and research, and monitor their implementation.
- **Strengthen scientific cooperation** at the European and international levels, by articulating its programming with European and international initiatives.
- **Analyze the evolution of the research offer** and measure the impact of funding allocated by the Agency on national scientific production.

Figure 3: *ANR and CAMI logos*

However, my internship was financed by the *CAMI LABEX* (Computer Assisted Medical Interventions Excellence Lab) organisation which is co-financed by the *ANR* in the framework of the *Investissement d'Avenir* Program. The *CAMI LABEX* mission is to foster an integrated approach to medical interventions that will lead to a significant breakthrough in the quality and widespread adoption of *CAMI* technology in clinical practice.

2.2 Project presentation :

2.2.1 Context :

Interventional radiology (**IR**) is a relatively young speciality that is increasingly in demand as a result of the benefits that this **minimally invasive** approach offers compared to standard surgery. **[1]**. An **IR** procedure involves the insertion of a tool through a predefined insertion point in order to follow a trajectory to a target where a therapy is needed (ablation, cryotherapy, etc) or to collect biological samples. The practitioner must guide the needle inside the body, in a very restricted space, using only information acquired by an **imaging device** such as an **MRI** or **CT** scans. The imaging devices often produce **non real-time** images of that are difficult to interpret. As such, this technique requires a **high level of skill** and thus the specialty possesses relatively few practitioners. Nevertheless, the advantages for the patients remain very attractive : **faster recovery time** (which also means less hospital care and expenses), **less pain** and overall **less risk of complications**.

Given the difficult nature of **IR** procedures, **training** is of utmost importance. However, this could pose a **logistical**, **technical** and **ethical problem**. Indeed, in the classic apprenticeship model, an expert must accompany the student while practicing on real clinical cases. However, supervisors often do not have the time to properly teach the procedure or can be all together unavailable due to time constraints or lack of trained staff. As such, the **risk of harming the patient** due to a **lack of skill** is very real.

In order to compensate for this, new technologies like **virtual/augmented reality simulations** and the use of **phantoms** or **cadaveric simulations** have been proposed as means of supplementing training. However, these strategies have many drawbacks, for example the physical properties of the materials used in **phantoms are often inconsistent** or subjected to variability depending on the environment (temperature, humidity, etc). Cadaveric simulations are undoubtedly a more realistic anatomical portrayal but have also been criticized. In fact, the cadaver cannot emulate **blood circulation**, **respiratory cycle** or the **internal movement of the organs** and it has even been stated that post-mortem tissue is not a good substitute of living tissue, especially regarding the texture and mechanical properties. Furthermore, operating on dead bodies presents many **logistical and ethical concerns** as it is difficult and expensive to acquired and preserve the body. For these reasons, implementation of cadaveric simulation training is limited **[1]**.

Figure 4: *Haptic interaction between human and environment, I = intentions, P = perception, M = movement, S = sensing*

As such, **haptic devices** have become increasingly popular as a way of providing a realistic simulation. A haptic device is defined as a system that interacts with a human (called the *user*) via the means of **haptic perception and interaction**. The device makes use of different aspects of the **human sensorimotor system**. Indeed, haptic sensations can involve temperature, pressure, vibrations and kinaesthetic feedback (also known as *propioception*). This **haptic feedback** is generated by acquiring information in the real or virtual world and communicating with the user via a **human-computer interface** (HCI) in which a computer processes the information and generate commands transmitted to the user via various actuation systems (joysticks, robotic arms, etc). In medical applications, this feedback should **mimic**, as close as possible, the behaviour of the tissues such as elasticity, resistance to rupture, tendering due to stress and so on. Therefore, the **modeling** of these parameters has to be done with great care.

2.2.2 Problem definition :

The goal of this project is to develop a **needle training device** that will provide the user with a **haptic force feedback** when manipulating the needle. As such, the mechanism that will be used as a haptic interface has to follow several constraints specific to haptic devices and needle manipulation devices:

1. **Needle orientation :** The device needs to simulate the orientation the needle can have in IR procedures, in which it can rotate along two perpendicular axis (thus 2 rotational DoF are needed). In some extreme cases the needle can reach an angle of 60 degrees with respect to the normal to the skin of the patient, however due to hardware limitations, we can limit ourselves to a target maximum angle of 45 degrees.
2. **Needle insertion :** The position of the needle tip has to be measured along its insertion axis.
3. **Force feedback :** The device should actuate some of the DoFs to generate a force at the needle, felt by the user.
4. **Low apparent mass :** To accurately simulate force feedback, the general mass and inertia of the device should be minimized. Particularly, it has to have low eccentricity (i.e the distance between the masses). Typically motors are the heaviest part of a mechatronic device, as such they should be place as not to be moved during the device operation.
5. **Low friction :** The friction in the different joints of the device should be minimized by using appropriate bearings and lubrication.
6. **Backdrivability :** The end-effector of the device should be controlled and moved by the user without him feeling any constraints.
7. **Sensors :** The position and orientation of the needle needs to be measured in order to be displayed on the pre-operative images.

Additionally, in the final system, the user should only interact with the needle and the mechanism will be visually hidden under some sort of fake skin in order to simulate the patient and place the student in a realistic environment. As such, we also need to design a Graphical User Interface (GUI) that should mimic the information available to the practitioner, in the real IR procedure.

2.2.3 Description of the current prototype:

In order to follow the project specifications presented in section **2.2.2**, an initial research work was carried out in 2023 by Mr Thomas CESARE **[2]**, a previous master student at the *Telecom Physique Strasbourg* engineering school. The insights and conclusions from his research will be incorporated throughout this report.

Indeed, T.CESARE carried out a detailed study of the different geometric designs used in haptic or needle manipulation devices available in the scientific literature and proposed the solution that will be detailed in this section. T.CESARE design was inspired by the works of G CAMPION et al **[3]** and a system from the company *Quanser* **[4]**. The current prototype uses a planar pantograph linked to a shaft via 2 3D printed ball joints, one of which is fixed to the structure holding the robot, thus constraining the shaft to move around a fixed point that represents the insertion point located at the skin of the patient. The needle is then inserted into the shaft, which is filled with silicone to simulate the friction encountered when inserting the needle through the skin and muscles.

The pantograph is actuated by a *capstan drive system* as seen in figure **6**, this transmission system operates on the principle of frictional force to control the motion of a cable or belt wound around a rotating drum (or *capstan*). The system uses a tensioning system to secure the cable, enabling precise control of movement with minimal slippage. This mechanism is widely used in haptic devices as it allows for smooth and accurate force feedback, essential for creating realistic haptic sensations.

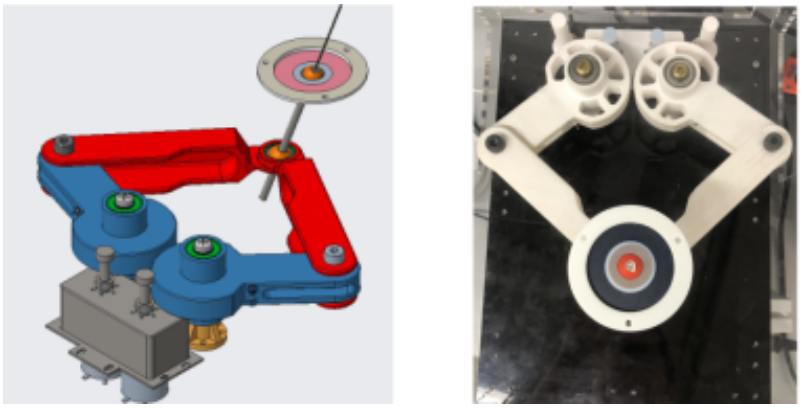

Figure 5: *Needle pantograph designed by T.CESARE,* ***[2]***

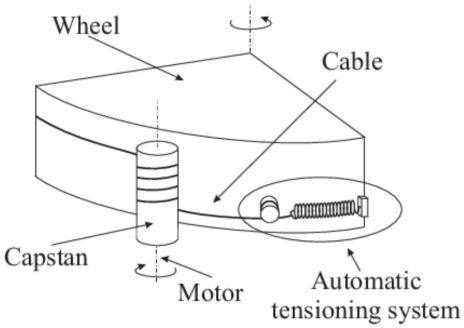

Figure 6: *Schematic of a capstan drive system, extracted from* ***[5]***

To reduce friction in the pivot joints, 2 ball bearings per joint are used, these are mounted with the outer ring locked into the housing internally, while the inner ring is externally pre-loaded. The bearings are separated by a distance H such that $H >> 2.D$ with D the diameter of the corresponding shaft, this compensates the shaft misalignment and ensures that the movement at the joint is purely a rotation.

The software aspect of the system is handled by a *ROS2* framework which will be explained in detail in section **5.1**. Finally, the interface between the commands sent by the PC and the motors is done using the *EtherCAT* technology, a high-performance industrial Ethernet-based fieldbus system designed to enable fast and efficient communication between various devices, using the drivers developed at the lab.

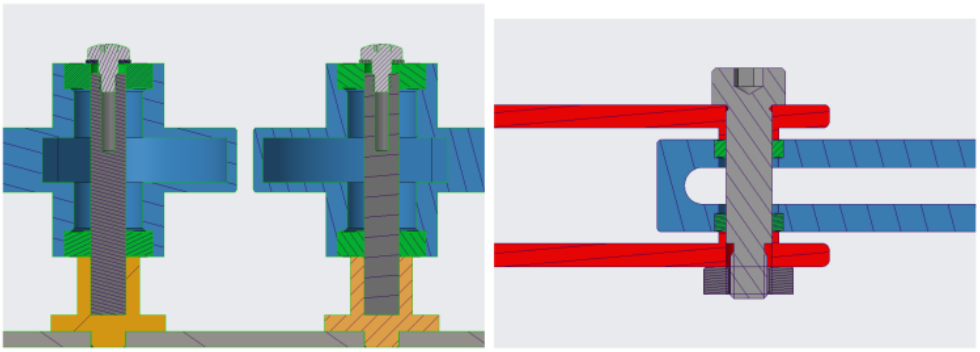

Figure 7: *Rotary joints design: grounded (left) and flying (right) joints. Ball bearings (green), Shafts (grey)*

Unfortunately, the device has some flaws which impact the accurate control of the robot. Indeed, the friction and heavy mass of the segments sometimes causes the robot to be stuck, especially at the level of the ball joints where the mechanical play and friction between the resin-resin contact is high. The motors must then send a large torque command to counteract the friction and this in turn causes the robot to oscillate around a target point. Additionally, this prototype does not have an actuator for the needle insertion and isn't capable of measuring the position of the needle in real time. This exclusive reliance on the model prevents the implementation of more complex control strategies.

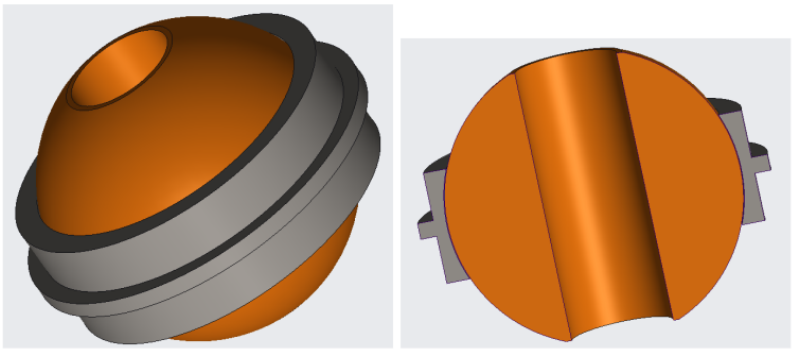

Figure 8: *3D printed ball joint*

Thus, the goal of my internship is to review the work previously done in **[2]** in order to address the issues previously mentioned and allow the device to be usable for a first set of experiments with medical students and practitioners.

Chapter 3: State of the art

3.1 Needle insertion devices :

In the context of needle manipulation, needle insertion and image guided procedures, the usage of robots has already been thoroughly studied in scientific literature. Some devices are even already commercially available such as the *Micromate* device by *Interventional Systems* **[6]**. However, many of these robotic systems are not devoted or adaptable to training purposes, nor they take into account the constraints needed to produce an accurate haptic feedback. Nevertheless, it is worth investigating these devices in order to understand their mechanical structure and the constraints they need to satisfy to ensure the needle manipulation and insertion guidance to eventually try to adapt one to our project.

Figure 9: *Micromate device by Interventional Systems*

The geometrical design of the different systems in the literature share a fundamental feature : a remote center of motion (RCM). A RCM is a point in space that appears to be fixed with respect to a moving object, even though it may be located some distance from the object itself. This is a very common feature to find in surgical tools manipulation devices because it renders the robot inherently safe, since it avoids potentially harmful translations along axis perpendicular to the insertion one, that would stir and damage the tissues around the entry point **[7]**.

As mentioned in section **2.2.2**, when trying to adapt a mechanism of such type to haptics, one has to consider several aspects: first of all the bulkiness should be limited and light weight devices have to be preferred. Also, motion transmission can be a problem for backdriveability, as such cable transmission is preferred rather than gears and belts. Actuators are generally low inertia coreless DC motors that minimize torque cogging and friction, while linear actuators are not so commonly used. Furthermore, the actuators should be placed in a position such that they do not move during system operation to minimize mass eccentricity.

3.2 Haptic devices in medicine :

The use of haptic simulators for medical training is not a recent development, nor is it exclusive to interventional radiology. Escobar et al. **[8]** have documented numerous examples of haptic simulators used in dental procedures, palpation, stitching, endoscopy, laparoscopy, and orthopedics. Correa et al.**[9]** reviewed 145 papers specifically focused on haptic training for needle insertion procedures. In the majority of these applications, the authors used a commercially available haptic device. However, despite these devices being relatively "plug and play" and commercially available, it is worth noting that their price range varies drastically, going from 200-2000 € , for the more affordable ones while the expensive variety (generally the ones with more actuated DoFs) can reach 40-50K €, as we can see in figure **9**. Additionally, often the end-effector of these devices is not adapted to a particular task and as such it is necessary to build a custom interface as seen in figure **11**.

Indeed, in this study by Young et al **[10]** a *Novint Falcon* device was augmented with a 3D printed interface composed of a gripper, an epidural needle and an IMU (Inertial Measurement Unit) controlled by an arduino. This allowed the team to retrieve the roll, pitch and yaw angles of the needle, thus, effectively transforming the initial 3 DoF device into a 6 DoF one.

Device	Number of studies	DoF	DoFF	Euros ×1000
Phantom Omni	48	6	3	2
Phantom Premium	26	6	3 or 6	18 to 70
Novint Falcon	8	3	3	0.2
Phantom Desktop	8	6	3	11
Delta	3	3 or 6	3 or 6	22 to 40
Cyberforce	3	6	3	45
Virtuose 6D Desktop	3	6	6	30
Quanser 3DOF	2	3	3	25
Omega 6	1	6	3	14 to 24

Figure 10: *Main commercial haptic devices ranked by the number of studies in which they are used.* ***[8]***

Figure 11: *Novint Falcon haptic device with custom-made interface to connect an epidural needle.* ***[10]***

In 2011 Ni et al. **[11]** developed a simulator for ultrasound guided biopsy training, composed of 2 haptic devices controlled independently: one *Phantom Omni* to simulate the ultrasound probe and a *Phantom Premium* for manipulating the needle. They enhanced the simulation by combining CT volumes and US images where they also were able to simulate breathing induced motion of the inner organs. The force feedback was computed as a 6 DoF feedback, fully exploiting the Phantom Premium capabilities. Both novices and experienced radiologists tested the simulator and reported a positive feedback regarding the visual and the force haptic rendering.

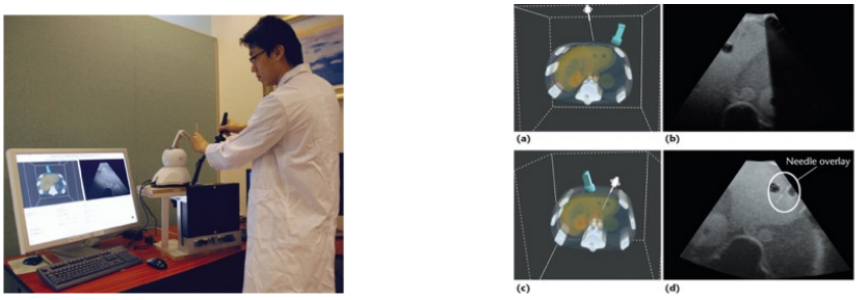

Figure 12: *Ni et al haptic system (left) and overlaid projection of the needle in the US image (right).* ***[11]***

Chapter 4: Modelling

4.1 Generation of the haptic feedback

As mention in the introduction of this report, the goal of the device presented in section **2.2.3** is to provide a haptic feedback to the user by manipulating the shaft in which the needle slides. As such, the general forces involved in this problem can be represented as shown in figure **13**.

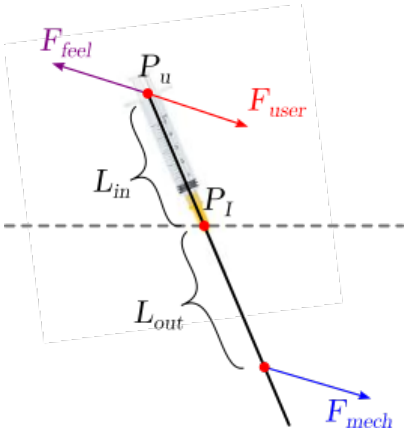

Figure 13: *Diagram of the different forces actuating on the needle*

The dashed line represents the "skin" under which the mechanism is located. The needle is constrained at the RCM point P_I which represents the insertion point on the skin of the patient and can be modelled as a sliding pivot (2 rotations and 1 translation). The user will manipulate the needle at point P_u, by applying the force F_{user}, the device has to in turn, apply a force F_{mech} at point P_{mech} in order to generate a force F_{feel} that counters the force applied by the user. The magnitude of the haptic force F_{feel} can be calculated, taking into account the lever effect, as follows :

$$F_{feel} = -F_{mech} \cdot \frac{L_{in}}{L_{out}} \tag{4.1}$$

In order to prevent the force F_{feel} to diverge when $L_{in} > L_{out}$, a portion of the needle can always be pre-inserted by adding a mechanical stop, thus fixing the minimum lengths such that the condition $L_{in} < L_{out}$ is always true.

Additionally, the maximum value of the force F_{feel} can be set to any desired amount (provided that the motors and structure support the load), for safety concerns, we kept this maximum at 5 N.

4.2 Kinematics analysis :

The goal of this section is to review the mathematical models established in T.CESARE report **[2]** in order to verify their correct implementation.
The first step in every robotics project is to determine the relation between the position of the end-effector of the robot X and the position of its actuated joints Q. The relation $X = f(Q)$ is known as the Forward Kinematic model (**FKM**) and its inverse $Q = f^{-1}(X)$ corresponds to the Inverse Kinematics model (**IKM**). In the case of a parallel structure such as the pantograph, finding the models can be a tricky task and the problem is often solved using geometric methods.

4.2.1 Forward kinematics of the pantograph

According to T.CESARE report **[2]**, the FKM of the pantograph can be calculated using the *Assur groups method*, a method very useful to analyse planar structures. By definition an *Assur* group is a kinematic chain without any internal mobility that, once added to the mechanism, does not change its degrees of freedom (DoF). The simplest *Assur* groups are known as *dyads* and are composed of 2 segments and 3 joints. In the case of the pantograph we can see that the joints P_2 P_3 and $P_{2'}$ form an RRR dyad, as such the position of P_3 can be determined by knowing the positions of P_2 and $P_{2'}$ and the lengths of the segments a_1 and a_2. The different model parameters are represented in the following figure, where the angles are oriented according to the trigonometric orientation :

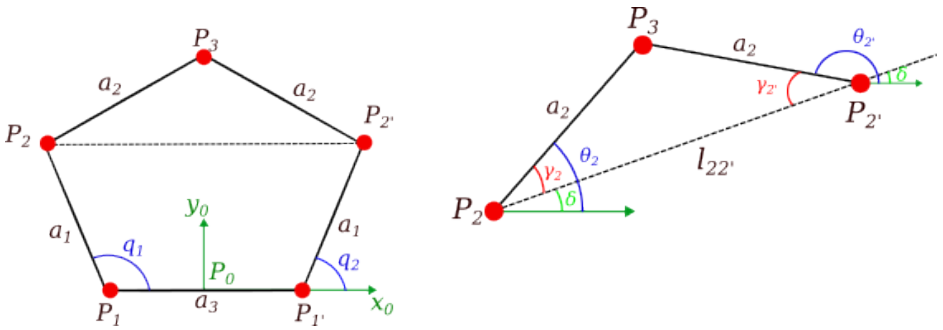

Figure 14: *Forward kinematics analysis*

First of all it is immediate that :

$$P_2 = \begin{bmatrix} a_1.\cos(q_1) - \frac{a_3}{2} \\ a_1.\sin(q_1) \end{bmatrix} \quad \text{and} \quad P_{2'} = \begin{bmatrix} a_1.\cos(q_2) + \frac{a_3}{2} \\ a_1.\sin(q_2) \end{bmatrix}$$

Let $l_{22'} = \|P_2 - P_{2'}\|$, according to Al-Kashi theorem, the angle γ_2 of the triangle $P_2P_3P_2'$ is :

$$\gamma_2 = \pm \operatorname{acos}\left(\frac{{l_{22'}}^2 + {a_2}^2 - {a_2}^2}{2.a_2.l_{22'}}\right) = \pm \operatorname{acos}\left(\frac{l_{22'}}{2.a_2}\right)$$

Here we chose only the first positive solution for γ_2, because the negative one corresponds to the symmetric (along the x_0 axis) assembly configuration of the pantograph.
The angle δ can be calculated as follows :

$$\delta = \operatorname{atan2}\left(\frac{y_{2'} - y_2}{x_{2'} - x_2}\right)$$

Knowing that $\theta_2 = \gamma_2 + \delta$, we can write :

$$P_3 = P_2 + a_2 \begin{bmatrix} \cos(\theta_2) \\ \sin(\theta_2) \end{bmatrix} \tag{4.2}$$

In the same way we can obtain the position of $P_{2'}$ from the other kinematic chain $(P_{1'}P_{2'}P_3)$, and using the angle $\theta_{2'} = \pi - \gamma_{2'} + \delta$. This last expression is useful in order to implement the FKM in *Matlab* and later in the *ROS2* environment.

4.2.2 Inverse kinematics of the pantograph :

First of all, in order to take into account the *ROS2* implementation we have to model the mechanism according to the Denavit-Hartenberg convention as presented in **[12]**. This method sets some rules in order to attach the frame of the solid *i* of the mechanism, and determined its position relative to the frame of solid *i-1* .

We chose to represent the robot as 2 independent serial kinematic chains linked with a constraint equation. This choice is useful to determine the Differential kinematics and Dynamic model of the robot and it will be explain in more detail in chapter **4.3**. As such, as we can see in the figure below, we have two chains : $P_1P_2P_3$ and $P_{1'}P_{2'}P_3$

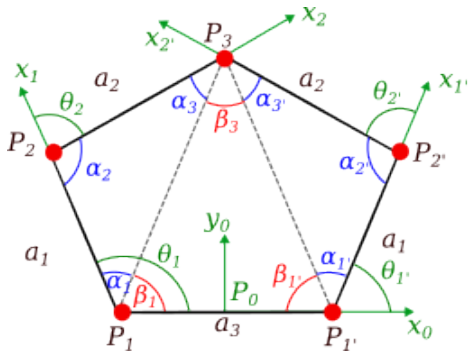

Figure 15: *Inverse kinematics analysis*

We assume that the position of point P_3 is known and we need to determine the angles θ_1 and $\theta_{1'}$ which corresponds to the actuated joints $(Q = [q_1, q_2]^T = [\theta_1, \theta_{1'}]^T)$. We also need to calculate the other θ_i angles, as they will be needed for the 3D simulation of the robot presented in section **5.1.4**.

Consider the triangle $P_1P_2P_3$, and let $l_{13} = \|P_3 - P_1\|$, once again using the Al-Kashi theorem we have:

$$\alpha_1 = \operatorname{acos}\left(\frac{a_1^2 + l_{13}^2 - a_2^2}{2.a_1.l_{13}}\right) = \alpha_1(X), \quad \alpha_2 = \operatorname{acos}\left(\frac{a_1^2 + a_2^2 - l_{13}^2}{2.a_1.a_2}\right) \quad \text{and} \quad \alpha_3 = \pi - \alpha_1 - \alpha_2$$

In the same way we can write for the triangle $P_{1'}P_{2'}P_3$:

$$\alpha_{1'} = \operatorname{acos}\left(\frac{a_1^2 + l_{1'3}^2 - a_2^2}{2.a_1.l_{1'3}}\right) = \alpha_{1'}(X), \quad \alpha_{2'} = \operatorname{acos}\left(\frac{a_1^2 + a_2^2 - l_{1'3}^2}{2.a_1.a_2}\right) \quad \text{and} \quad \alpha_{3'} = \pi - \alpha_{1'} - \alpha_{2'}$$

And if we consider the triangle $P_1P_3P_{1'}$ we have :

$$\beta_1 = \operatorname{atan}2\left(\frac{y_3}{x_1 - x_3}\right) = \beta_1(X), \quad \beta_{1'} = \operatorname{atan}2\left(\frac{y_3}{x_{1'} - x_3}\right) = \beta_{1'}(X) \quad \text{and} \quad \beta_3 = \pi - \beta_1 - \beta_{1'}$$

Finally, we can calculate all the θ_i angles :

$$\begin{matrix} \theta_1 = q_1 = \alpha_1 + \beta_1 \\ \theta_2 = \pi - \alpha_2 \end{matrix} \quad , \quad \begin{matrix} \theta_{1'} = q_2 = \pi - \beta_{1'} - \alpha_{1'} \\ \theta_{2'} = \pi - \alpha_{2'} \end{matrix}$$

As such, the different angles depend on the position of the end-effector $X = [x_3, y_3]^T$ and we have a relation of the type $Q = f^{-1}(X)$ that links the joint variables q_1 and q_2 to the end-effector position X:

$$Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \alpha_1(X) + \beta_1(X) \\ \pi - \alpha_{1'}(X) + \beta_{1'}(X) \end{bmatrix} \tag{4.3}$$

However, given the involvement of different trigonometric functions as well as the *norm* function the previous relation is highly non-linear.

4.2.3 Forward kinematics of the complete system :

In the previous subsection, we determined the geometric models of the pantograph, as such in this part we will consider the pantograph as a "black box" in order to simplify the different equations. We now want to add to the pantograph model, the shaft and the needle. The rotation of the needle around the insertion axis is not taken into account and as such this link can be modelled as a prismatic joint. As a first approach, we consider that the needle does not slide inside the tube and is instead fixed a predefined insertion depth. As such, we want to determine the relation between the point P_u, corresponding to the point where the user holds the needle, and the actuated joints $Q = [q_1, q_2]^T$, our approach can be illustrated by the following figure.

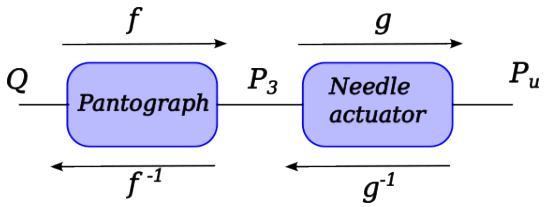

Figure 16: *Bloc diagram of the complete system*

Where f,g and f^{-1},g^{-1} represent respectively the FKM and IKM of the 2 parts of the system.
In order to calculate the FKM of the complete system we define the parameters presented in the following figure :

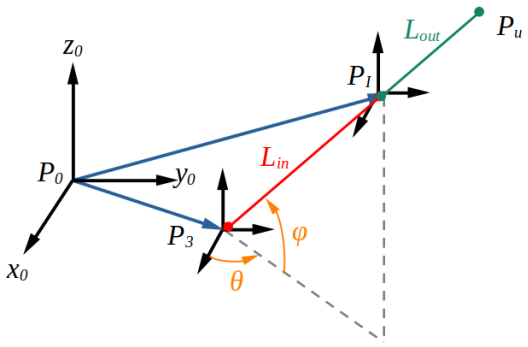

Figure 17: *FKM of the complete system*

We have :

- $\mathbf{P_I}$: The insertion point of the needle, its coordinates are fixed by the mechanical design of the robot.
- $\mathbf{P_u}$: The point where the user holds the needle.
- θ : The azimuth angle of the vector associated to the needle.
- φ : The elevation angle of the vector associated to the needle.
- $\mathbf{L_{in}}$: The portion of the shaft comprise between the insertion point, P_I and the pantograph end-effector, P_3.
- $\mathbf{L_{out}}$: The portion of the needle comprise between the insertion point, P_I and the point at which the user holds the needle, P_u

In figure **17** , we assume that the frames F_I and F_3 have their axis parallel to the ones of F_0 and that the translations $\overrightarrow{P_0P_I}$ and $\overrightarrow{P_0P_3}$ are known as we explain before. We also considered as a first approach, that the we have $L_{in} + L_{out} = C$ with C a fixed constant, later the length L_{out} will be measured in real time by a sensor. Therefore, our goal is to express the relation between the vector $\overrightarrow{P_0P_u}$ and the joint variables Q.

First of all, we can decompose the vector $\overrightarrow{P_0P_u}$ as such:

$$\overrightarrow{P_0P_u} = \overrightarrow{P_0P_3} + \overrightarrow{P_3P_I} + \overrightarrow{P_IP_u}$$

Where :

$$\overrightarrow{P_3P_I} = \overrightarrow{P_0P_I} - \overrightarrow{P_0P_3} = \begin{bmatrix} x_I - x_3 \\ y_I - y_3 \\ z_I - z_3 \end{bmatrix}_{F_0}$$

To simplify the notations we write :

$$\overrightarrow{P_3P_I} = \begin{bmatrix} x_{I3} \\ y_{I3} \\ z_{I3} \end{bmatrix}_{F_0}$$

Knowing that $\overrightarrow{P_3P_I}$ and $\overrightarrow{P_IP_u}$ are co-linear, we can determine the position of point P_u by extending the vector $\overrightarrow{P_3P_I}$ along a direction given by the angles θ and φ as seen in figure **17**. These angles can be calculated by expressing the vector $\overrightarrow{P_3P_I}$ coordinates in the spherical coordinates system in F_3 frame :

$$\overrightarrow{P_3P_I} = \begin{bmatrix} x_{I3} \\ y_{I3} \\ z_{I3} \end{bmatrix}_{F_0} = \begin{bmatrix} \theta \\ \varphi \\ L_{in} \end{bmatrix}_{e_\theta, e_\varphi, e_z}$$

Where:

$$\theta = \operatorname{atan2}\left(\frac{y_{I3}}{x_{I3}}\right) \quad , \quad \varphi = \operatorname{atan2}\left(\frac{z_{I3}}{\sqrt{x_{I3}^2 + y_{I3}^2}}\right) \quad \text{and} \quad L_{in} = \sqrt{x_{I3}^2 + y_{I3}^2 + z_{I3}^2}$$

As such the coordinates of the vector $\overrightarrow{P_IP_u}$ in frame F_I are :

$$\overrightarrow{P_I P_u} = L_{out}.\begin{bmatrix}\cos(\varphi).\cos(\theta)\\ \cos(\varphi).\sin(\theta)\\ \sin(\varphi)\end{bmatrix}_{F_I} \quad \text{With :} \quad L_{out} = C - L_{in}.$$

Finally, we can calculate the vector $\overrightarrow{P_0 P_u}$:

$$\overrightarrow{P_0 P_u} = \begin{bmatrix}x_I + L_{out}.\cos(\varphi).\cos(\theta)\\ y_I + L_{out}.\cos(\varphi).\sin(\theta)\\ z_I + L_{out}.\sin(\varphi)\end{bmatrix}_{F_0} \tag{4.4}$$

Given that the angles θ and φ depend on the position of the point P_3 we have found a relation between the end-effector position P_u and the joint variables Q of the type : $P_u = \mathrm{g}(P_3) = \mathrm{g}(\mathrm{f}(Q))$

4.2.4 Inverse kinematics of the complete system :

Similar to the previous analysis, we assume that the IKM of the pantograph is known and is written as follows : $Q = f^{-1}(P_3)$. We also assume that the coordinates of point P_u in F_0 are known. As such we need to find the relation linking the position of the points P_3 and P_u.

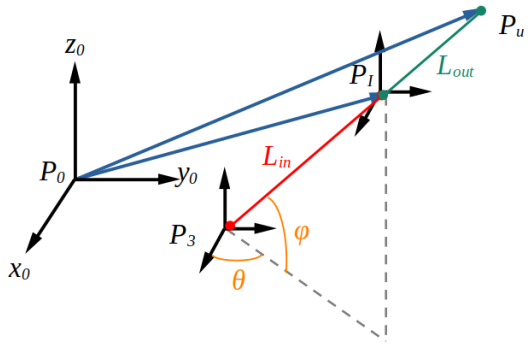

Figure 18: *IKM of the complete system*

First, we decompose the vector $\overrightarrow{P_0 P_3}$:

$$\overrightarrow{P_0 P_3} = \overrightarrow{P_0 P_I} + \overrightarrow{P_I P_3}$$

Once again, knowing that the vectors $\overrightarrow{P_I P_3}$ and $\overrightarrow{P_I P_u}$ are co-linear, we need to calculate the angles θ and φ of the previous figure to determine the orientation of vector $\overrightarrow{P_I P_3}$ in F_I frame. As such, we express the vector $\overrightarrow{P_I P_u}$ in spherical coordinates in frame F_I :

$$\overrightarrow{P_I P_u} = \overrightarrow{P_0 P_u} - \overrightarrow{P_0 P_I} = \begin{bmatrix}x_u - x_I\\ y_u - y_I\\ z_u - z_I\end{bmatrix}_{F_0} = \begin{bmatrix}x_{uI}\\ y_{uI}\\ z_{uI}\end{bmatrix}_{F_0} = \begin{bmatrix}\theta\\ \varphi\\ L_{out}\end{bmatrix}_{e_\theta, e_\varphi, e_z}$$

Where :

$$\theta = \mathrm{atan2}\left(\frac{y_{uI}}{x_{uI}}\right) \quad , \quad \varphi = \mathrm{atan2}\left(\frac{z_{uI}}{\sqrt{x_{uI}^2 + y_{uI}^2}}\right) \quad , \quad L_{out} = \sqrt{x_{uI}^2 + y_{uI}^2 + z_{uI}^2}$$

In order to calculate the vector $\overrightarrow{P_I P_3}$ in frame F_I we can draw the following figure:

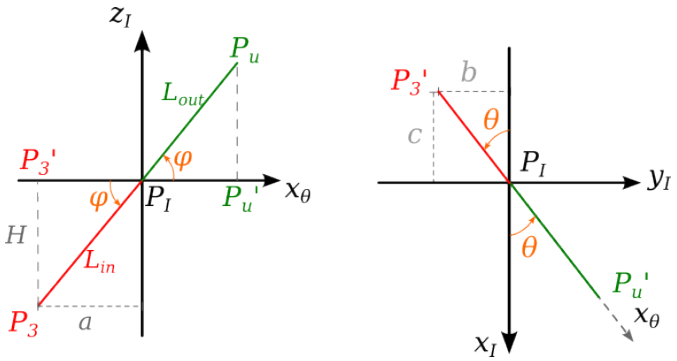

Figure 19: *Calculation of* $\overrightarrow{P_I P_3}$

In this figure we have:

$$\overrightarrow{P_I P_3}_{/F_I} = \begin{bmatrix} -c \\ -b \\ -H \end{bmatrix}_{F_I} = \begin{bmatrix} -a.\cos(\theta) \\ -a.\sin(\theta) \\ -L_{in}.\sin(\varphi) \end{bmatrix}_{/F_I}$$

With $a = L_{in}.\cos(\varphi)$ and $L_{in} = C - L_{out}$, as such we can write $\overrightarrow{P_I P_3}$ in F_I :

$$\overrightarrow{P_I P_3} = -L_{in}.\begin{bmatrix} \cos(\varphi).\cos(\theta) \\ \cos(\varphi).\sin(\theta) \\ \sin(\varphi) \end{bmatrix}_{F_I}$$

Finally, we can calculate the vector $\overrightarrow{P_0 P_3}$ in F_0 :

$$\overrightarrow{P_0 P_3} = \begin{bmatrix} x_I - L_{in}.\cos(\varphi).\cos(\theta) \\ y_I - L_{in}.\cos(\varphi).\cos(\theta) \\ z_I - L_{in}.\sin(\varphi) \end{bmatrix}_{F_0}$$

Once again we can verify that we have a relation of the type $Q = g^{-1}(P_3) = g^{-1}(f^{-1}(P_u))$ because the angles θ and φ depend on the position of the point P_u.

Note : When implementing the previous models in a ROS node, the following modifications are needed:

We replace the angle φ by $\varphi' = \frac{\pi}{2} - \varphi$, this has to be done because in RViz, the angles are expressed by reference to the parent frame, as such we have :

$$\overrightarrow{P_I P_3} = L_{in}.\begin{bmatrix} \cos(\varphi').\cos(\theta) \\ \cos(\varphi').\cos(\theta) \\ -\sin(\varphi') \end{bmatrix}_{F_0} = L_{in}.\begin{bmatrix} \sin(\varphi).\cos(\theta) \\ \sin(\varphi).\cos(\theta) \\ -\cos(\varphi) \end{bmatrix}_{F_0} \tag{4.5}$$

4.3 Differential kinematics analysis :

The goal of this section is to determine the differential kinematic model of the pantograph that will be needed by the controller logic. We can proceed with the simple derivation method which consists of directly deriving the FKM equations in order to find a relation between the end-effector and actuated joint velocities of type : $\dot{X} = \mathrm{DKM}(Q, \dot{Q})$ with $Q = [\theta_1, \theta_{1'}]^T$. According to T.CESARE report **[2]**, we can consider the kinematic chain $P_1P_2P_3$, where the position of the end-effector P_3 is :

$$P_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} a_1.c_1 - \frac{a_3}{2} + a_2.c_2 \\ a_1.s_1 + a_2.s_2 \end{bmatrix}$$

With $s_i = \sin(\theta_i)$ and $c_i = \cos(\theta_i)$ to simplify the notations.
By directly deriving this equation we have :

$$\dot{P_3} = \begin{bmatrix} \dot{x_3} \\ \dot{y_3} \end{bmatrix} = \begin{bmatrix} -a_1.\dot{\theta_1}.s_1 - a_2.\dot{\theta_2}.s_2 \\ a_1.\dot{\theta_1}.c_1 - a_2.\dot{\theta_2}.c_2 \end{bmatrix}$$

We now need to find the relation between $\dot{\theta_2}$ and $\dot{Q}$, for this we consider the dyad $P_2P_3P_{2'}$ where:

$$P_{2'} = \begin{bmatrix} x_{2'} \\ y_{2'} \end{bmatrix} = \begin{bmatrix} -\frac{a_3}{2} + x_2 + a_2.c_2 - a_2.c_{2'} \\ y_2 + a_2.s_2 - a_2.s_{2'} \end{bmatrix}$$

Deriving this equations yield :

$$\dot{P_{2'}} = \begin{bmatrix} \dot{x_{2'}} \\ \dot{y_{2'}} \end{bmatrix} = \begin{bmatrix} \dot{x_2} - a_2.\dot{\theta_2}.s_2 + a_2.\dot{\theta_{2'}}.s_{2'} \\ y_2 + a_2.\dot{\theta_2}.c_2 - a_2.\dot{\theta_{2'}}.c_{2'} \end{bmatrix}$$

We can write this in matrix form like such :

$$\begin{bmatrix} -a_2.s_2 & a_2.s_{2'} \\ a_2.c_2 & -a_2.c_{2'} \end{bmatrix} . \begin{bmatrix} \dot{\theta_2} \\ \dot{\theta_{2'}} \end{bmatrix} = \begin{bmatrix} \dot{x_{2'}} - \dot{x_2} \\ \dot{y_{2'}} - \dot{y_2} \end{bmatrix}$$

Solving for $\dot{\theta_2}$ and $\dot{\theta_{2'}}$ yields :

$$\begin{bmatrix} \dot{\theta_2} \\ \dot{\theta_{2'}} \end{bmatrix} = \frac{1}{{a_2}^2.s_{2-2'}} \begin{bmatrix} a_2.s_2 & a_2.s_{2'} \\ a_2.c_2 & a_2.c_{2'} \end{bmatrix} . \begin{bmatrix} \dot{x_{2'}} - \dot{x_2} \\ \dot{y_{2'}} - \dot{y_2} \end{bmatrix}$$

The angles θ_2 and $\theta_{2'}$ can be computed according to the pantograph FKM as seen in section **4.2.1** and the velocities of points P_2 and $P_{2'}$ and can be obtained by directly deriving the pantograph FKM. Finally, we can write the pantograph DKM as follows :

$$\dot{P_3} = J_{panto}.\dot{Q} \tag{4.6}$$

With J_{panto} the analytical Jacobian of the pantograph :

$$J_{panto} = \begin{bmatrix} -a_1.\left(s_1 + s_2.\frac{s_{1-2'}}{s_{2'-2}}\right) & a_2.s_2.\frac{s_{2-2'}}{s_{2'-2}} \\ a_1.\left(c_1 + s_2.\frac{s_{1-2'}}{s_{2'-2}}\right) & -a_2.c_2.\frac{s_{2-2'}}{s_{2'-2}} \end{bmatrix} \tag{4.7}$$

4.4 Control strategy

Once the mechanical design of the robot and hardware/software interface have been completed, the next step is to implement a control architecture to satisfy the user requirements. The control software will operate inside the user PC, taking advantage of the *ROS2 _control* framework. The main advantage of the *ROS2* environment, is to enable the concurrent management of sensor data acquisition, motor control commands, display of the user interface and the execution of the master software responsible for the *EtherCAT* system. The characteristics of the ROS2 environment will be presented in detail in section **5.1**.

4.4.1 Statics analysis :

First of all, it is necessary to determine the relation between the generalized forces applied to the end-effector and the generalized forces applied to the joints at an equilibrium configuration. This approach can be sufficient to model and accurately control the robot, since the needle moves slowly and has low mass, therefore the inertia of the end-effector is low and it isn't necessary to carry out a detailed dynamic analysis.

According to the *virtual work principle* explained in **[12]**, the mechanical system is considered to be time-invariant and posses holonomic constraints, thus its configurations depends only on the joint variables Q and not explicitly on time. This implies that virtual displacements coincide with elementary displacements. The notable result of this principle is the following :

$$\tau = J^T(Q).\gamma \tag{4.8}$$

Where τ represent the vector of joint torques and γ the vector of the forces applied to the end-effector of the pantograph. In this study, the pantograph only has 2 actuated joints and it can only apply forces on the (x,y) plane. As such, both τ and γ are of dimensions 2x1. The equation **4.8**, shows that the relationship between the end-effector forces and the joint torques is given by the transpose of the manipulator geometric Jacobian.

The Jacobian of the pantograph can be calculated as seen in equation **4.7**, our goal is therefore, to find the the force that needs to be generated at the pantograph end-effector when the user manipulates the needle at point P_u in order to guide it to a predefined trajectory.

4.4.2 Virtual fixtures presentation

As explained in the introduction of this report, the goal of the system is to provide a haptic feedback to the user in order to help in the training of needle insertion procedures. The following article titled, *Active Constraints/Virtual Fixtures: A Survey* by Stuart A. Bowyer et al. **[13]**, presents a variety of techniques that can be used to assist the user in human-machine collaboration tasks.

The terms "**active constraint**" and "**virtual fixture**" refer to the concept of collaborative control strategies in which motion is regulated relative to predefined trajectories and known restricted regions. The controller then attenuates or nullifies the user command if it causes the manipulator to digress from the trajectory or enter a restricted area.

The Generalized active constraint implementation framework is the following :

1. **Constraint definition :** Define the geometry/trajectory to follow/avoid.
2. **Constraint evaluation :** Evaluate the relative configuration between the robot and the constraint.
3. **Constraint enforcement :** Decide how motion should be applied based on the constraint evaluation method.

As such, we begin by the constraint definition. In our study, the trainee has to follow a predefined trajectory which in the case of needle insertion procedures, consist of a straight line going from an insertion point located at the skin of the patient, to a target point located inside the patients body (often corresponding to a tumor or an organ). It is possible to create an "attractive" or "repulsive" constraint, as illustrated in the figure **20** extracted from **[13]**, the constraint acts at the dashed line with the arrows showing the direction of the "encouraged" motion.

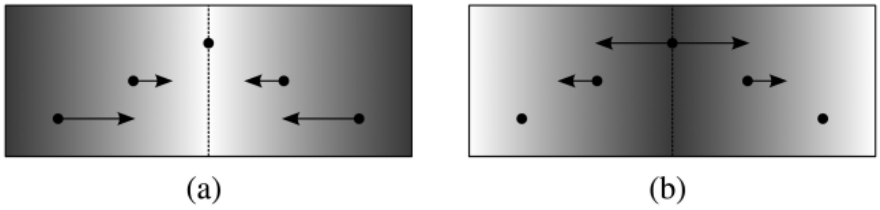

Figure 20: *Example of attractive (a) and repulsive (b) constraints*

The choice between this 2 types of constraints enables the creation of a variable "difficulty level" when the trainee is trying to follow the predefined trajectory. At an easier difficulty, the trainee should feel a force that tries to bring the needle back to the trajectory if he deviates from the path. A more challenging exercise would involve the mechanism applying a repulsive force to the needle whenever a slight deviation from the predefined trajectory is detected, thus forcing the trainee to perfectly follow the path.

4.4.3 Direct force control :

As a first approach, our goal is to implement a direct force controller in order to apply a force at the point where the user holds the needle, that brings the needle back to a predefined trajectory if the user deviates from it. This "spring-dampener" force will be generated by the pantograph end-effector taking into account the lever effect, as explained in the following section.

First it is necessary to define the geometry constraint of the problem, in our case we assume that user just needs to follow a fixed straight line trajectory, that is defined in advance. Then, we need to measure the error between the current needle orientation and the planned trajectory, this can be done as illustrated in figure **21**

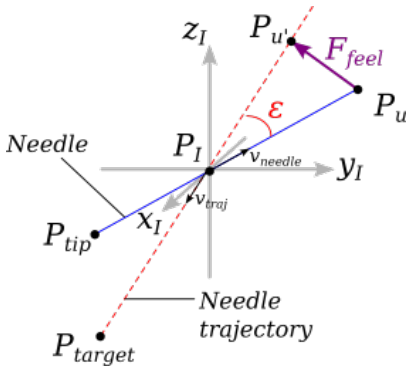

Figure 21: *Constraint and error definition*

We assume that the position of the target point P_{target} is known, as it was defined in advance by the user. The position of point P_u can be determined using the FKM of the complete system as detailed in section **4.2.3**.

Let $\hat{v}_{needle}$ and $\hat{v}_{traj}$ be the unitary vectors associated to the needle and the needle trajectory respectively. The angle ε can be calculated as follows :

$$\varepsilon = \pi - \text{acos}\left(\frac{\hat{v}_{needle}.\hat{v}_{traj}}{\|\hat{v}_{needle}\|.\|\hat{v}_{traj}\|}\right) \tag{4.9}$$

Let $\overrightarrow{F_{feel}} = F_{feel}.\hat{v}_u$ be the force felt by the user at the point P_u, the unit vector $\hat{v}_u$ giving the direction of this force is orthogonal to the trajectory and can be calculated as follows :

$$\hat{v}_u = \frac{\overrightarrow{P_I P_{u'}} - \overrightarrow{P_I P_u}}{\|\overrightarrow{P_I P_{u'}} - \overrightarrow{P_I P_u}\|}$$

with:

$$\overrightarrow{P_I P_{u'}} = -P_I P_u.\cos(\varepsilon).\hat{v}_{traj}$$

The force magnitude F_{feel} can be adjusted in several ways, as a first approach we consider a proportional gain based on the angle error ε, the more the needle deviates from the trajectory, the stronger the force $\overrightarrow{F_{feel}}$ will be :

$$\overrightarrow{F_{feel}} = K_p.\varepsilon.\hat{v}_u \tag{4.10}$$

However, because of the mechanical limitations of the pantograph and to ensure the safety of the user, the force must have an upper limit. As such, we choose K_p such that $F_{feel} \in [0, 5]$ N.

To determine the orientation of the force to be applied by the pantograph end-effector, let Π be the plane formed by the vectors $\hat{v}_{traj}$ and $\hat{v}_{needle}$. According to figure **23**, it is clear that all of the forces vector belong to the plane Π. Furthermore, the vector of the force $\overrightarrow{F_{mech}}$ corresponds to the intersection between Π and the $(\vec{x_0}, \vec{y_0})$ plane of the pantograph.

Let $\vec{r}_{traj} = \vec{r}_0 + t.\hat{d}$ be the equation representing the trajectory line where $\vec{r}_0$ corresponds to the vector of the position of the insertion point P_I (Which is known and fixed), $\hat{d}$ is a unit vector representing the direction of the trajectory and t is a parameter that allows us to place a point somewhere along the line. The coordinates of the target point P_{target} are defined by the user, as such $\hat{d}$ can be calculated as follows :

$$\hat{d} = \frac{\overrightarrow{P_I P_{target}}}{\|\overrightarrow{P_I P_{target}}\|}$$

Then, in order to calculate the direction of the force $\overrightarrow{F_{mech}}$ we need to find the intersection point between the trajectory line and the $(\vec{x_0}, \vec{y_0})$ plane. This point corresponds to the point where the dot product between the normal to the pantograph plane, $\vec{z}_0$ and the line equation $\vec{r_{traj}}$ is null :

$$\vec{z}_0.\vec{r}_{traj} = \vec{z}_0.(\vec{r}_0 + t_{int}.\hat{d}) = 0 \Leftrightarrow t_{int} = \frac{-\vec{z}_0.\vec{r}_0}{\vec{z}_0.\hat{d}}$$

This equation yields the parameter t_{int} that we then replace in the equation of the trajectory $\vec{r}_{traj}$, to find the coordinates of the intersection point P_{int}.

The direction unit vector of the force $\overrightarrow{F_{mech}}$ is as such :

$$\hat{v}_{mech} = \frac{\overrightarrow{P_3P_{int}}}{\|\overrightarrow{P_3P_{int}}\|} = [v_{mech_1}, v_{mech_2}, 0]^T$$

Then, we define the angle α as the angle between the force vector $\overrightarrow{F_{mech}}$ and the x axis of the pantograph end-effector frame, such that :

$$\alpha = \text{atan2}(v_{mech_2}, v_{mech_1})$$

This angle will be useful for the calculation of the magnitude of $\overrightarrow{F_{mech}}$ as we will see in the following section. Additionally, in order to prevent the pantograph from oscillating around the equilibrium point (when the value of the error is low) we need to add a damping term, K_d that depends on the end-effector velocity to $\overrightarrow{F_{mech}}$, as such we have :

$$\overrightarrow{F_{mech}} = F_{mech}.\hat{v}_{mech} + K_d.\dot{P}_3 \tag{4.11}$$

It is therefore necessary to calibrate the system in order to choose the values of the K_p and K_d gains.

4.4.4 Modelling of the force exerted by the mechanism

As explained in the previous subsection, the system must produce a force on the needle, felt by the user at point P_u, by applying a force at the base of the needle with the pantograph end-effector. The equations defining this problem were determined in **[2]**, as such in this section I will just rewrite the equations using a more explicit notation. The forces involved are the following :

- $\mathbf{F_{guide}}$: The magnitude of the desired guiding force to be applied to the user
- $\mathbf{F_{feel}}$: The force actually felt by the user
- $\mathbf{F_{mech}}$: The force generated by the mechanism to produce F_{feel} :
- $\mathbf{F_{perp}}$: The perpendicular component of the force F_{mech} :

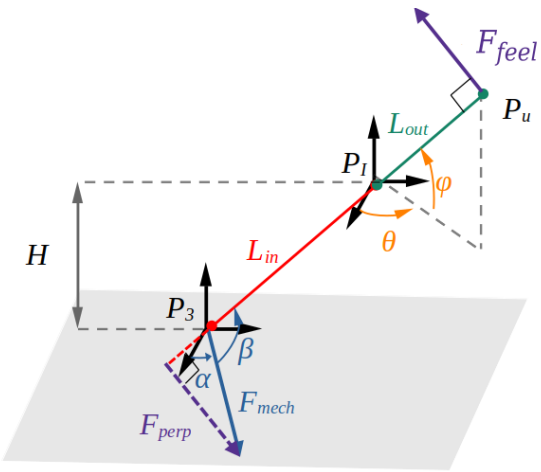

Figure 22: *Diagram of the forces involved*

We assume that the direction of the force that the user needs to feel is known. This is given by the angle α of the previous figure which corresponds to the angle between the force vector $\overrightarrow{F_{mech}}$ and the $\vec{x}_3$ axis.
The needle is linked to the robot structure in P_I as such we have the apparition of a lever effect. Depending on the desired guiding force, the perpendicular component of $\overrightarrow{F_{mech}}$ is :

$$F_{perp} = F_{guide}.\frac{L_{out}}{L_{in}} \quad \text{With : } L_{in} = \frac{H}{\sin(\varphi)}$$

However, the pantograph can only produce forces in the (x_0, y_0) plane, as such we need to determine a relation between the force applied by the mechanism F_{mech} and the orientation of the needle, so that the perpendicular component of F_{mech} on the needle is equal to the desired force felt ($F_{feel} = -F_{perp}$).

Let:

$$\overrightarrow{F_{mech}} = F_{mech}.\hat{v}_{mech} \quad \text{with} \quad \hat{v}_{mech} = \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \\ 0 \end{bmatrix}$$

and :

$$\overrightarrow{P_3P_u} = (L_{in} + L_{out}).\hat{v}_{needle} \quad \text{with} \quad \hat{v}_{needle} = \begin{bmatrix} \cos(\varphi)\cos(\theta) \\ \cos(\varphi)\sin(\theta) \\ \sin(\varphi) \end{bmatrix}$$

Let : Π be the plane formed by the vectors $\overrightarrow{F_{mech}}$ and $\overrightarrow{F_{feel}}$

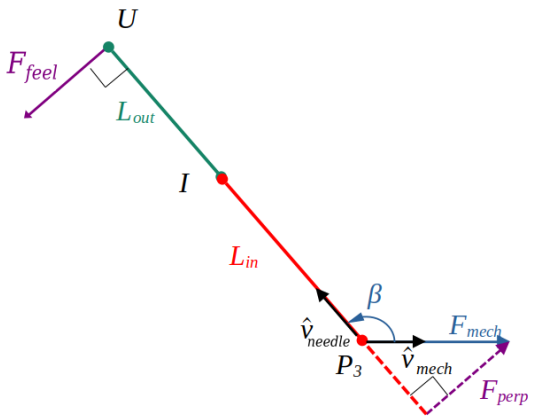

Figure 23: *Diagram of the forces involved viewed in the Π plane*

In this plane it is clear that :

$$F_{mech} = \frac{F_{perp}}{\sin(\beta)} = \frac{F_{perp}}{\sqrt{1 - \cos(\beta)^2}}$$

In figure **23** we can calculate $\cos(\beta)$ as follows :

$$\cos(\beta) = \frac{\hat{v}_{mech}.\hat{v}_{needle}}{\|\hat{v}_{mech}\|.\|\hat{v}_{needle}\|} = \cos(\varphi).\cos(\theta - \alpha)$$

As such we have :

$$F_{mech} = \frac{F_{guide.L_{out}}}{H} . \frac{\sin(\varphi)}{\sqrt{1 - (\cos(\varphi).\cos(\theta - \alpha))^2}} \tag{4.12}$$

Finally, by knowing the orientation of the needle given by the angles θ and φ and knowing the angle α as calculated in section **4.4.3**, we can calculate the force F_{mech} that the pantograph has to produce in order for the user to feel the desired force at point P_u .

We can then use the result of equation **4.8** to determine the torques that we need to apply to the pantograph motors in order to generated the force F_{mech} . As such, we have :

$$\tau = J^T_{panto}(Q).\gamma \quad \text{With : } \gamma = \begin{bmatrix} \overrightarrow{F_{mech}}.\vec{x}_0 \\ \overrightarrow{F_{mech}}.\vec{y}_0 \end{bmatrix}$$

In order to test the validity of the different equations, in the next section we will implement them in *MATLAB* and calculate the different forces and torques involved in the problem. Then, it will be necessary to measure the forces applied by the real system, using a force sensor located at the pantograph end-effector.

Chapter 5: Validation of the models

In order to test the accuracy of the different mathematical models determined in section **4.2**, we first implemented them in *MATLAB* because of the simplicity of the programming. We chose an arbitrary set of joint space coordinates Q and proceeded to calculate the position of all the pantograph joints according to its FKM an then plot the resulting points. Then we take the position of the end-effector point and used it to calculate the actuated joints coordinates according to the IKM. The results are presented in figure **24**.

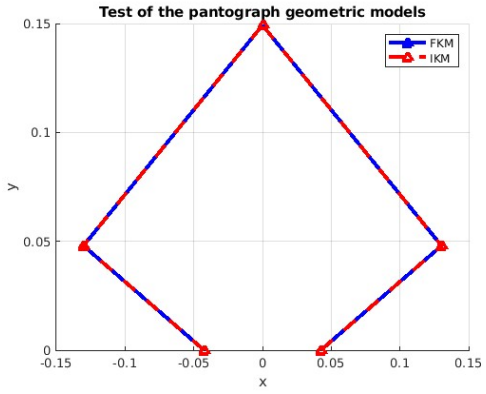

Figure 24: *Test of the pantograph geometric models*

As we can see, the position of the different joints calculated by the FKM and the IKM overlap perfectly thus validating the equations of section **4.2**.

In the same way, to verify the FKM and IKM models of the complete system, we calculate and plot the position of the different joints of the complete system. In the following figure we calculated the position of the different joints in 2 different configurations.

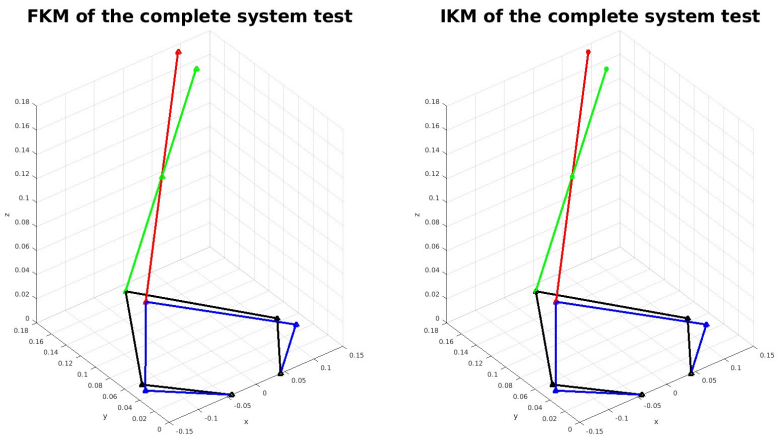

Figure 25: *Test of the geometric models of the complete system*

Given the 3D nature of the system, to verify that the 2 previous graphics overlap, we plot the error graph of the different joints. The error is defined as the difference between the Cartesian positions expressed in the robot world frame of each joint calculated by the FKM and the IKM respectively.

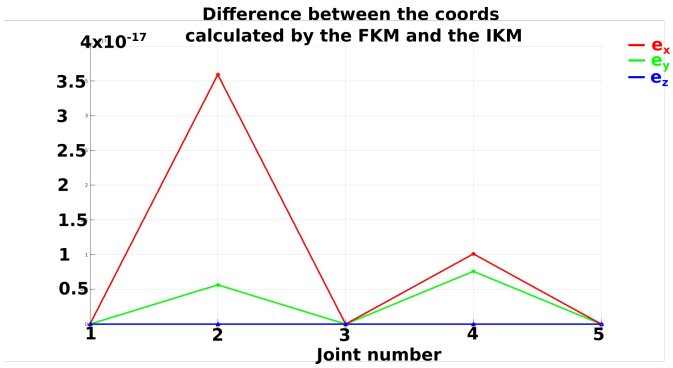

Figure 26: *Error graph of the geometric models*

As we can see, for each joint we get an error in the order of $10^{-}17$ mm, as such the positions calculated by the FKM and the IKM virtually overlap.

In order to test the calculation of the forces involved, we implemented the equations of section **4.4.3** in a *MATLAB* script and plotted the resulting force vectors in 3D using the *plot3* and *quiver* functions :

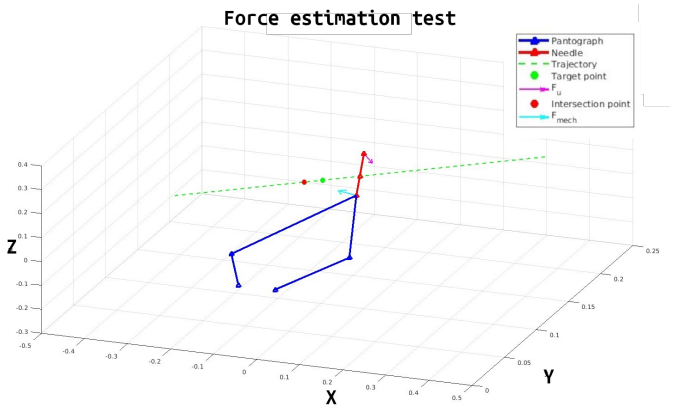

Figure 27: *Test of the force estimation equations*

As we can see, the forces $\overrightarrow{F_u}$, $\overrightarrow{F_{mech}}$ and the needle trajectory belong to a same plane and their orientation is accurate. The magnitude of the forces can be adjusted based on user input (by changing the gain K_p) and as such a calibration phase is needed to ensure the security of the robot and the regulate the intensity of the haptic feedback.

Once the all the calculations for the theoretical models of the robot have been done, the next step is to implement them in a robot application using the *ROS2* framework, as presented in the following section.

5.1 ROS2 implementation :

5.1.1 ROS environment presentation :

The **Robot Operating System** (**ROS**) is a set of software libraries and tools for building robot applications. The latest version, *ROS2*, offers improved performance, real-time capabilities, and enhanced security than its predecessor. The *ROS2* architecture supports a wide range of functionalities essential for robotics, such as **sensor data processing**, **actuator control**, and **state management**. *ROS2*'s modularity and extensive libraries built by the community make it suitable for any kind of robotics applications, from autonomous vehicles and industrial automation to research and education, allowing developers to build **complex**, **scalable**, and **distributed** robotic systems.

In summary, *ROS2* provides an **abstraction layer** between the software and the hardware, that allows the robotic application to be more **versatile** and **modular** eliminating the need to rewrite portions of the code if a new hardware is used. A *ROS2* application can be represented by a graph which corresponds to a network of elements processing data together at the same time. The main elements of this graph are the **nodes**, **topics** and **services**.

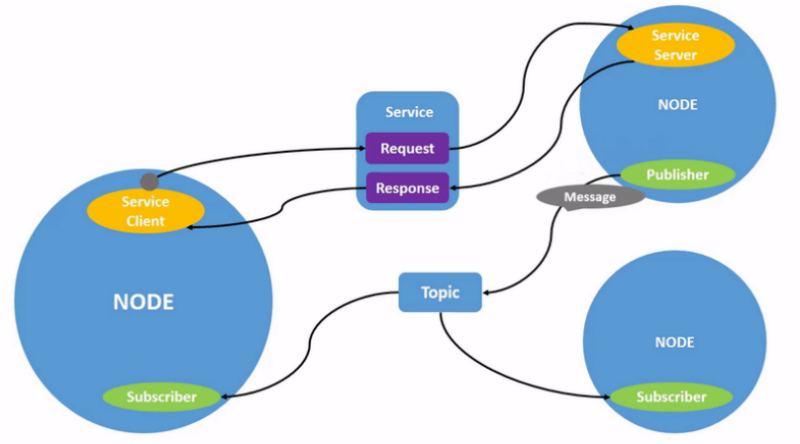

Figure 28: *Generic ROS2 graph*

A **node** represents a program that should have a unique and simple purpose, such as controlling a motor or processing the data coming from an infrared sensor. Each node can send and receive data from the other nodes via **topics**, **services** and **actions**, a more complex node can also have **parameters** which act as the node settings and can be dynamically changed to modify the behaviour of the node.

The main way data is moved around the *ROS2* application is via **topics**. A topic acts as a bus for the node messages which can be of a variety of types (scalar values, vectors, image data, etc). A node can then, **publish** a message in the topic and other nodes receive the message by **subscribing** to the topic.

Services are similar to topics, with the difference that a service only sends the data when a node specifically requests it.

Finally, **actions** are build on services and topics and are intended for long running tasks, the main characteristic of the actions is that they can be canceled and they provide constant feedback.

5.1.2 Project structure :

The first step to build a robot application using the *ROS2* environment is to setup the project structure needed for the application. The project is composed of different **packages** which contains specific parts of the robot application. A project is typically composed of a **description**, a **library** and one or more **controller** packages.

The goal of the **description** package is to establish a model of the robot and its properties. This is done by modelling the robot using the **URDF** (**Unified Robot Description Format**) language. In this language, a robot is described in a single file as a tree of links containing the geometry, joint types and physical properties of the robot. As such, the rest of the software can look for the characteristics of the robot in a single location. The URDF model can also be used to build a simulation of the robot using tools like *Gazebo* and *RViz*.

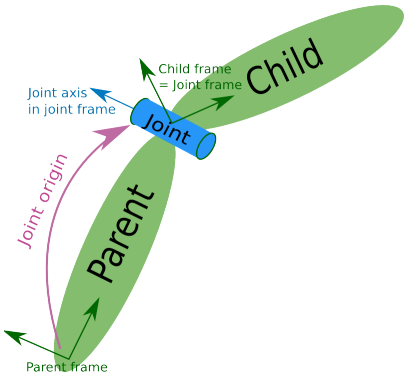

Figure 29: *Diagram of a robot description using URDF*

The **library** package contains the implementation of the different models and parameters of the system as defined in chapter 4 is done in a C^{++} file called *pantograph_model.cpp*. This package will be used by the other parts of the software mainly to calculate the different joint positions for the simulation in *Rviz* and the computation of the torque commands in the controller packages.

Finally, the **controller** packages contain all the programs needed to implement a control architecture using the *ROS2_control* framework.

5.1.3 Controller logic

As detailed in section **4.4.3** we need to define a direct force controller using the *ROS2_control* framework in which controllers are implemented as plugins that conforms to the *ControllerInterface* public interface. The controller must implement 9 public methods that correspond to the controller configuration and allows it to received messages from other topics if needed. The main method of the controller is the **update** method, which is responsible for taking the different inputs (joint positions, joint velocities, angular error, etc) and calculating the commands that have to be sent to the motors of the robot. The update rate of the controller can be set in the configuration file of the controller manager. In order to properly provide a haptic sensation, we set the **update frequency** of the controller to **500 Hz**.

In summary, the **update** method begins by reading the position of the active joints using the motor encoders and estimating the rotational velocities using a simple numerical derivation. Indeed, the controller frequency provides a constant period $\Delta T = 2ms$ at which a position is measured and taking the value of the last position, the rotational velocity can be estimated by the following formula :

$$v(k) = \frac{x(k) - x(k-1)}{\Delta T}$$

With $x(k)$ and $x(k-1)$ the current and last measured position respectively.

Then the controller listens to the topic */angle_error* to retrieve the **angle error** as defined in section **4.4.3**, which is calculated by a separate node. The controller also retrieves the position of the target point defined by the user by listening to the topic */visualization_marker*. Then, using the equations defined in section **4.4.3**, the controller can calculate the torque commands that are then sent to the motors. Additionally, as a security measure, if the calculated torque values are too high, the controller sends an error message and sets the torques to 0, thus preventing the robot from breaking or harming the user.

5.1.4 RViz simulation

In order to provide a visual feedback and verify that the models are accurate, the robot can be displayed in the 3D visualizer *RViz*. This tool listens to the different topics of the *ROS2* project, in order to display the robot in motion as well as the trajectory defined by the user in real time.

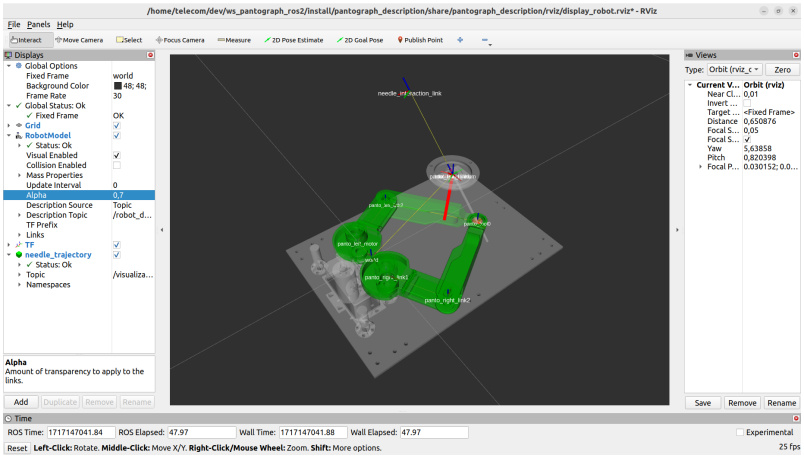

Figure 30: *Visualization of the robot (green) and the trajectory (red) using RViz*

The *RViz* tool can also be used to simulate the behavior of the robot in order to test the different aspects of the project before applying them to the real robot. In figure **30** we can observe, the pantograph 3D model in green, the needle shaft in gray linking the two ball joints in orange and we also have the trajectory defined by the user, in red. This different models are purely used for visualization in our project, however they can also be used to check for collision detection, which could be useful for a future development in the project.

5.1.5 User interface :

In order to allow the students who will use the robot to train in a realistic environment, it is necessary to design an interactive graphical interface, that mimics what you can find in the operation room. The images used for the needle insertion procedures are obtained by a CT scan an are often in the **DICOM** format, a format widely used for this type of medical imaging and whose different characteristics will be detailed below.

The DICOM images are assembled to form a 3D volume from which 3 views (axial, sagittal and coronal) along different axis can be extracted. The user must be able to interact with the different views, and place a point in the 3D volume by clicking on the images which correspond to the target that must be reached by the needle. The coordinates of this point will then be transformed from the image frame (in pixels) to the frame associated with the DICOM file in order to obtain a point in 3D space.

In the DICOM standard, we begin by associating a frame to the patient called *Reference Coordinate System* (RCS) as shown by the following figure :

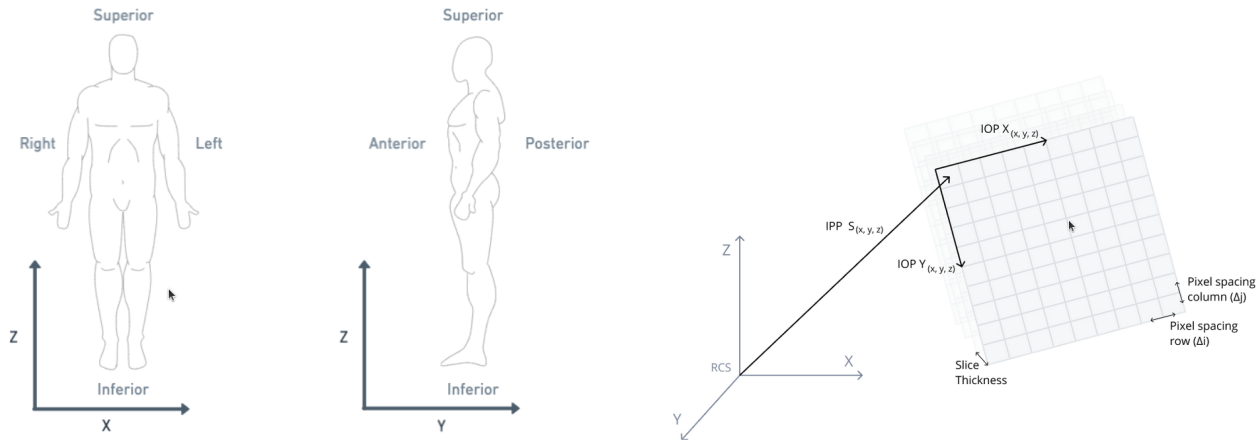

Figure 31: *Definition of the frames and parameters of the DICOM standard*

Then, a CT or MRI scan is taken which produce a slice of the patient's body. The position and orientation of this image is known in relation to the RCS frame and by assembling several DICOM images we can obtain a 3D volume corresponding to the scanned part of the patient's body. To simplify notations, we will refer to the RCS frame as the "patient frame", noted P.

Each image is made up of individual pixels and has 5 parameters that allows it to be completely defined in relation to the patient frame :

- **IPP (Image Position Patient) :** 3×1 Vector that represents the position of the origin of the image (located at the top left corner) in the patient frame
- **IOP (Image Orientation Patient) :** 6×1 Vector containing the 2 axes that define the orientation of the image in relation to the patient frame. By convention the x axis goes from left to right and the y axis is orientated from top to bottom
- $\mathbf{\Delta_i}$, $\mathbf{\Delta_j}$ **:** Physical distance between each pixel along the x and y axes of the image respectively. These coefficients make it possible to transform a distance in pixels on the image to its real value in mm.
- **Slice thickness :** Distance in mm between 2 layers.

As such, in order to transform the position of an object in the image (in pixels) to its position in 3D space

expressed in the patient frame we can apply the following formula:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_1.\Delta_i & Y_1.\Delta_j & 0 & S_x \\ X_2.\Delta_i & Y_2.\Delta_j & 0 & S_y \\ X_3.\Delta_i & Y_3.\Delta_j & 0 & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} i \\ j \\ 0 \\ 1 \end{bmatrix}$$

$$^{P}\tilde{P} = {}^{P}T_{im}.{}^{im}\tilde{P} \tag{5.1}$$

Where:

- ${}^{im}\tilde{P}$ are the homogeneous coordinates of a pixel in the image frame, where *(i,j)* are integers indices associated to the x and y axis respectively.
- ${}^{P}\tilde{P}$ are the homogeneous coordinates associated to the pixel *(i,j)* in the 3D space expressed in the patient frame in mm.
- $S = [S_x, S_y, S_z]^T$ is the position of the image origin in the patient frame.
- $X = [X_1, X_2, X_3]^T$ and $Y = [Y_1, Y_2, Y_3]^T$ are the image axes, written in the patient frame.

We can therefore use the transformation matrix ${}^{P}T_{im}$ to convert the pixels of the different images into 3D points whose coordinates correspond to physical distances.

In order to create the GUI for our project, we build upon the open source DICOM viewer project made by Wen-Ya Lin, researcher at the University of Taiwan, **[14]**. This program was written entirely in Python and uses the module *pydicom* to read and extract the useful information of the DICOM images. This viewer also has many useful features already implemented, mainly an intuitive GUI made with the Python *PyQt* library that calculates and displays the axial, sagittal and coronal views of the DICOM volume. As such, we just need to add the buttons that allow the user to select the insertion and target points in the DICOM volume, calculate their respective coordinates in the patient frame, then convert them to the robot frame and finally publish the coordinates in a *ROS2* topic.

To achieve this, first we need to create and configure a Python *ROS2* package in which we create a simple publisher node called *pointPublisher* that takes a 2 set of points coordinates and publishes them to the topic */image_points*. Then we add this node to the *PyQt* app, making sure that the node configuration allows it to run at the same time as the GUI window. Then, using *PyQt designer*, we create the buttons *set_insertion_point* and *set_target_point* which, when clicked, save the current position of the cursor (which correspond to the pixel coordinates of a point in the DICOM volume), extract the *IPP*, *IOP*, Δi and Δj parameters of the appropriate DICOM slice and calculate the 3D position of the point as explained in **5.1**.

However, in most cases the patient frame and the robot frame are not the same. As such, another homogeneous transform is needed as illustrated in figure **32**.

In summary, we have a DICOM volume, which is arbitrarily orientated with respect to the patient frame and we want to place this volume somewhere inside the robot workspace. For this, we need to calculate the homogeneous transform between the robot and the patient frame :

$${}^{W}T_{P} = \begin{bmatrix} {}^{W}R_{P} & {}^{W}\vec{t}_{P} \\ \emptyset_{3x1} & 1 \end{bmatrix}$$

With ${}^{W}R_{P}$ a rotation matrix defined by the user and ${}^{W}\vec{t}_{P}$ the translation between the robot world frame and the patient frame.

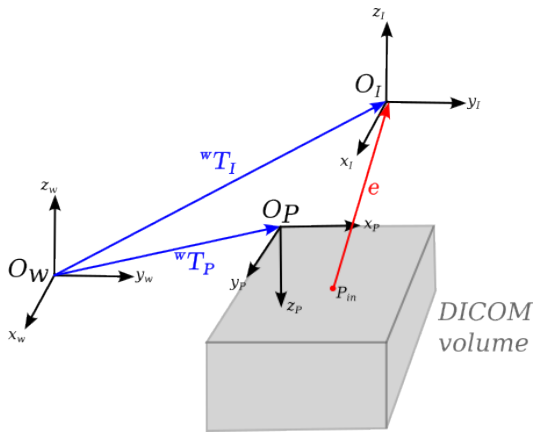

Figure 32: *Placement of the DICOM volume in the robot workspace*

In our case we choose the rotation matrix such that the x and y axis of the DICOM volume are aligned with the ones of the robot world frame. Additionally, when the user clicks on the *set_insertion_point* button, the translation ${}^W\vec{t}_P$ has to be calculated such that the chosen point overlaps with the fixed insertion point of the robot, this is equivalent to choosing the distance $e = 0$ in figure **32** and can be done by solving the following equation :

$$\vec{e} = {}^W P_I - {}^P P_{in} - {}^W\vec{t}_P = \vec{0}$$

Where:

- ${}^W P_I = \overrightarrow{O_w O_I}$ is the position of the insertion point of the robot as defined by the mechanical design, expressed in the world frame.
- ${}^P P_{in} = \overrightarrow{O_P P_{in}}$ is the position of insertion point chosen by the user in the DICOM volume, expressed in the patient frame.

As such, when the user clicks on the *set_target_point* button, the pixel coordinates of the point in the DICOM volume are transform as follows :

$$\tilde{P}_W = {}^W T_P . {}^P T_{im} . \tilde{P}_{im}$$

Then the coordinates of the insertion and target points are published in the topic */image_points* which is read by the node that displays the trajectory in RViz. The following figure shows a demonstration of the complete user interface.

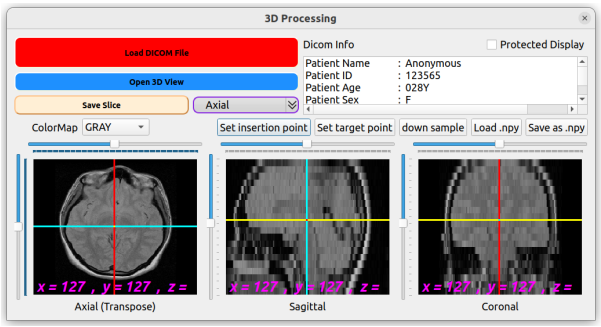

Figure 33: *DICOM Viewer*

Chapter 6: Stereo vision system

In this project, we want to implement a stereo vision system in order to measure the position of the needle tip. Computer stereo vision corresponds to the extraction of 3D information from digital images obtained by cameras, by comparing information about a scene from two different points, similar to the human binocular vision. The 3D information can be extracted by examining the relative positions of objects in the two images and knowing the position and orientation of the cameras relative to each other.

6.1 Pin hole camera model

First of all, the camera model used in this project is the pinhole model because of its simplicity and large implementation in libraries such as *OpenCV*. This *python* library was based on the work done in computer vision by Richard Hartley and Andrew Zisserman in their book *Multiple View Geometry in Computer Vision* **[15]**.

The pinhole camera model describes the relation between the coordinates of a point in 3D space and its projection onto the image plane of an ideal pin-hole camera (a camera whose aperture is described as a point and no lenses are used to focus light). This model is only an approximation and does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and the aperture size. The model validity depends on the quality of the camera and, in general, decreases from the center of the image to the edges as lens distortion effects increase.

However, some of the effects that the pinhole camera model does not take into account can be compensated and others can be neglected if a high quality camera is used.

Additionally, in our application the speed at which the robot end-effector move is relatively slow and the 3D space in which the needle tip moves is small, as such the pinhole camera model can be used as a reasonable description of how a camera depicts the 3D scene.

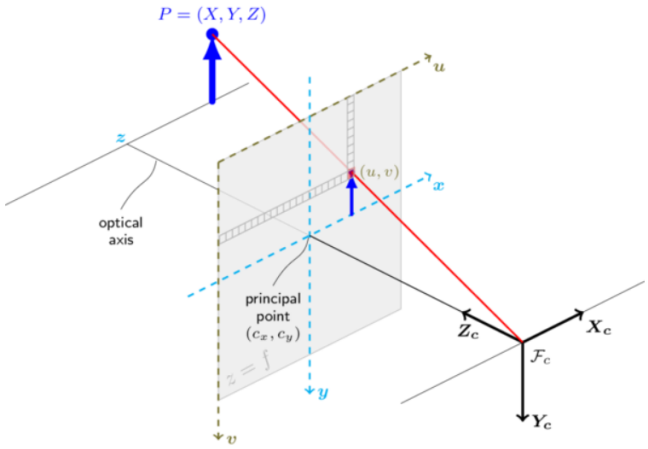

Figure 34: *Pin hole model of a single camera as described in* ***[16]***

The details of the pin-hole model equations can be found in annex **B**.

6.2 Stereo vision model

In traditional stereo vision, two cameras displaced horizontally from one another, are used to obtain two differing views on a scene. By comparing these two images, the relative depth information can be obtained in the form of a disparity map, which encodes the difference in horizontal coordinates of corresponding image points. The values in this disparity map are inversely proportional to the scene depth at the corresponding pixel location.

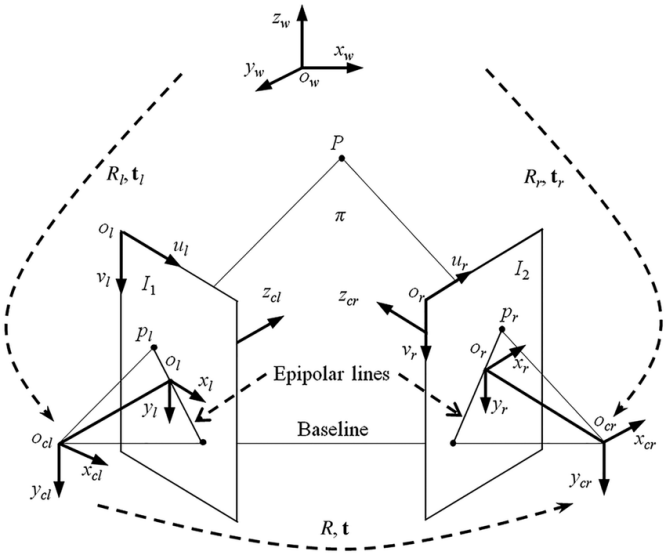

Figure 35: *Stereo vision model*

A computer vision system, follows these steps :

- **Image undistortion :** Elimination of the barrel and tangential distortion (as seen in figure **36**) to ensure that the observed image matches the projection of an ideal pinhole camera.
- **Image rectification :** The image must be projected back to a common plane to allow comparison of the image pairs.
- **Solve the triangulation problem :** Triangulation in computer vision is the process of identifying a point in 3D space from its projections onto two images. The equations can be complex and there is a variety of techniques to solve this problem (SVD decomposition, neural network, etc.) each with different computational cost.

Figure 36: *Different types of image distortion*

6.3 Stereo vision system

6.3.1 Hardware design :

In order to simplify the stereo vision system, we decided to utilize 2 usb cameras that do not need any particular setup to be connected to a PC. Additionally, all the image processing and calculations will be done in a *Python* scrip using the *OpenCV* library.

First, we begin by searching a camera that matches our needs for resolution, frame rate and field of view (FOV). The camera has to be chosen such that the stereo-vision system is not too bulky or expensive, in order to easily mount the vision rig inside the current robot Plexiglass box. As such we chose the camera module based on the OV5640 sensor made by Waveshare and whose specifications can be found in the following table.

Pixel resolution	5 Mp	
Sensor	OV5640	
Camera	Sensor size	1/4 inch
	Aperture (F)	2.8
	Focal length (EFL)	3.29 mm
	Field of View (FOV)	68°
	Distortion	$< 1\%$
image	Static image resolution	2592 x 1944p
	Video recording	15 FPS 2592 $\times$ 1944 30 FPS 1920 $\times$ 1080 30 FPS 1280 $\times$ 720

Table 6.1: *OV5640 Camera specifications*

Then, using the CAD software *CREO*, we design a support to mount the cameras such that the conical workspace of the needle is visible at all times. The support is a simple 3D printed piece, which allows us to accurately know the distance between the 2 cameras and their relative orientation.

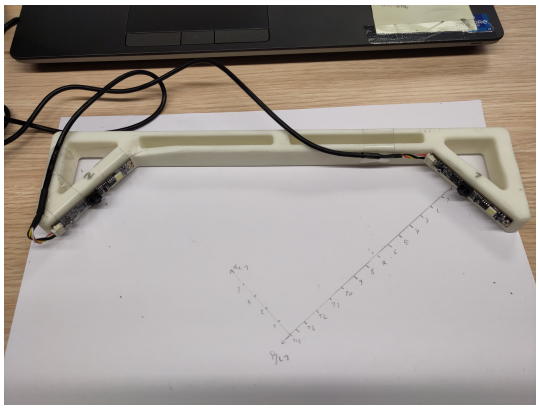

Figure 37: *Stereo vision system using 2 WaveShare OV5640 cameras and a 3D printed support*

6.3.2 Camera calibration :

The next step is to calibrate the 2 cameras in order to get the intrinsic and extrinsic camera parameters that will be used to correct the images distortion and are also needed for solving the triangulation problem. This is done using Stereo camera calibration package made by Dr. Temuge BATPUREV, which consists on various *Python* scripts using *OpenCV*, the software is open source and available at GitHub **[17]**.

Following the instructions on Dr. BATPUREV post, we begin by printing a calibration pattern, which consist of a checkerboard of known dimensions. Then, we acquire multiple images of the calibration board making sure to cover a lot of different positions and orientations.

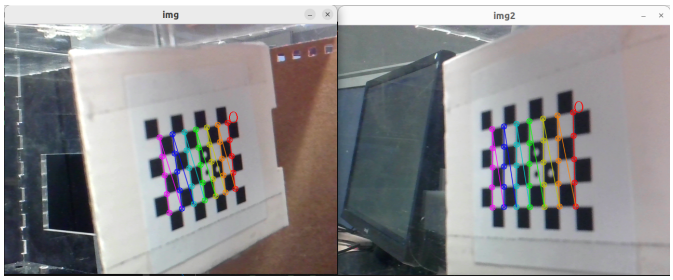

Figure 38: *Stereo camera calibration using Dr. BATPUREV calibration package,* ***[17]***

The calibration software then automatically detects the corners of the checkerboard pattern and computes their respective reprojection according to the camera default intrinsic parameters, as shown by the figure below:

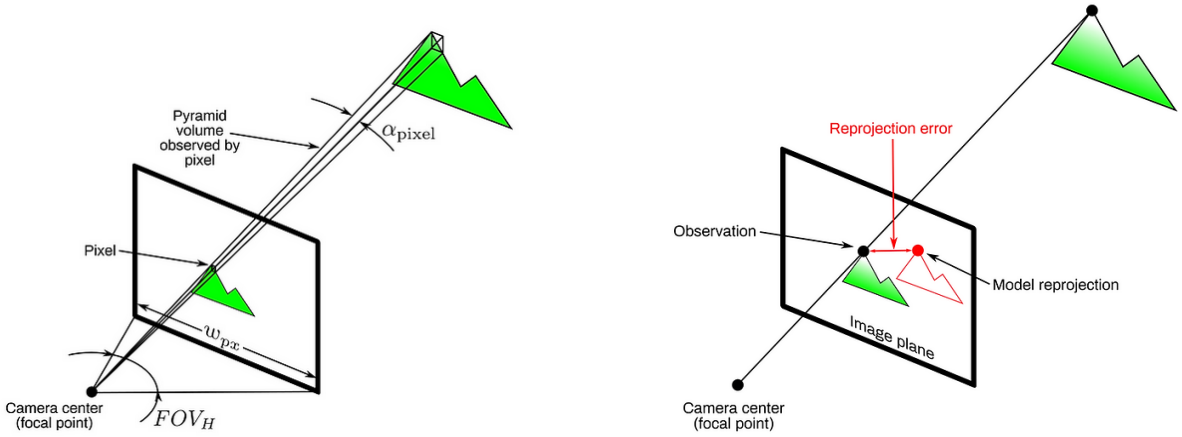

Figure 39: *Volume observed by a single pixel of the camera and reprojection error according to [18]*

Then, the software iteratively modifies the camera parameters in order to minimize the reprojection error, defined as the average L2 norm of point correspondence error in a set of N feature points.

$$e_{RMS} = \frac{1}{N} \cdot \sum_{i=0}^{N-1} \|p_i - q_i\|^2 \tag{6.1}$$

Where p_i are the observed feature points in the image plane and q_i, their respective reprojection using the lens model and calibration results

Additionally, the calibration software allows us to define an intermediate frame using a calibration pattern. This allows us to place a calibration pattern in a well known position relative to the world frame of the robot

and then accurately calculate the homogeneous transformation from the robot world frame to the calibration pattern frame and then from the pattern to each camera frame. This will be useful to transform the coordinates obtained in the triangulation step (which are given in the intermediate frame defined by the pattern), to the the world frame of the robot and display them in Rviz.

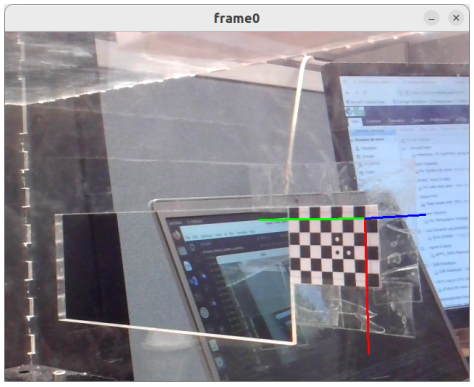

Figure 40: *Automatic definition of the intermediate frame of the stereo-vision rig using a calibration pattern*

The calibration software produces two *.yaml* parameters file with the intrinsic matrix, distortion coefficients and projection matrix of each camera, the *yaml* format allows us to easily change the camera parameters when we recalibrate the stereo-vision system without the need to change the code of the project. The parameters files are then read by the python program that will handle the rest of the calculations.

6.3.3 Marker extraction

Once the calibration step is completed, we need to extract the position of the needle marker in each of the images. To accomplish this we make use of classic image processing techniques for object detection.

As a first approach, we fix a marker to the needle, which is simply a red ribbon wrapped at the needle tip. Then, using the *OpenCV* library, we convert the image pixels to the HSV color space using the *cvtColor()* function and invert the colors of the image. The color inversion is a trick that facilitates the color detection, because in the HSV color space it is easier to detect cyan pixels than red ones. Then we create a mask that only keeps the cyan pixels of the image using the *inRange()* function. Finally, using the *SimpleBlobDetector()* class we can detect and calculate the approximate center of the blob corresponding to the marker.

The lighting conditions and presence of other objects in the scene can greatly affect the colors detection. As such in order to accurately identify the red pixels in the images, we can adjust the HSV threshold values used by the *inRange()* function and make use of morphological transformations (erosion, opening, filtering, etc) that help to reduce the noise in the image. We can also configure the *SimpleBlobDetector* object in order to detect the shape of the marker in the images (a rectangle) and ignore small blobs that could be caused by noise. The following image presents the result of the marker extraction after choosing the appropriate parameters.

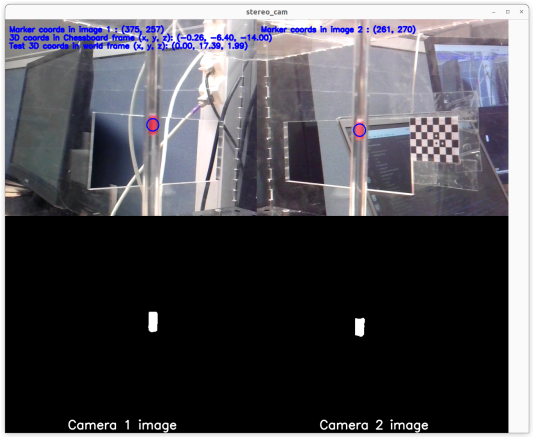

Figure 41: *Extraction of the marker position in the images*

6.3.4 Solve the triangulation problem

Once the position of the marker is known in the two image frames, first we eliminate the distortion in the images caused by the camera lens, using the *OpenCV* function *undistortPoints()* and each camera intrinsic parameters. Then, we use the stereo system extrinsics and the *OpenCV* function *triangulatePoints()* to calculate the 3D position of the marker in the intermediate frame shown in figure **40**. According to the *OpenCV* documentation, the method used to solve the triangulation problem is based on the Direct Linear Transforms (DLT) method, as explained in annexe **C**. Finally, we multiply the coordinates outputted by *triangulatePoints()* by the homogeneous transform matrix from the intermediate frame to the robot world frame, this will be needed to display the marker position in *Rviz*.

6.3.5 Estimation of the error in the 3D reconstruction :

There are many sources of error during the calculation of the 3D position of the needle marker using triangulation which is important to take into account. First, the extraction of the marker position in each camera frame is subjected to noise and uncertainty. As we can see in figure **41**, the detected point is is a blob that changes shape and size depending on its orientation relative to the cameras. As such, the calculation of the center of the marker can be some pixels off from the correct value. However, the impact of this uncertainty is difficult to evaluate. Therefore, in this section we will only use the parameters calculated in the calibration step to evaluate the precision of the stereo-vision system.

To accomplish this, we follow a method developed by Dr. Florent NAGEOTTE, researcher at the ICUBE lab and professor at the Strasbourg University. The method consist on the computation of covariance on 3D point positions obtained from triangulation, the details of the calculations can be found on annex **D**.

Let Σ_{pi} be the covariance matrix of camera 1 or 2 respectively, this matrix can be calculated using the RMS errors obtained in the calibration step (e_{RMS_1} and e_{RMS_2}) by assuming that in each image the error is isotropic, Gaussian and uncorrelated along the two directions of the image. As such we have :

$$\Sigma_{pi} = \begin{bmatrix} {\sigma_i}^2 & 0 \\ 0 & {\sigma_i}^2 \end{bmatrix}$$

With ${\sigma_i}^2 = \frac{e_{RMS_i}^2}{2}$. The stereo images noise covariance is :

$$\Sigma_{im} = \begin{bmatrix} \Sigma_{p1} & 0 \\ 0 & \Sigma_{p2} \end{bmatrix}$$

The covariance on the 3D point position estimation in camera 1 can be calculated as follows :

$$\Sigma_P = (J^T.\Sigma_{im}^{-1}.J)^{-1}$$

Where J is a Jacobian matrix whose computation is explained in annex **D**. Then we compute the singular value decomposition on Σ_P :

$$\Sigma_P = U.S.V^T$$

The matrix of singular values S, contains the variance of the coordinates of the point P along the x, y and z axis, which corresponds to the uncertainty in mm in the position of the point. Additionally, we can define a general "stereo-error" as follows :

$$E_{stereo} = \frac{\sqrt{S(1,1) + S(2,2) + S(3,3)}}{3}$$

To evaluate the impact of this error, we move around the needle marker in the workspace of the robot and calculate its 3D position using triangulation. Then, for each recorded point we calculate E_{stereo} as previously explained, the results are presented in the following figure :

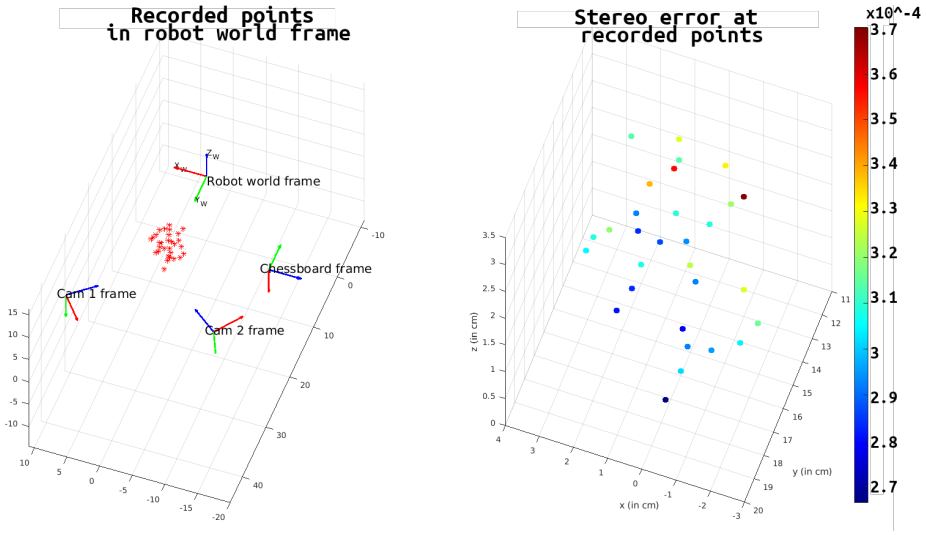

Figure 42: *Estimation of the error on the triangulated position*

As we can observe, the farther away the marker is from the cameras, the higher the error on its reconstructed 3D position. Additionally, the calculation of E_{stereo} doesn't take into account the fact that the noise can affect the 2 images independently. Therefore, in reality the effect of this error could be higher that what we estimated in this section.

6.3.6 ROS2 integration

Finally, the totality of the previous program is included in a *ROS2* node, in order to allow it to publish the 3D position of the marker corresponding to the needle tip as a geometry message of type *tf2* which corresponds to a homogeneous transform. This can then be used by the *Rviz* simulation node and the haptic _controller node. Additionally, the camera images and masks can still be displayed for debugging purposes.

By knowing the length of the needle, we can also estimate the position of the point where the user holds the needle (assuming that the user always holds the needle at the same point), this allow us to calculate in real time the parameter L_{out} of the robot models, which in chapter **4** was considered constant. However, it is important to take into account the uncertainty in the measurement of the position of the needle tip produced by the stereo-vision reprojection error, as explained in section **6.3.5**.

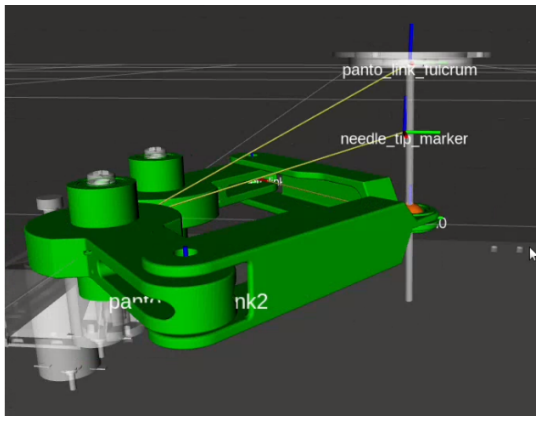

Figure 43: *Needle marker reconstructed position displayed in RViz*

A possible way of correcting this error could be to "snap" the position of the needle marker, to the closest point in the needle shaft, by doing simple vector calculations. However, this was not implemented in the project at the end of my internship.

Chapter 7: Mechanical design

The goal of this chapter is to optimize the mechanical design of the pantograph in order to improve the performance of the control strategy. In fact, the first version of the pantograph segments is bulky, oversized, and the high friction in the joints (especially in the ball joint) required to send a high torque to the motors to make the robot move, which then sometimes caused the robot to oscillate around its target position. As such, using *CREO Parametric 7* we can modify the previous design of the parts to further reduce the friction and the inertia of the different joints, as well as eliminate the mechanical play present in the first version of the ball joints.

7.1 Pantograph proximal segment redesign

In order to avoid spending too much time tuning the design of the parts, we choose to reuse as many components as possible. As such, the different joint axis and corresponding ball bearings were reused. Additionally, the outer diameter of the drum for the capstan system also has to stay equal to 80 mm to keep the transmission ratio chosen in **[2]**. The main differences between the final design of this part and its previous iteration is the increase in the void ratio of the part, decrease of the wall thickness and decrease of the distance between the ball bearings of the motor axis, which only needs to be equal to two times the axis diameter to compensate for the shaft misalignment.

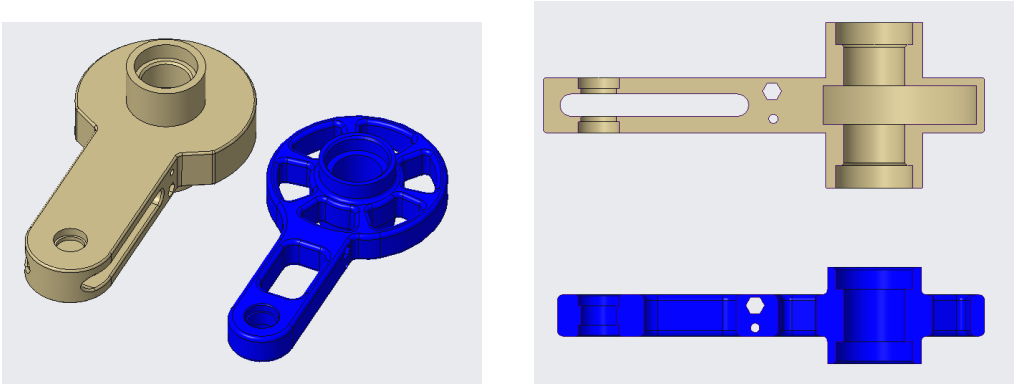

Figure 44: *Comparison of the old (in beige) and new (in blue) versions of the proximal link*

7.2 Pantograph distal segment redesign

For this part the main change is the modification of the part section in order to render it symmetric along the vertical axis. This was done to allow the use of a strain gauge which will be placed on the top surface of the part and will allow us to measure the deformations of the part in the x and y directions. By knowing the material properties (Young modulus, section area, etc), we can then estimate the strain in the part and the forces it is subjected to, thus enabling the implementation of force sensing and more advanced control strategies. However, for time limitations, in this report we do not tackle this challenge and the implementation of the strain gauge will be carried out in a further study by another master student. Nevertheless, we still have to take into account the conditions required for the strain gauge to work correctly in order to simplify the future work which will be done with the pantograph.

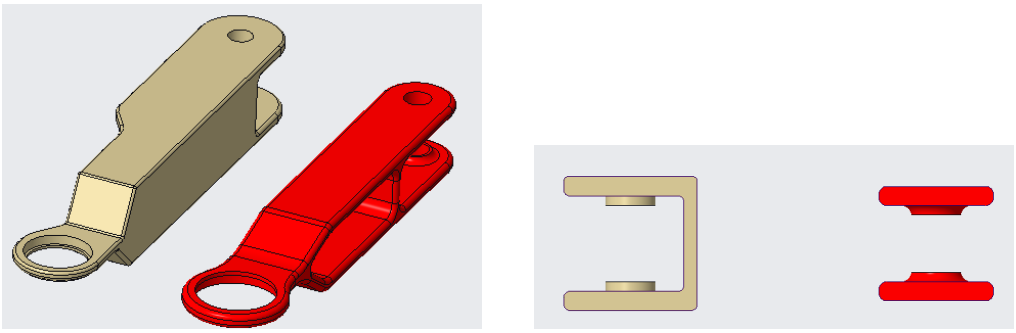

Figure 45: *Comparison of the old (in beige) and new (in red) versions of the dist link*

7.3 Ball joint redesign

The ball joint design presented some unique challenges, the main one is that the contact between the ball and the ball cage has to be without play and have as little friction as possible. The previous ball joint design tried to solve this by 3D printing the ball and the ball cage already assembled using the *PolyJet* printing technology available at the lab. This can produce detailed and smooth parts, however, a small gap between the ball and the cage is needed in order to prevent the parts from fusing together. Despite the simplicity of this design, the resin-resin contact between the ball and the cage produces a lot of friction, this coupled with the mechanical play causes the ball to often be stuck, thus impacting the accurate control of the robot.

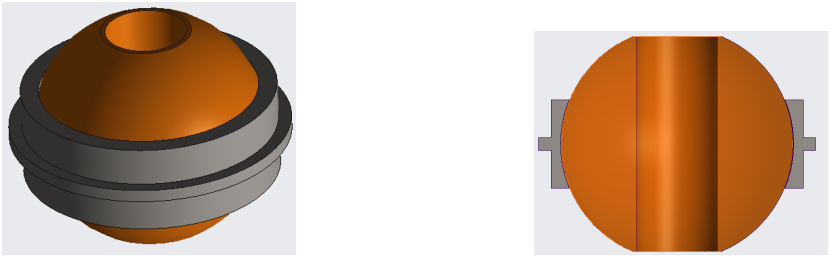

Figure 46: *Previous design of the ball joint (3D printed using PolyJet technology)*

In order to solve this problem, the first step is to change the materials used for the joint. Ideally the ball would be made of steel and the cage of PTFE because the friction coefficient for this particular couple of materials (approximately 0.04) is much lower than the one of a resin-resin contact. Then, to solve the issue of the mechanical play, we designed a system in which the ball is clamped between two PTFE washers, separated by an O-ring which acts like a spring and allows the washers to always be in contact with the ball. The PTFE washers (in orange) are guided inside a support (in red), separated by the O-ring (in black) and clamped together with a screw cap (in grey).

The main challenge of this new design was the limited space available, in fact, despite our best efforts, we had to reduce the radius of the ball from 9.2 mm to 8 mm and increase the internal diameter of the ball bearings from the previous 20 mm to 25 mm to make space for all the parts. Additionally, the screw cap needed to put pressure on the PTFE washers reduces the angular range of the ball from 45° to 39°. However, although the joint does not meet the specifications established in **2.2.2**, it was decided to proceed with this design because in real situations, the angular offset between the needle and the normal to the skin of the patient, rarely exceeds 30°.

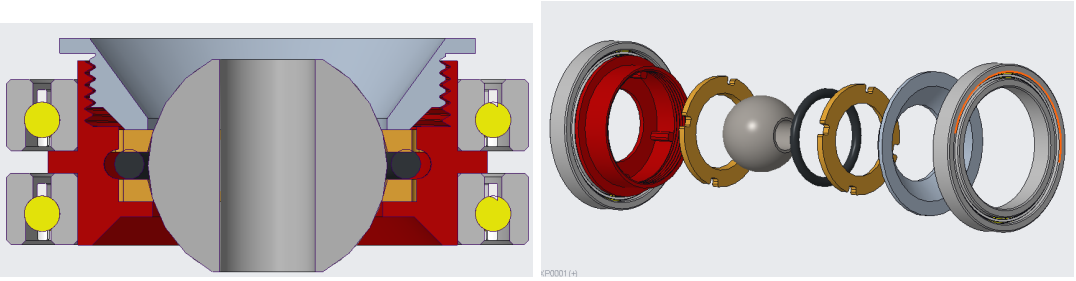

Figure 47: *New ball joint design*

The red and grey parts in figure **47** can be 3D printed using the *Stratasys J826 PolyJet* printer available at the lab. A particular attention was given to threads, indeed it is necessary to take into account the printing tolerances to ensure the correct assembly of the parts. The PTFE washers where planed to be hand made by shaping a solid PTFE bar using a lathe and carving the four guiding notches with a *dremel* tool.

Unfortunately, due shipping delays, the parts needed to assemble the new version of the pantograph did not arrive before the end of my internship. As such, the new design of the ball joint was only tested with 3D printed parts. While these tests confirmed that the new design eliminates mechanical play, the friction remains too high. However, this issue should be resolved when using the proper materials, such as using PTFE for the washers and the ball. Additionally, the lab did not posses the equipment needed make the ball using steel because this requires a CNC lathe and a special carbide drill bit to pierce the steel. As such, the ball was 3D printed as well and sanded to obtain a smooth surface. Still, the friction coefficient between the resin ball and PTFE washer should be significantly lower than the previous resin-resin contact.

Figure 48: *Test of the new ball joint design*

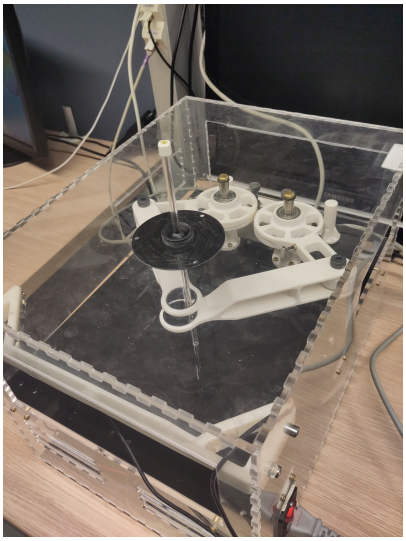

Figure 49: *new version of the Pantograph (only missing the ball joint)*

Chapter 8: Sustainable development and social responsibility analysis

Sustainable development in the design of medical training devices is key to enhancing healthcare education and improving patient safety globally.

Indeed, disparities in the healthcare between industrialized and developing countries are numerous and impact both the quality and accessibility of medical services. In industrialized nations, advanced infrastructure, well-funded research, and widespread access to high-quality training for healthcare professionals contribute to a robust healthcare system. This typically lead to lower mortality rates, better disease prevention programs and more effective treatment options.

In contrast, developing countries often face significant challenges, such as underfunded healthcare systems, a lack of essential medical equipment, and limited access to training for healthcare professionals. This results in higher rates of preventable diseases, insufficient care for chronic conditions, and higher maternal and infant mortality rates. Additionally, the scarcity of specialized training means that medical professionals in these regions may not be equipped to handle complex medical cases, further widening the gap in healthcare outcomes.

By designing accessible medical training devices and promoting open-source collaboration, we can help bridge this gap, enabling healthcare professionals especially those in developing countries, to receive the training they need to improve patient outcomes and advance their healthcare systems.

In fact, the use of open-source software and collaborative platforms like *GitHub* along with open-source licenses such as the *Apache 2.0* licence, fosters innovation and global research by allowing unrestricted access and collaboration on these tools.

As such, the majority of the code developed for this project has been uploaded to the lab's GitHub **[19]** and is freely available for public use. However, certain specialized code required for the hardware used in the lab remains private. Nonetheless, this code is accessible within the lab, allowing team members to continue the project or use it as a reference for other projects.

Figure 50: *Apache foundation and GitHub logos*

Chapter 9: Conclusion

9.1 Discussion :

In this project we reviewed the previous work done by T. CESARE **[2]** in order to improve various aspects of the the device. First, we started by modifying the mathematical models to adopt a more coherent notation and also to add the modelling of the needle. Then we designed a simple force based controller using the concept of virtual fixtures in order to implement a haptic guidance system on our project. Then, we designed, built and tested a stereo-vision system using *Python* and *OpenCV*, in order to track the position of the needle in real time. Finally, we modified the mechanical design of the pantograph in order to further reduce the inertia and friction of the parts.

Many of the software aspect of this project was done using *ROS2* while making sure to follow the coding guidelines of the ICUBE lab in order to share the work in the lab's GitHub and make it modular and usable for future projects.

At the end of my internship, the software aspect of the project is finished and ready for a demonstration. However due to time constraints and shipping delays, I wasn't able to finish assembling the new version of the pantograph. As such, some work still needs to be done, mainly ensuring that the ball joints work as intended. Indeed, it is necessary to replace the 3D printed washers and ideally also the ball itself, with pieces made of PTFE that can be fabricated using the lathe available at the lab. Other quality of life improvements can be made to the project, such as a more intuitive GUI in which the position and orientation of the needle is overlayed in real time, over the DICOM images. In addition, there's still work to be done with the pantograph mathematical models. Indeed, a future development on this project would be to determine the dynamic model of the pantograph according to the method presented in **[12]** in order to implement more advanced control techniques such as impedance control.

After implementing all these improvements, the device must be tested with students and practitioners to confirm that the haptic feedback provides a realistic training experience and enhances users' overall accuracy when performing IR procedures.

9.2 Personal conclusion :

This internship at the ICUBE lab has been an invaluable experience, allowing me to explore the research domain in robotics and healthcare, gaining essential skills in not only building a robotic application but also effectively sharing the progress done in order to contribute to the research community. Working in the field of medical robotics aligns perfectly with my career aspirations and personal values, particularly my commitment to making robotics more accessible and helping people around the world. As such, I am excited to continue my journey in this field by pursuing a PhD in soft robotics at INRIA institute in Lille.

Annexes

A Organizational chart of the ICUBE lab

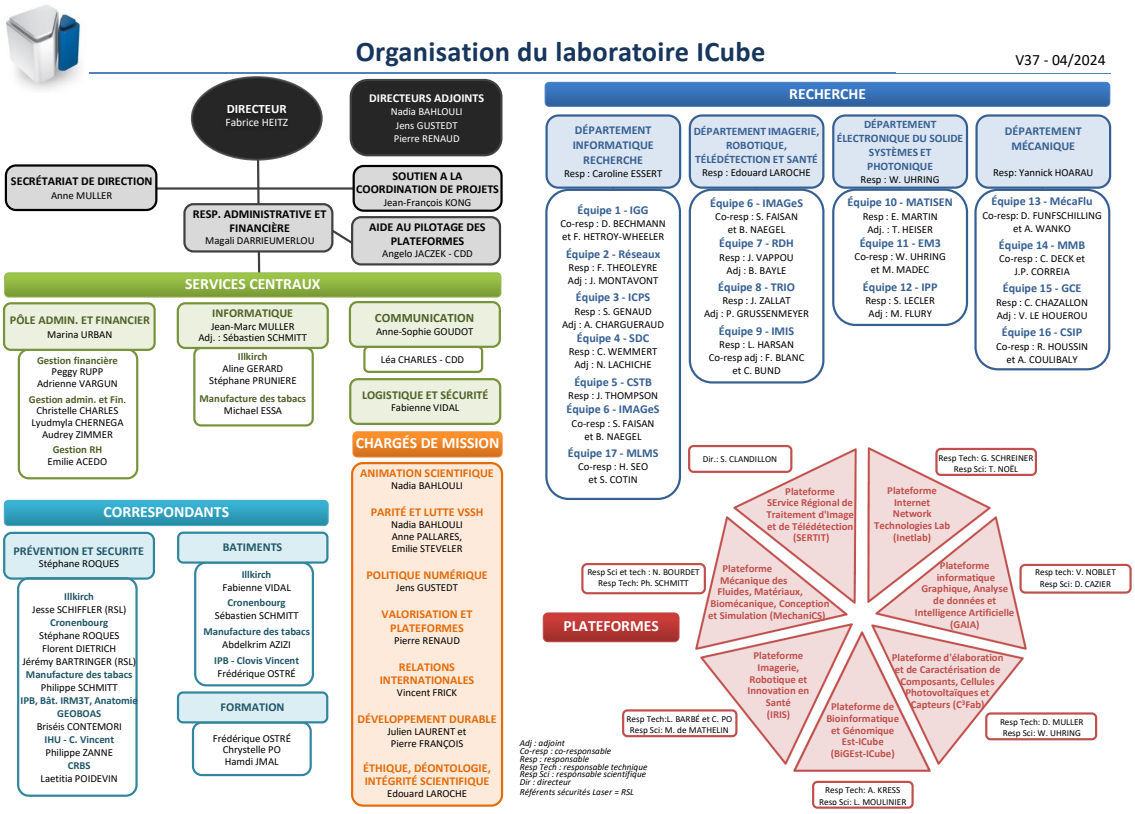

Figure 51: *Organizational chart of the ICUBE lab in 2024*

B Pin hole camera model

Extract from the open CV documentation **[16]**:

The functions in this section use a so-called pinhole camera model. The view of a scene is obtained by projecting a scene's 3D point P_w into the image plane using a perspective transformation which forms the corresponding pixel p. Both P_w and p are represented in homogeneous coordinates, i.e. as 3D and 2D homogeneous vector respectively. [...] For more succinct notation, we often drop the 'homogeneous' and say vector instead of homogeneous vector.

The distortion-free projective transformation given by a pinhole camera model is shown below.

$$^{s}p = A[R|t]P_w$$

Where P_w is a 3D point expressed with respect to the world coordinate system, p is a 2D pixel in the image plane, A is the camera intrinsic matrix, R and t are the rotation and translation that describe the change of coordinates from world to camera coordinate systems (or camera frame) and s is the projective transformation's arbitrary scaling and not part of the camera model.

The camera intrinsic matrix A (also generally notated as K) projects 3D points given in the camera coordinate system to 2D pixel coordinates, i.e.

$$p = AP_c$$

The camera intrinsic matrix A is composed of the focal lengths f_x and f_y, which are expressed in pixel units, and the principal point (c_x, c_y), that is usually close to the image center:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

And thus

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

The matrix of intrinsic parameters does not depend on the scene viewed. So, once estimated, it can be re-used as long as the focal length is fixed (in case of a zoom lens). Thus, if an image from the camera is scaled by a factor, all of these parameters need to be scaled (multiplied/divided, respectively) by the same factor.

The joint rotation-translation matrix $[R|t]$ is the matrix product of a projective transformation and a homogeneous transformation. The 3-by-4 projective transformation maps 3D points represented in camera coordinates to 2D points in the image plane and represented in normalized camera coordinates $x' = X_c/Z_c$ and $y' = Y_c/Z_c$:

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

The homogeneous transformation is encoded by the extrinsic parameters R and t and represents the change of basis from world coordinate system w to the camera coordinate system c. Thus, given the representation of the point P in world coordinates, P_w, we obtain P's representation in the camera coordinate system, P_c, by

$$P_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_w$$

[...]

Putting the equations for instrincs and extrinsics together, we can write out ${}^s p = A[R|t]P_w$ as :

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} . \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

If $Z_c \neq 0$, the transformation above is equivalent to the following :

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{X_c}{Z_c} + c_x \\ f_y \frac{Y_c}{Z_c} + c_y \end{bmatrix}$$

with :

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

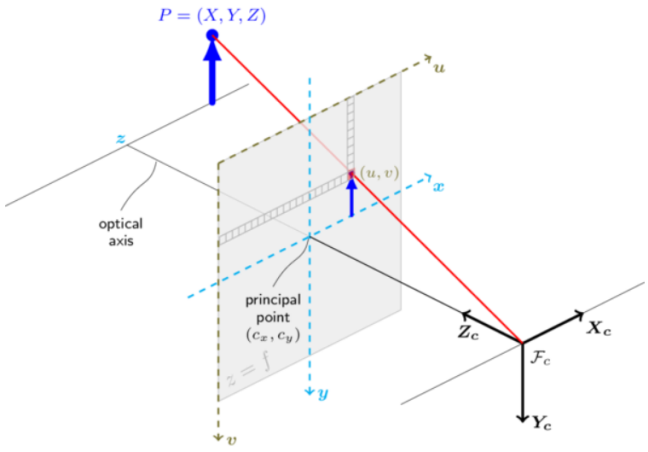

Figure 52: *Pin hole model of a single camera as described in* ***[16]***

Real lenses usually have some distortion, mostly radial distortion, and slight tangential distortion. So, the above model is extended as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix}$$

where

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2(r^2+2x'^2) + s_1 r^2 + s_2 r^4 \\ y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1(r^2+2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4 \end{bmatrix}$$

with

$$r^2 = x'^2 + y'^2$$

and

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \end{bmatrix}$$

if $Z_c \neq 0$

The distortion parameters are the radial coefficients k_1, k_2, k_3, k_4, k_5 and k_6, p_1 and p_2 are the tangential distortion coefficients, and s_1, s_2, s_3 and s_4 are the thin prism distortion coefficients. Higher-order coefficients are not considered in OpenCV.

C Solving the triangulation problem using Direct Linear Transforms

Extracted from Dr. T.BATPUREV blog, **[20]**.

C.1 Motivation for DLT

DLT is a method for calculating a matrix equation of the form $AX = 0$, where A is some matrix and $X$is the vector unknowns that we want. This problem setting occurs in many forms in photo-grammetry.

Suppose we have a 3D point in real space with coordinates given as $X = [x, y, z, w]^T$ in homogeneous coordinates. Suppose we observe this point through two cameras, which have pixel coordinates $U_1 = [u_1, v_1, 1]^T$ for camera 1 and $U_2 = [u_2, v_2, 1]^T$ for camera 2. Using the camera projection matrix P_1, we can write U_1 as :

$$U_1 = \alpha P_1 X$$

In a triangulation problem, we don't know the coordinates of X. But we can determine the pixel coordinates and also assume we found the projection matrix through camera calibration. Our task is then to determine the unknowns in X. Since U_1 and P_1X are parallel vectors, the cross product of these should be zero. This gives us:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} .X = \begin{bmatrix} v_1 p_3 & p_2 \\ p_1 & u_1 p_3 \\ u_1 p_2 & v_1 p_1 \end{bmatrix} .X = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

I've written the row vectors of P_1 as p_i, which are 4 dimensional vectors. This gives us an equation of the form $AX = 0$. But the third row is a linear combination of the first two rows which then only gives 2 systems of equations, which is not enough to solve the 3 unknowns in X. This is expected, since we can't determine a 3D coordinate from a single camera view. Since we have two cameras, we can extend the matrix to have more rows. In fact, we simply add on more rows for any number of views. This gives us the equation:

$$AX = \begin{bmatrix} v_1 p_3 & p_2 \\ p_1 & u_1 p_3 \\ u_1 p_2 & v_1 p_1 \\ \vdots & \vdots \end{bmatrix} .X = 0$$

In camera triangulation, we are given A and we want to determine X. In this setting, we use DLT to determine X.

C.2 Direct Linear Transform

We want to obtain the non-trivial solution of an equation of the form $AX = 0$. In the real world, there can be some noise, so we write the equation as $AX = w$, and we solve for X such that w is minimized. The first step is to determine the SVD decomposition of A.

$$AX = USV^T X$$

Our goal is to minimize w for some x. This can be done by taking the dot product:

$$w^T w = (W^T V S U^T)(USV^T X) = X^T V S^T V^X$$

Remember that U and V are orthonormal matrices and S is a diagonal matrix. Moreover, the entries on the diagonal of S are decreasing, so that the last entry on the diagonal is the minimum value. These are guaranteed by the SVD decomposition. Exploiting the property that V is an orthonormal matrix, if we simply select X to be one of the column vectors of V^T

$$v_i^T V S^T V^T v_i = s_i^2$$

In the above equation, I've written the i'th entry on the diagonal of S as s_i. Since our goal was to minimize $w^T w$, this tells us that it is equivalent to choosing the smallest value of S^2 by selecting the corresponding v_i column vector of V^T as X. In other words, the minimum value is obtained if we choose the last column vector of V^T as X. Thus we have solved the $AX = w$ equation in the presence of noise. If there is no noise, this SVD method will still work.

D Computation of the covariance on 3D points positions obtained from triangulation

We consider the estimation of 3D point positions from stereoscopic images. The geometry of the acquisition system is described by the following parameters :

- K_1 and K_2 : Intrinsic matrix of the two cameras
- ${}^2R_1 = [r_1, r_2, r_3]^T$ and ${}^2t_1$: Rotation and translation from camera 2 to camera 1 expressed in camera 2 frame
- e_{RMS_1} and e_{RMS_2} : The reprojection errors calculated in the calibration of camera 1 and 2 respectively.

The noise from the image extraction process is ignored.

Let p_1 and p_2 be two corresponding points in image 1 and 2. Let $P_{cam1} = [X_1, Y_1, Z_1]^T$ be the reconstructed 3D position of the point calculated by the triangulation using points p_1 and p_2, expressed in camera 1 frame. By knowing the stereo-vision system extrinsics we have :

$$P_{cam2} = {}^2R_1.P_{cam1} + {}^2t_1 = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}$$

Let m_1 and m_2 be the image points given in mm in the normalized image planes. We have for $i \in [1, 2]$:

$$m_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{X_i}{Z_i} \\ \frac{Y_i}{Z_i} \\ 1 \end{bmatrix}$$

$$p_i = K_i m_i$$

Then by differentiation of the measurements (p_1 and p_2) with respect to the P_{cam1} we obtain :

$$J_1 = \frac{\partial p_1}{\partial P_{cam1}} = K_1(1:2, 1:1). \begin{bmatrix} \frac{1}{Z_1} & 0 & \frac{-X_1}{{Z_1}^2} \\ 0 & \frac{1}{Z_1} & \frac{-Y_1}{{Z_1}^2} \end{bmatrix}$$

Where $K_1(1:2, 1:2)$ is the upper corner of K_1. In the same way we have :

$$J_2 = \frac{\partial p_2}{\partial P_{cam1}} = K_2(1:2, 1:1). \begin{bmatrix} \frac{r_{11}}{Z_2} - \frac{X_2.r_{31}}{Z_2^2} & \frac{r_{12}}{Z_2} - \frac{X_2.r_{32}}{Z_2^2} & \frac{r_{13}}{Z_2} - \frac{X_2.r_{33}}{Z_2^2} \\ \frac{r_{21}}{Z_2} - \frac{Y_2.r_{31}}{Z_2^2} & \frac{r_{22}}{Z_2} - \frac{Y_2.r_{32}}{Z_2^2} & \frac{r_{23}}{Z_2} - \frac{Y_2.r_{33}}{Z_2^2} \end{bmatrix}$$

Let $\Sigma_p i$ be the 2x2 covariance matrix on the image i points. It can be obtained from the RMS errors e_{RMS_1} and e_{RMS_2} by assuming that in each image the error is isotropic, Gaussian and uncorrelated along the two directions of the image. As such one has :

$$\Sigma_{pi} = \begin{bmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix}$$

With $\sigma_i^2 = \frac{e_{RMS_i}^2}{2}$ (the RMS error accounts for the two dimensions, which explain why it should be divided by 2).

Noting the stereo images noise covariance as :

$$\Sigma_{im} = \begin{bmatrix} \Sigma_{p1} & 0 \\ 0 & \Sigma_{p1} \end{bmatrix}$$

and defining $J = [J_1, J_2]^T$, the covariance on the 3D point (noted P) position estimation in camera 1 is given by :

$$\Sigma_P = (J^T \Sigma_{im}^{-1} J)^{-1}$$

Note that in general it is not meaningful to estimate an RMS error on the reconstructed 3D point as this error is usually not isotropic along the 3 spatial dimensions, with a larger uncertainty along the z axis of the central frame of the stereoscopic system. Nevertheless, if one wants to grasp the uncertainty on the 3D position with a single value, one can proceed the following way:

- Compute the eigen values decomposition (or singular values decomposition) of Σ_P :

  $$\Sigma_P = USV^T$$

  The matrix of singular values (or eigen values) contains the variance of P along the principal axes.

- Then $E_{P_{RMS}} \approx \frac{\sqrt{S_{1,1}+S_{2,2}+S_{3,3}}}{3}$

Bibliography

[1] Meghavi Mashar, Andrew Nanapragasam, and Philip Haslam. "Interventional radiology training: Where will technology take us?" In: *BJR|Open* (2023) (cit. on p. 15).

[2] Thomas CESARE. "Design and control of a training device for needle insertion". In: (2023) (cit. on pp. 16–18, 22, 28, 32, 51, 55).

[3] G. Campion, Qi Wang, and V. Hayward. "The pantograph mk-II: A haptic instrument." In: *EEE/RSJ International Conference on Intelligent Robots and Systems* (2005) (cit. on p. 16).

[4] *Quanser 3 DoF pantograph*. `https://docs.quanser.com/quarc/documentation/quanser_3dof_pantograph_blocks.html`. 2022 (cit. on p. 16).

[5] Pierre Letier et al. "Survey of actuation technologies for body-grounded exoskeletons." In: *Eurohaptics Conference* (2006) (cit. on p. 17).

[6] *Micromate device by Interventional systems*. `https://www.interventional-systems.com/`. 2024 (cit. on p. 19).

[7] Chin-Hsing Kuo, Jian S. Dai, and Prokar Dasgupta. "Kinematic design considerations for minimally invasive surgical robots: an overview." In: *The International Journal of Medical Robotics and Computer Assisted Surgery* (2012) (cit. on p. 19).

[8] David Escobar-Castillejos et al. "A review of simulators with haptic devices for medical training." In: *Journal of Medical Systems* (2016) (cit. on pp. 19, 20).

[9] Cléber G. Corrêa et al. "Haptic interaction for needle insertion training in medical applications: The state-of-the-art." In: *Medical Engineering & Physics* (2019) (cit. on p. 19).

[10] Joss Moo-Young et al. "Development of unity simulator for epidural insertion training for replacing current lumbar puncture simulators." In: *Cureus* (2021) (cit. on p. 20).

[11] Dong Ni et al. "A virtual reality simulator for ultrasound-guided biopsy training." In: *IEEE Computer Graphics and Applications* (2011) (cit. on p. 20).

[12] Bruno Siciliano et al. *Robotics: Planing, Modelling and Control*. London: Springer-Verlag, 2009 (cit. on pp. 23, 29, 55).

[13] Stuart A. Bowyer, Brian L. Davies, and Ferdinando Rodriguez y Baena. "Active Constraints/Virtual Fixtures: A Survey". In: (2014) (cit. on pp. 29, 30).

[14] Wen-ya Lin. *Python + QT Dicom viewer*. `https://github.com/wenyalintw/Dicom-Viewer/blob/master/readme.md`. 2019 (cit. on p. 41).

[15] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge: Cambridge University Press, 2003 (cit. on p. 43).

[16] *OpenCV documentation*. `https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html`. 2024 (cit. on pp. 43, 57, 58).

[17] Temuge Batpurev. *Stereo calibration Python script*. `https://github.com/TemugeB/python_stereo_camera_calibrate`. 2022 (cit. on p. 46).

[18] *Camcalib documentation*. `https://www.camcalib.io/post/what-is-the-reprojection-error`. 2024 (cit. on p. 46).

[19] *Icube Lab's GitHub for the pantograph project*. `https://github.com/ICube-Robotics/needle_pantograph_ros2`. 2024 (cit. on p. 54).

[20] Temuge Batpurev. *Direct Linear Transforms*. `https://temugeb.github.io/computer_vision/2021/02/06/direct-linear-transorms.html`. 2021 (cit. on p. 60).

Résumé

La radiologie interventionnelle (RI) est une spécialité relativement nouvelle qui présente de nombreux avantages par rapport à la chirurgie standard (réduction du temps de récupération, réduction des risques pour le patient, etc.). Le but de ce projet est de développer un dispositif d'entraînement pour l'insertion d'aiguilles afin d'aider à la formation des étudiants de médecine aux procédures RI. Le prototype consiste en un robot pantographe plana, qui tient une aiguille à son effecteur. Le robot produit ensuite une force de guidage sur l'aiguille qui fournit un retour haptique à l'utilisateur et l'aide à rester sur une trajectoire prédéfinie. Mon travail vise à améliorer la version précédente du système en modifiant les modèles mathématiques du robot, en ajoutant une interface utilisateur et en developpant un contrôleur en force pour fournir un retour haptique à l'aide du framework ROS2. De plus, j'ai construit un système de vision stéréoscopique qui permet de mesurer la position 3D de l'aiguille en temps réel et j'ai refait certaines pièces du système pour réduire davantage le poids global et la friction.

Abstract

Interventional radiology (IR) is a relatively young speciality, that presents many advantages compared to standard surgery (faster recovery time, less risk for the patient, etc). The goal of this project is to develop a training device for needle insertion to assist in the training of medical students in IR procedures. The prototype consists of a planar pantograph robot, that holds a needle at its end effector. The robot will then produce a guiding force on the needle which provides haptic feedback to the user and assist them to stay on a predefined path. My work aims to improve the previous design of the system by modifying the robot mathematical models, adding a user interface and writing a force based controller to provided haptic feedback using the ROS2 control framework. Additionaly, I build a stereo-vision system that tracks the 3D position of the needle in real time and redesigned some parts of the system to further reduce the overall weight and friction.