FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF MECHANICAL AND MECHATRONICS ENGINEERING

N-2020 Blood Vessel Detector

Prepared by:

Judah Sleibi 1180678

Marian Asbah 1181548

Tasneem Badwan 1180144

Supervised by:

Dr. Ihab Abu Ajameih

A Graduation Project submitted to the Mechanical and Mechatronics Engineering Department in partial fulfillment of the Requirements for the degree of B.Sc. in Mechanical/Mechatronics Engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

وعلى اعتابِ النهايةِ سأتوقفُ قليلاً التقطتُ من انفاسي قليلاً اطالعُ ماضي القريبَ والبعيدَ قائلاً فعلتها نعم فعلتها، وصلت لها تلك النهاية حُسن الخاتمة ان اجدتُ وصفها، لم يكن ما قضيناه من سنينٍ في سبيل هذا الحلم في سبيل هذا الختام نزهةً بل كانت معركة لا رحمةَ فيها، جهاداً ان اصابت اقوالي فكانت حرباً لا استسلام فيها. فاليوم انت بقارئ لما انا بكاتبٍ فهذا خير دليل على اني مجاهدٌ عنيد مقاتل شريف فها انا اليوم اقفُ هازماً لا مهزوماً مسنوداً من اب وام من اخ واخت من صديق وقريب، احاصرُ الصعاب لا هي من تحاصرني لست وحدي كما كنت قبلاً لم اعد ضعيفاً فسندي سندٌ ثابتٌ لا يميل .

وكي لا اطيل بالكلامَ بالكثير سأبدئ شاكراً الى عائلتي الى ابي والى امي الى من اسرف من الليالي في رعايتي وتعليمي يا من حملني في دفئ شديد يا من اعطاني القليل والكثير فشكري بدايةً سيكون الى عائلتي الى ابي وامي مصابيحُ نجاحي، منتقلاً الى من حمل على عاتقه رسالة الامانةِ والتعليم طاقمنا التدريسي المقور واشد بالشكر والعرفان الى الدكتور ايهاب ابو عجمية مشرف المشروع على كل ما قدم لإنجاح هذا العمل الهندسي. وكيف أنسي واغلق اقلامي وابعد اوراقي قبلما ان اوجه كلمة شكرٍ الى اصدقاء الكلية وزملاء التخصص على جعلهم هذه السنين خالدة في الذاكرة تقبع في اعماقها محفوظة.

ABSTRACT

With increased reliance on the medical laboratory tests as a primary step for medical diagnoses, the number of venipuncture and cannulation processes performed annually increased significantly. However, the success rates of the first trial injection are heavily dependent on clinician skill and patient physiology, skin color, mass body weight, gender, etc. According to studies, venipuncture failure rates can reach up to 27% of patients without visible veins, 40% of patients without palpable veins, and 60% of emaciated patients, even when done by professionals.

To improve the success rate of the venipuncture process, the project tends to develop an autonomous venipuncture device that can robotically inject a needle into a suitable vein under nurse supervision that selects the vein from the processed image. The project consists of research-based and system-designing parts. The research part mainly focuses on studying the capability of NIR light to safely penetrate through the skin layers up to the Subcutaneous tissue where blood vessels lie and its benefits in obtaining a clear and distinct image of the patient's veins at the anti-cubital fossa. The system-designing part is concerned with the mechanical designing of the 6 Degrees of Freedom robot and performing the necessary kinematics and dynamics studies and control; to insecure the robot capability to move to the desired injection point.

المستخلص

في الوقت الراهن تعتمد أغلب التشخيصات المرضية بشكل رئيسي على الفحوصات الطبية التي تتم من خلال تحليل عينات الدم المسحوبة من أوردة المريض باستخدام الابر الطبية المخصصة التي لطالما تكون هذه العملية مصحوبة بقلق وألم وذلك بسبب محاولة الطاقم الطبي إيجاد الوريد المناسب وحقن الابرة من خلاله بعد عدة محاولات مؤلمة.

مع زيادة الاعتماد على الفحوصات المخبرية الطبية كخطوة أولية للتشخيص الطبي، ازداد عدد عمليات سحب الدم والحقن بالإبر و Cannulations التي تتم سنويًا إلى حد بعيد . ومع ذلك، فإن معدلات نجاح الحقن التجريبي الأول تعتمد بشكل كبير على مهارة الطبيب وفسيولوجيا المريض، ولون الجلد، ووزن الجسم، والجنس، إلخ. ووفقاً للدراسات، فإن معدلات فشل سحب الدم والحقن بالإبر قد تصل إلى 27% من المرضى الذين لا يملكون أوردة مرئية، و40% من المرضى الذين لا يملكون أوردة ملموسة، و60 %من المرضى الذين يعانون من الهزال، حتى عندما يتم ذلك على يد محترفين.

لتحسين معدل نجاح عملية الحقن بالإبر، يميل المشروع إلى تطوير ذراع آلي يحقن الإبر بشكل المستقل أي يمكن أن يحقن إبرة من خلاله في الوريد المناسب الذي يختاره طاقم طبي من الصورة المعالجة.

يقوم المشروع على البحث وتصميم أجزاء للنظام المارد استخدامه. بحيث يركز الجزء البحثي بشكل رئيسي على دراسة قدرة ضوء الأشعة تحت الحمراء على اختراق طبقات الجلد بأمان وصولا إلى الأنسجة حيث تقع الأوعية الدموية. ومن ثم الحصول على صورة واضحة ومميزة من أوردة المريض الموجودة في منطقة منتصف باطن الذراع.

ويتعلق جزء تصميم النظام، بالتصميم الميكانيكي للذراع الآلي يتحرك ب 6 درجات حرة الحركة(6DOF) من خلال دراسة حركته واستخراج المعادلات الحسابية اللازمة والتحكم فيها؛ لتأمين قدرة الروبوت على الانتقال إلى نقطة الحقن المطلوبة

LIST OF ABBREVIATIONS

** The borderlines of this table should be removed at the end.

NOTATIONS

$\overrightarrow{M_{iJ}}$	The moment generated by the force $\overrightarrow{f_i}$ about the axis $\overrightarrow{a_j}$	[Nm]
$\overrightarrow{f_i}$	The gravitational force acting on body i at its gravitational center	[N]
$\overrightarrow{a_j}$	A unit vector along the j^{th} tip-over axis	
$\overrightarrow{r_i}$	A position vector pointing from any point on the tip-over axis to the line of action of the force.	[m]

** The borderlines of this table should be removed at the end.

CHAPTER 1: INTRODUCTION

Blood is a vital tool used by medical professionals for both monitoring patients' health and diagnosing various diseases. Its accessibility makes it an extremely effective method for quickly assessing a patient's overall health status and identifying any potential issues. In addition, blood is often used to deliver essential substances to the body in order to maintain good health or treat an illness. As a result, blood is considered one of the fastest and most effective ways to provide the body with the necessary nutrients and substances.

1.1 Background

One of the most often invasive procedures done in hospitals is venipuncture, which involves inserting a needle into the peripheral veins to obtain a sample of venous blood. [1], followed by cannulation, which involves the process of inserting an intravenous cannula (IV) into a peripheral vein either in hand or forearm, in order to admit fluids, medications, Parenteral nutrition...etc. [2]. A study shows that over one billion venipuncture processes are performed annually as a prerequisite for most diagnostic tests, and about 90% of hospitalized patients receive treatment through intravenous cannulation [3]. Figure 1.1 and Figure 1.2 shows the venipuncture and cannulation process respectively

Figure 1.1: Venipuncture process [4]

Figure 1.2: Cannulation process [5]

A successful Phlebotomy process mainly depends on the capability of medical staff to easily and quickly locate the suitable vein, along with the procedures that the medical personnel adhere to while drawing blood and injecting a needle. Therefore, medical professionals tend to use several techniques to facilitate the visualization of veins. The most commonly used methods are: Firstly, Venous tourniquets, where the nurse tight a strap around the patient's limb to partially to apply pressure to it causing the blood to temporarily pool in the vein; hence the veins become more distinguished [1]. Secondly tapping the site technique, where the nurse taps gently on the injection site with his fingers to make the veins more prominent while holding the skin taut with his non-dominant hand [6]. The previous mentioned techniques were put into practice in Reshmawi Laboratory to examine the challenges medical personnel encounter while executing venipuncture procedures.

1.2 Problem Statement

Doctors use several techniques to enhance and facilitate venipuncture and cannulation processes; however, the process of injecting a needle, either for blood drawing or for supplying minerals and medication, is still considered challenging for many clinicians to successfully executed on the first try due to multiple reasons presented by: skin pigmentation, obesity, age, poor vein quality, and dehydration [7]. This unsuccessful attempt in localizing the suitable vein is usually followed by several tries in needle injection, which can cause pain, discomfort, anxiety, swelling in the skin, hematoma, bleeding, infection, and even frustration for the clinician himself [6].

One of the major reasons that increase the complexity of venipuncture process is the limitation and effect of used techniques to prominent veins, on the collected blood samples, especially the tourniquet, which is commonly used nowadays in hospitals. Several researches are made to study the effect of tourniquet technique on the blood sample, which prove that the use of a tourniquet has a significant effect on the measured hemorheological parameters (the physical properties of the blood) in the blood sample [8]. These parameters are important in doctors' diagnoses since they are used as biomarkers of atherosclerosis and thrombosis. In addition, the tourniquet technique can't be used for the calcium test, because blood must be drawn from a vein that is able to flow freely because venous stasis might cause calcium levels to be artificially elevated (also based on medical staff interview at Reshmawi Laboratory).

1.3 Scope

This project seeks to develop a vein-detecting and injection system capable of successfully injecting the needle in a suitable vein selected by the medical staff, on the first try, despite the patient's age, health situation, and skin pigmentation. The system will initially capture a real-time photoaged of the patient's arm veins, then process and project it on the computer so the nurse can select the suitable vein for injection. Based on the obtained coordinate for the selected point, the robot will

precisely move to the point, then insert the needle at the center of the vessel to collect blood samples. The project aims to increase the success rate of first-stick, as well as to reduce the complexity accompanied by the venipuncture process, thus making it a faster and less painful experience. The project seeks to collect data from hospitals and the Faculty of Pharmacy, Nursing, and Health Professions investigating the issues they face while performing Intravenous injections and sampling blood, especially in the anti-cubital fossa and dorsal hand regions, then exploit this information to build a full system capable of visualizing the veins and safely inject the needle, that can hopefully be used later on in hospitals and clinical laboratories.

The project vein detector will have four main goals:

- Test the capability of using NIR Light technology to obtain a clear image of veins despite the different physical appearances of the patient and without the need for a tourniquet.
- Perform the required image processing techniques on the captured photo to obtain a clear image of the patient veins, free of noises and background surroundings
- Design and build a prototype of Detective Vein robot that can move with precision to the selected point of injection (without performing injection)
- Inquire some medical experts about the quality of the processed image and how accurately the device reflects the locations of the veins.

CHAPTER 2: LITERATURE REVIEW

The last century witnessed a significant increase in the prevalence of chronic diseases worldwide, including the rising incidence of cardiovascular diseases and diabetes, which required early detection (reference Reshmawi). These chronic diseases are diagnosed with the help of the venipuncture process, where a blood sample is collected and tested. Even though the venipuncture process has emerged as a vital tool in the hands of medical professionals for diagnosing various diseases, its complexity and low rate of first trial success, makes it a painful experience for the patient, as well as a stressful process for the nurse. In order to overcome these issues, a secure technique must be identified for mapping the blood vessels and inject needle in the patient's suitable vein. Subsequently, the dependability of the developed device must be examined to ensure that it can be easily and safely used in needle injection. The literature review will focus on two main aspects: Near-infrared (NIR) Light characteristic and its ability to reach the arm blood vessels. Additionally, examine several examples of case studies that are associated with concepts for vein finder devices that have been created.

2.1 Study the characteristics of NIR Light

Human skin consists of three main layers: The Epidermis, the Dermis, and the Subcutaneous Tissue (also known as the Hypodermis layer), Figure 2.1 shows the different layers of human skin. Subcutaneous tissue is the bottom layer of the skin and where blood vessels lie [9]. For the developed device to detect the veins, NIR

light technology should be able to penetrate skin layers up to the subcutaneous tissue to reach the veins.

2022 Finlayson et al., [10] presents the thickness of each skin layer coupled with the average depth of the layers taking the skin surface as a reference point. Table.2.1 shows the thickness of skin layer

Table 1: Thickness of skin layers and the average depth of each layer refer to the skin surface [10]

Parameter	Layer Name	Layer Thickness (mm)	Average depth (mm)
1	Stratum Corneum	0.02	0.02
2	Epidermis	0.064 (at thickest point)	0.084
3	Melanin Layer	0.01	0.094
4	Basal Layer	0.01	0.104
5	Dermis	1.996 (at thickest point)	2.1
6	Subcutaneous Fat	3.004	5.0

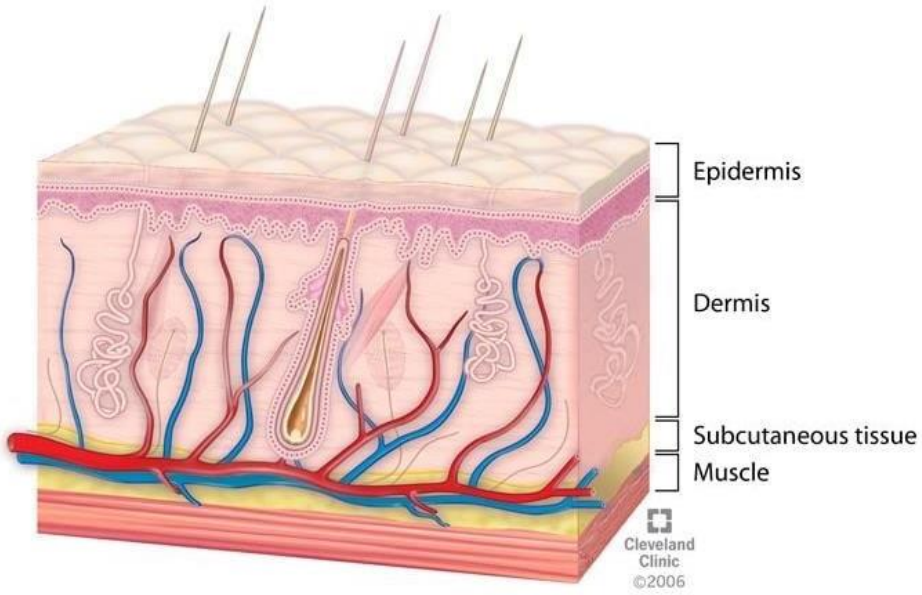


Figure 2.1: Human skin layers [11]

This section will discuss the NIR light in terms of its properties and ability to penetrate deep down into the skin Layers.

2.1.1 Near Infrared Light (NIR)

Near-infrared Light (NIR) is a type of light that lies in the invisible wavelength spectrum to the human-naked eye. The NIR wavelength extends from 700 to 2500 nanometers, as shown in Figure 2.2, which presents the wavelength region for NIR Light [12].

NIR Light is noninvasive and safe light commonly used in light therapy due to its natural healing benefits. 2016 Sakudo et al., [13] presents various applications, where NIR spectroscopy is applied in the health and medical field including NIR light imaging, monitoring the oxygenation state of tissues, analyzing brain function, and study a wide range of diverse conditions such as type 1 and 2 diabetes mellitus, aging, breast cancer, Alzheimer's disease, etc.

For NIR technology to be useful in localizing veins in patient's arm, it should be able to penetrate down the skin up to Subcutaneous tissues (where blood vessels lie) which is around 5 mm depth from the surface of the arm skin. Based on 2017 Ash et al., [14], and 2017 Chandra et al., [15], the NIR Light with wavelength between 700-900nm can penetrate the skin up to 5mm or more. Figure 2.3 shows the penetrate depth of difference Types of light including NIR light.

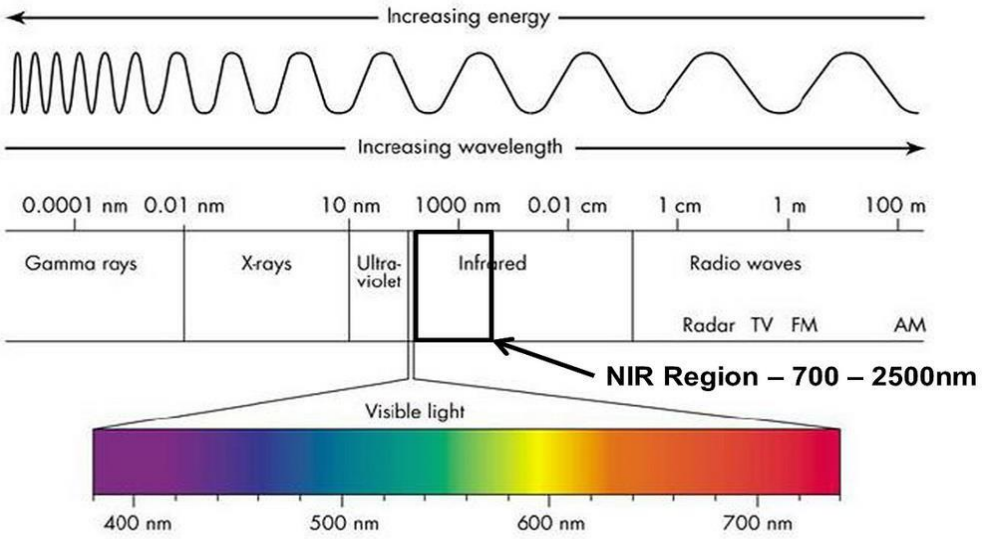


Figure 2.2: Wavelength range for NIR Light [12]

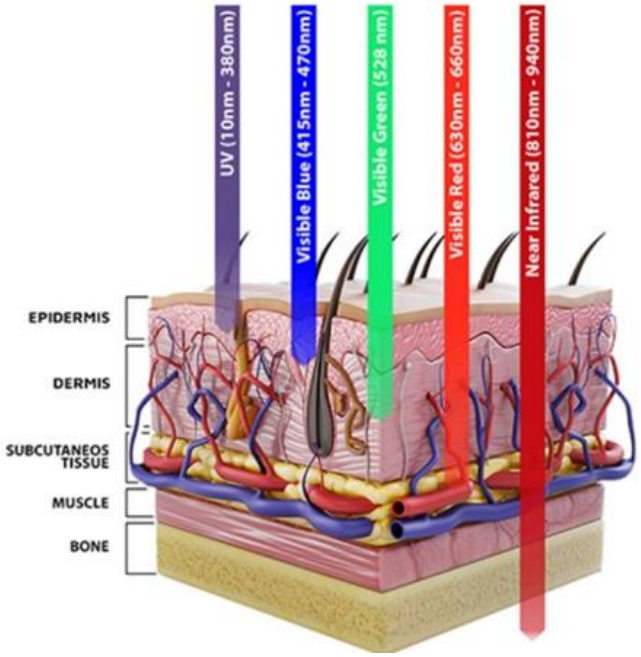


Figure 2.3: Penetration of NIR Light depth through human skin

In addition to the NIR penetration property, NIR light of wavelength 750-950nm is strongly absorptive by hemoglobin and deoxyhemoglobin within the blood vessels while being reflected by the surrounding tissues [7]. Thus, with the help of an infrared camera, the blood vessel can be detected since they will appear in darker shade (black line) compared to the surroundings.

2.2 Case Studies

Vein visualization technologies have captured the interest of many scientists and developers in the last two decades with the mindset of inventing a new technology capable of reducing the complexity of vein localization for the needle injection process. Several studies were performed to develop a vein finder device

2.2.1 Case Study 1

2021 Francisco et al., [16] shows the potential of using NIR technology by developing a device capable of detecting and localizing veins in patients' arms despite their physical differences (parameters: gender, age, mass index (BMI), and skin tone). The developed device depends on the usage of the following components:

- A set of three light-emitting diodes of 960nm wavelength placed two centimeters apart and powered by a 1.5 Volts DC Battery. This set of lights is used to produce a NIR light with a suitable wavelength capable of penetrating the skin to reach the veins.
- A Complementary metal oxide semiconductor (CMOS) camera with 1920X1080 UXCA that is set at the center of the Light-emitting diode (LEDs) with a distance of 3.5 cm, to capture an image of the patient's arm that is subjected to the NIR light.
- An Infrared (IR) filter to block other wavelengths outside the NIR light spectrum.
- The captured vein image was processed using open-source software (Vds Vein display system 4.0) with a laptop acting as the image processing unit (IPU).

The capability of this developed device has been tested on 242 human subjects with different physical appearances (gender, age, skin color, and mass index). This test has mainly focused on the efficiency of the device in giving clear images that show the location of veins in two different sites, the arm, and the dorsal hand. Unfortunately, this developed device has two main weaknesses: The design of the device has been made to be portable (the instability of the device during the process of obtaining the vein's image) and the device isn't capable of reflecting the method of choosing the best vein for venipuncture. Despite these lacks, the device has given

a future opportunity in facilitating the venipuncture process in comparison to the traditional methods that mainly depend on the naked eye.

2.2.2 Case Study 2

2017 Dhakshayani et al., [17], a veins finder device circuit was designed as follows: 24 infrared lights (IR LEDs) with 740,765,770 nm wavelengths where the experimental testing shows these rays exhibit high absorption for deoxyhemoglobin in veins used (six lights for each length), powered by a 12-volt adapter with light dependent resistor (LDR) that its resistance and voltage decrease with high intensity of IR LEDs. These LEDs are arranged circularly to give the best diffusion and illumination on a Printed Circuit Board (PCB) board. Then a webcam was used to show the effectiveness of IR rays this camera contains a CMOS sensor instead of Display Data Channel (DDC) Chip due to it has a quicker reading, uses less power, has better noise immunity, and is highly sensitive to both visible and IR light. Also, the hot mirror filter was replaced by IR film and a Kodak Wratten 87 IR filter to allow just IR lights to access and block all visible light, and a Liquid Crystal Display (LCD) that is used for display purposes this circuit was connected to Raspberry pi. To stream a live video a webcam motion package was used. The test was on the elderly and pediatric people, fat people, and dark persons cephalic in the dorsal hand and antecubital veins in the forearm. The visibility of veins can be affected by the amount of pigment in the skin. In a dark-skinned individual, the antecubital vein in the forearm and the cephalic vein in the back of the hand can be depicted. The amount of light that penetrates the skin and reaches the veins can vary due to differences in skin pigmentation, however, by using a specific multispectral wavelength that has a low absorption rate for melanin and an optimal infrared detector, it is possible to easily visualize the veins regardless of skin pigment. In an individual with a high body mass index, the antecubital vein in the forearm and the cephalic vein in the back of the hand may be difficult to visualize due to the presence of subcutaneous fat under the skin. However, the proposed prototype was able to locate these veins using an optimal detector and a multispectral wavelength that has deep tissue penetration, even in the presence of fat. In elderly individuals, it can be difficult to locate veins in the antecubital region of the forearm and the cephalic

vein in the back of the hand due to the thinning and fragility of the veins as they lose elasticity with age. However, the proposed method was successful in locating veins with less difficulty. In pediatric patients, it can be difficult to locate veins in the antecubital and cephalic regions due to the smaller size of the veins, the presence of subcutaneous fat, and the tendency for vasoconstriction. However, the proposed prototype was successful in identifying these veins using a multispectral wavelength and an optimal detector that was selected to optimize peak absorption and deep tissue penetration at these locations. The prototype was tested on four groups of subjects: 25 individuals with dark skin, 25 individuals with high body mass indexes, 25 pediatric subjects between 1.5 and 2 years old, and 25 elderly subjects between 55 and 60 years old. The success rate was 95% for the first group, 90% for the second and third groups, and 85% for the fourth group. These initial observations suggest that the prototype is effective in helping phlebotomists and nurses access veins in a variety of individuals, including those with dark skin, high body mass indexes, and advanced age, with minimal effort and without the use of a tourniquet.

2.2.3 Case Study 3

2016 Wadhwani et al., [18] mentioned that the near-infrared vein imaging technique is not affected by the skin color or pigmentation of the individual, and tattoos do not interfere with the imaging process. In addition, this article mentioned the way that the image of veins can be appeared by using some techniques such as Image Acquisition and Preprocessing to enhance the imaging quality in order to make it easier to detect vein patterns during the segmentation process, Segmentation and Post-processing to distinguish the vein pattern from the surrounding background and LabVIEW Implementation with a special block called NI Vision Acquisition Software to obtain real-time images and displaying, acquiring, monitoring images, & logging from a multitude of camera types Gaussian and median filters were used to remedy the effect of this noise. The conditions under which the snapshots were taken were controlled and had good lighting, but it may be necessary to make additional adjustments or use software to calibrate the images for use in other environments or for other purposes.

2.2.4 Case Study 4

2013 A. Chen et al., [19] states that an array of twelve light-emitting diodes (LED) providing reflectance illumination with two CMOS cameras with increased NIR sensitivity to capture images in real-time to detect veins of the human hand. Where the human hand is placed on a disposable paper sheet which serves as a sterile barrier between the device and the patient's skin.

A number of standard optical techniques are applied such as, a 940 nm band-pass filter positioned in front of each camera to eliminate ambient light with holographic diffusion filters placed over the LEDs to increase optical isotropy and to allow forearm backscattering to be modeled as a Lambertian phenomenon and NIR polarizers are positioned orthogonally over the camera and LEDs to reduce specular reflection from superficial skin layers.

CHAPTER 3: METHODOLOGY

Generally, the developed device should be composed of two subsystems; the first is responsible for processing and analyzing the anatomy of the patient's venous structure and determining the suitable vein for injection. The second subsystem is responsible for moving the robot smoothly and precisely to the predetermined injection point. The first subsystem should communicate with the second subsystem as to provide it with the required injection point coordinate (x, y, z), as well as the orientation of the patient's arm so the robot can move to the goal position.

In order for the system to fulfill its intended purpose of smoothly and safely injecting the needle for collecting blood samples on the first try, each subsystem should achieve several requirements:

The first subsystem should take full advantage of the invasive NIR technology by selecting the optimal number of LEDs, their arrangements, and the suitable surrounding conditions; to obtain a clear image that truly reflects the vein's locations in the patient's arm in real-time. The processor should be small and compact but powerful enough to perform image processing quickly and with high quality.

Meanwhile, the second subsystem, which is the robot arm, should be flexible, easy to operate, precise, and capable of reaching the selected point with a slope of 15° from the patient's arm. It should be made of a material that satisfies both; the robot's structural requirements and the safety to use in medical applications. Figure (3.1) shows the general working concept of the developed system

This chapter will discuss the criteria for different components of the system, then the working concept of the device.

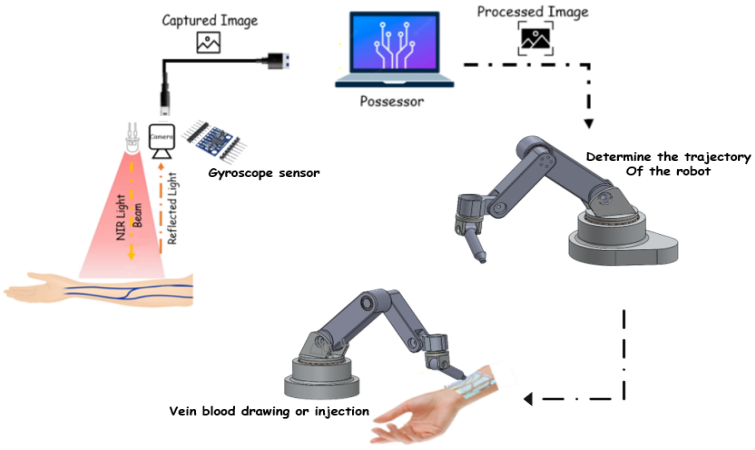


Figure 3.1: Working concept of Detective Vein robot system

3.1 Vision machine (Camera)

Camera selection is a key element in any image processing process, where camera type determines the quality and the clarity of the captured image, which will immediately affect the complexity of image processing. Normally, camera selection is done according to several criteria, such as resolution, size, color or monochrome image type, speed transfer, etc. And based on the application, those criteria are determined in order to obtain the best results.

Since the project is concerned with locating a suitable vein for injection using the NIR technology, a monochrome camera with a high resolution, capable of capturing the NIR light is required. A monochrome image or monochrome photography is a type of image that displays a single-color image with different shades, which is usually various shades of neutral gray with no other colors included. Due to the property of color reduction into a single color of different shades, the background images appear less prominent than the centralized topic of the picture. The monochrome cameras are more accurate than color cameras in dimensional-type measurements using edges which is an essential requirement in determining the vein's edges and locations.

See3CAM_CU55M camera (the latest edition of the e-con systems ‘family of USB3.1 Gen1 UVC cameras) is a type of camera that satisfy most of the project camera requirements. It is a 5MP monochrome camera capable of delivering

monochrome images with exceptional quality and low noise; This is possible due to its 2.2m X 2.2m pixel BSI technology [20]. Figure 3.2, represents the appearance of See3CAM_CU55M camera.

Figure 3.1: SeeCAM_CU55 camera [20]

According to Mr. Ashot Bah, the president of e-con systems Inc, and the ability of the camera to present a monochrome format image, the See3CAM_CU55 can generate an excellent image; in both visible and Near Infra-red regions, making it suitable for different applications, including iris recognition, NIR imaging, driver monitoring, and digital microscopy.

The See3CAM_CU55 camera is lightweight, versatile, and portable in design. It is mainly composed of the following parts:

- Two-board solution of size 30 mm x 30 mm
- 5.0MP Monochrome CMOS image sensor.
- Standard M12 lens holder for use with customized optics or lenses for various applications
- USB 3.1 GEN 1 device with Type-C reversible interface connector
- Plug-and Play setup (UVC compliant) for windows 8.1/10 and Linux
- 10-pin GPIO header for standard and custom operations. GPIO pins are accessible from the PC host application

See3CAMCU55 uses CMOS sensor to create images and videos. Complementary metal oxide semiconductor (CMOS) sensors are electronic chips that rely on the principle of photoelectric effect to convert photons into voltage directly within the pixels. Table 3.1.2, presents the specifications of the CMOS sensor used in See3CAMCU55 camera.

Table 3.1.2: CMOS Image sensor Specification used in SeeCAMCU55 camera [20]

Sensor Specification	
Type/Optical Size	1/2.5” Optical format CMOS Image sensor
Resolution	5MP
Sensor type	Monochrome – Y12, Y8 – CMOS Rolling Shutter sensor
Pixel Size	2.2μm x 2.2 μm
Sensor Active Area	2592 (H) x 1944 (V)
Responsivity	36 ke-/lux*sec
SNR	40dB
Dynamic Range	*3.3 dB*

The operating conditions, camera supplied voltage specifications, and the camera dimensions are presented in the Appendix SeeCAMCU55.

3.2 NIR LEDs Arrangement

To determine the optimal status that gives the best results to show the veins, the NIR LEDs were arranged in many ways and shapes trying to make NIR light absorbed through the patient’s veins as much as possible. In addition to site the Raspberry PI NIR Camera in a suitable position to capture a perfect image of the veins.

The first notation is the number of NIR LEDs that affect mainly the visibility of the veins. So that more NIR LEDs more visibility of the veins is seen through the

camera, the trails started with 20 LEDs ...a. 30 LEDs … 60 LEDs until reached 99 LEDs.

The second notation is the arrangement of these LEDs, many attempts of designing are applied to get the principal effect of the NIR LEDs on the veins as shown in Figure (3.7)

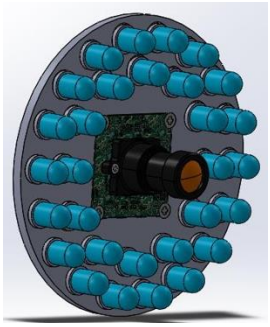


Figure 3.3: 1st Trail

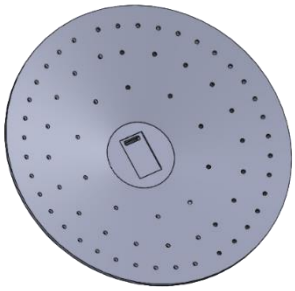


Figure 3.4: 2nd Trail

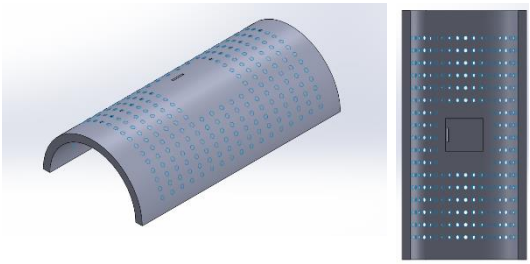


Figure 3.5: 3rd Trail

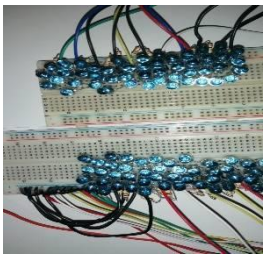


Figure 3.6: 4th Trail

Figure (3.3) shows 30 NIR LEDs are arranged in circular form, this arrangement shows the veins through the camera with less than a good result due to some of the LEDs light did not penetrate through the vein based on the shape of the arm. In Figure (3.4) design the shape of the LEDs base helps to focus the NIR light on the patient's arm but the wiring of the LEDs was very bad due to the ends of the wires touching each other. The design of Figure (3.5) deployed the LEDs in arc shapes that increase the focusing of the NIR light-emitting on the arm, but it will impede the movement of the robot. At last, the arrangement of LEDs that showed in Figure (3.6) was adopted which gives the best result of the vein as shown in Figure (3.7), here 99 NIR LEDs were added in four rows with two columns distributed on two

Figure 3.7: Veins detection using the arrangement of 4th Trail

breadboards in make the wiring easier in sync gives high emitting and penetration in the veins.

The site of the camera for all previous designs and LEDs arrangements has been at the center which is the finest site that the camera can be, so it can capture as much as a perfect image of the veins.

3.3 Robot Design, Material and Control

The Detective Vein robot arm is responsible of injecting the needle at the injection point selected by the medical staff, where the robot will move from its current position to the desired position in a smooth, precise movement. Since the needle injecting process is a very delicate process, and require interaction with human, the designed robot should be flexible, precise, and made of a material that is safe to use in medical applications.

To fulfill the system requirements; the developed robot should be a collaborative robot of 6 Degrees of Freedom (DOF). Collaborative robots are types of robots designed for direct human-robot interaction, where they will completely stop if they collide with anything other than the work item. The developed robot will have 6 DOF to ensure flexibility in reaching the injecting point with the suitable injection angle. Figure (3.8) shows the Detective Vein robot design, where the robot consists

of 6 simple revolute joints along with seven main parts. The end effect part of the robot will contain the needle inside, as well as a linear mechanism to insert the needle once the robot reaches the injection joint. However, this part of the system will not be covered in this project.

As mentioned before, the robot should be made of a material that balance between the robot requirements (rigid material) and the medical application approval. There are several types of materials that can be safely used in medical applications such as stainless steel, copper, aluminum, as well as plastic which is lightweight and biocompatible. Aluminum material is the selected material for the real implementation of the robot, since it is lightweight, durability, strength, has excellent corrosion resistance, can be easily formed, and cost less when compared to stainless steel. However, since the project only aims to build a prototype model of the system, to test the system concept and design, the robot will be made of plastic (PLA material in specific, since it be 3D printed).

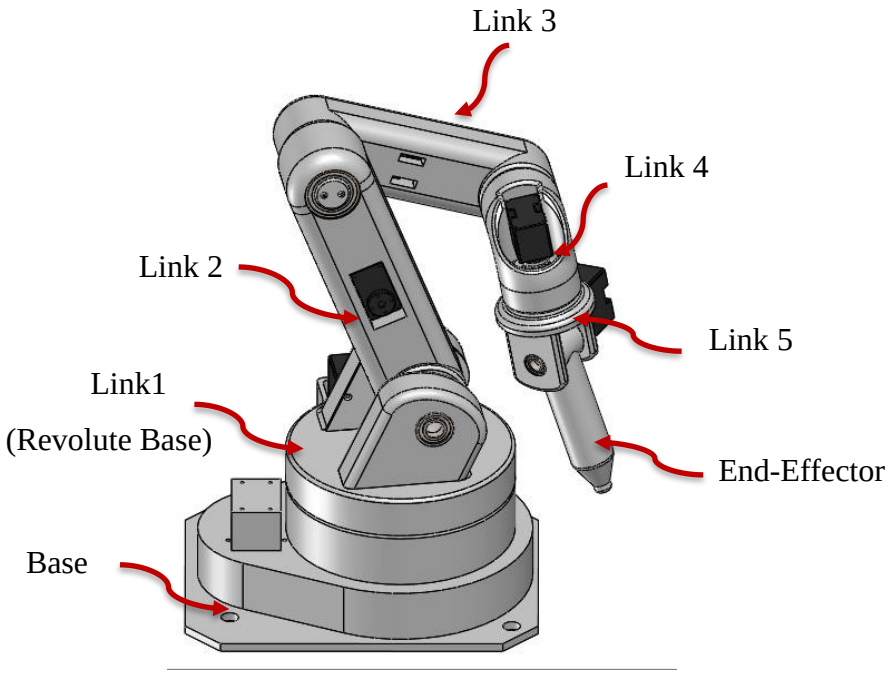


Figure 3.8: Detective Vein Robot

The system will contain a type of controller, either PD or PID type; to ensure that the robot end effector reaches the injection point within a reasonable time (maximum 5 seconds) and a zero steady-state error. PD or PID controllers are selected since they are the most common controllers used in robots and in particular the PD controllers.

CHAPTER 4: ROBOT KINEMATICS

Kinematics concerns the study of Detective-Vein robot motion without considering the forces and torques that cause it. Generally, there are two different spaces used in the kinematics modeling of robot manipulators: the Cartesian and Quaternion space. The developed study uses the Cartesian space for studying the Detective-Vein robot's forward, inverse and velocity Kinematics that are necessary in determining the needle tip location and orientation from joints parameters, calculating the joint angles needed to place the needle tip at the desired position, and the needle tip velocity relating to the joints' velocities.

4.1 Forward Kinematics

As mentioned in the previous chapter, the Detective-Vein robot consists of serial links, which are attached to each other via a set of six simple revolute joints, each with a single degree of freedom (DOF). The process of determining the pose (position and orientation) of the robot needle tip (end effector) in terms of the joint variables is known as the forward kinematics problem. Since all robot joints are revolute, the joint variables are the angles between the links.

To be able to perform the forward kinematics analysis and describe how the robot's link is oriented with respect to others or patient's arm, a coordinate frame should be attached to each robot link ($x_i\ y_i\ z_i, o_i$), where it experiences the same motion as the link when the robot joints are actuated. Furthermore, the frame ($x_0\ y_0\ z_0, o_0$) attached to the Detective-vein robot base is called the reference or global frame, where this coordinate frame is fixed at a known position in space and doesn't move over time. Figure 1.1, shows the assigned coordinate for each joint of the Detective-Vein robot as well as the reference and end-effector frames.

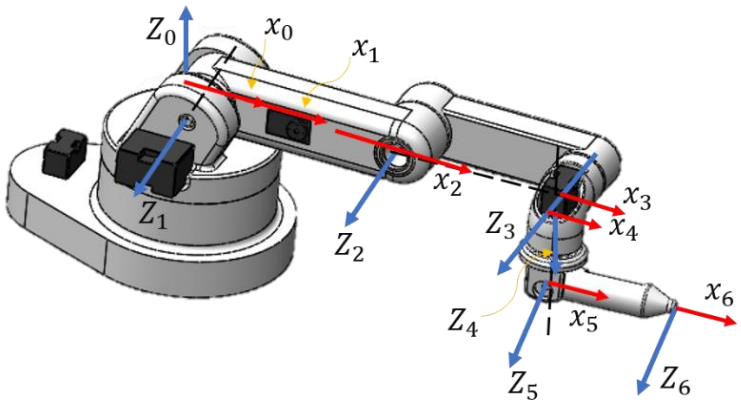


Figure 4.0: Detective Vein robot assigned frames

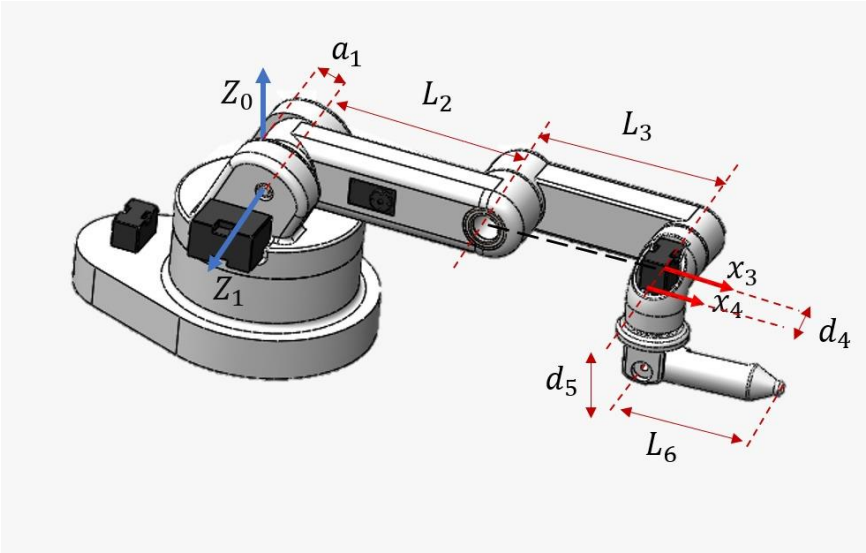


Figure 4.2: Dimensions used in DH parameter Table

Table 4.1: DH Parameter for Detective Vein robot based on the determined coordinate frames

i (Joint number)	α_i	a_i	d_i	θ_i
1	$\pi/2$	a_1	0	θ_1
2	0	L_2	0	θ_2
3	0	L_3	0	θ_3
4	$\pi/2$	0	d_4	θ_4

5	$-\pi/2$	0	d_5	θ_5
6	0	L_6	0	θ_6

The Detective-vein robot forward kinematics equations are obtained using the Denavit-Hartenberg method (DH). Denavit-Hartenberg is a systematic method that uses a combination of four parameters to model robot kinematics. These parameters are the link length (a_i), link twist (α_i), link offset d_i, and joint angle θ_i. The fundamental description for each parameter is as follows:

- a_i : The common normal distance between the axes Z_i and Z_{i+1}
- d_i = The distance between the axes X_{i-1} and X_i along Z_i. It is considered variable if joint i is prismatic.
- α_i = The angle between Z_i and Z_{i+1} measured about X_i.
- θ_i = The angle between X_{i-1} and X_i measured about Z_i. It is considered a variable if joint i is revolute.

Table 4.1 contain the DH parameters used in solving Detective-Vein robot's forward kinematics, according to the assigned coordinate frames shown in Figure 4.1. The general homogeneous transform matrix of the DH methodology is given by Equation (4.1). The transformation matrix is a 4×4 matrix used for describing the orientation and position of robot components by combining the 3×3 rotation matrix with the 3×1 displacement vector into a single matrix.

The solution of forward kinematics is obtained by multiplying the six calculated homogeneous transform matrices, as shown in Equation (1.1). The resultant homogeneous transformation matrix (${}^0_6T$) describes the pose of the robot needle tip (end effector) with respect to its base reference.

$$ {}^0_6T = {}^0_1T\ {}^1_2T\ {}^2_3T\ {}^3_4T\ {}^4_5T\ {}^5_6T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1) $$

Where:

$$n_x = C\theta\_6\,[(S\theta\_1 * S\theta\_5\,) + (C\theta\_1 * C\theta\_5 * C\theta\_234\,)] - (C\theta\_1 * S\theta\_6 * S\theta\_234\,) \tag{4.2}$$

$$n_y = -C\theta_6[(C\theta_1 * S\theta_5) - (S\theta_1 * C\theta_5 * C\theta_{234})] - (S\theta_1 * S\theta_6 * S\theta_{234}) \tag{4.3}$$

$$n_z = C\theta_{234} * S\theta_6 + C\theta_5 * C\theta_6 * S\theta_{234} \tag{4.4}$$

$$n_z = C\theta_{234} * S\theta_6 + C\theta_5 * C\theta_6 * S\theta_{234} \tag{4.5}$$

$$o_x = -S\theta_6[(S\theta_1 * S\theta_5) + (C\theta_1 * C\theta_5 * C\theta_{234})] - (C\theta_1 * C\theta_6 * S\theta_{234}) \tag{4.6(}$$

$$o_y = S\theta_6[(C\theta_1 * S\theta_5) - (S\theta_1 * C\theta_5 * C\theta_{234})] - (S\theta_1 * C\theta_6 * S\theta_{234}) \tag{4.7(}$$

$$o_z = C\theta_{234} * C\theta_6 - C\theta_5 * s\theta_6 * S\theta_{234} \tag{4.8}$$

$$a_x = C\theta_5 * S\theta_1 - (C\theta_1 * C\theta_{234} * S\theta_5) \tag{4.9}$$

$$a_y = -C\theta_5 * C\theta_1 - (S\theta_1 * C\theta_{234} * S\theta_5) \tag{4.10}$$

$$a_z = S\theta_{234} * S\theta_5 \tag{4.11}$$

$$p_x = a_1 * C\theta_1 + d_4 * S\theta_1 + L_2 * C\theta_1 * C\theta_2 + L_6 * C\theta_6 * [S\theta_1 * S\theta_5 + C\theta_1 * C\theta_5 * C\theta_{234}] + d_4 * C\theta_1 * S\theta_{234} + L_3 * C\theta_1 * C\theta_2 * C\theta_3 - L_3 * C\theta_1 * S\theta_2 * S\theta_3 - L_6 * C\theta_1 * S\theta_6 * S\theta_{234} \tag{4.12}$$

$$p_y = a_1 * S\theta_1 + d_4 * C\theta_1 + L_2 * S\theta_1 * C\theta_2 - L_6 * C\theta_6 * [C\theta_1 * S\theta_5 - S\theta_1 * C\theta_5 * C\theta_{234}] + d_4 * S\theta_1 * S\theta_{234} - L_3 * S\theta_1 * S\theta_2 * S\theta_3 + L_3 * S\theta_1 * C\theta_2 * C\theta_3 - L_6 * S\theta_1 * S\theta_6 * S\theta_{234} \quad (4.13)$$

$$p_z = L_3 * S\theta_{23} + L_2 * S\theta_2 - d_5 * C\theta_{234} + L_6 * C\theta_{234} * S\theta_6 + \frac{L_6 * C(\theta_5 - \theta_6) * S\theta_{234}}{2} + \frac{L_6 * C(\theta_5 + \theta_6) * S\theta_{234}}{2} \quad (4.14)$$

Where

- S is a shorten for sin
- C is a shorten for cos
- $S\theta_{234}$ represents $sin(\theta_2 + \theta_3 + \theta_4)$

To Validate the correctness of the obtained forward kinematics equation and DH table, a simple test is used, where all joint parameters were set to zeros, as shown in Figure 4.2. In order for the model to be correct, the position value in the x-axis direction should be equal to the summation of a1, link 2, link 3, and link 6 lengths. Meanwhile, the position in the y-axis should be equal $-d_4$, and finally position value in the z-axis should be equal to negative value of the distance between joint 4 and joint 5 (in other words, the value of d5).

By using MATLAB software to perform the calculation, the result of the translation vector when all Detective Vein robot's joint parameter is set to zero is:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 0.377 \\ -0.013 \\ -0.065 \end{bmatrix}$$

4.3 Inverse Kinematics

Forward kinematics can determine the end effector location in the cartesian space based on the provided joint rotation angles. However, in real-world, the robot operates the opposite; the pose of the end effector is known, yet the controller needs to know the joint angles to drive the robot end effector to the desired position and orientation. Accordingly, the inverse kinematics is used to solve this problem.

Inverse kinematics is essentially the reverse operation of forward kinematics; it refers to the use of kinematics equations to find a configuration (or configurations) so that the robot needle tip (end effect) can reach the desired position that is predetermined by the nurse for needle injection, with the suitable injection angle. In other words, the inverse kinematics problem finds the set of joint angles $q = \{q_i\}$, where $q_i = (\theta_1^i, \theta_2^i, \theta_3^i, \theta_4^i, \theta_5^i, \theta_6^i)$ that satisfies

$$
\begin{aligned}
{}_6^0T\left(\theta_1^i, \theta_2^i, \theta_3^i, \theta_4^i, \theta_5^i, \theta_6^i\right) &= {}_1^0T(\theta_1)\,{}_2^1T(\theta_2)\,{}_3^2T(\theta_3)\,{}_4^3T(\theta_4)\,{}_5^4T(\theta_5)\,{}_6^5T(\theta_6) \\
&= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{4.15}
$$

Where:

${}_6^0T$: is the transformation matrix that describes the desired position and orientation of the end effector based to the reference frame.

4.3.1 Solving for θ_1

To solve for the first joint angle (θ_1) Equation (2.1) is used, which represents the transformation matrix from frame 5 to frame 1, [22]

$$
{}_5^1T = {}_1^0T^{-1}\,{}_6^0T\,{}_6^5T^{-1} = {}_2^1T\,{}_3^2T\,{}_4^3T\,{}_5^4T \tag{4.16}
$$

By comparing the matrix element of third row and fourth column from the left side of Equation (4.16) with the third row and fourth column element from the right side, the following relation is obtained

$$p_x \sin\theta_1 - p_y \cos\theta_1 + L_6 * n_y * \cos\theta_1 - L_6 * n_y * \sin\theta_1 = d_4 \quad (4.17)$$

By rearranging Equation (4.17), Equation (4.18) is obtained

$$m\cos\theta_1 - n\sin\theta_1 = d_4 \quad (4.18)$$

Where:

$m = L_6 * n_y - p_y$

$n = L_6 * n_x - p_x$

To solve an equation of this form, trigonometric substitutions are made:

$$n = \rho * cos\varphi \quad (4.19)$$

$$m = \rho * \sin\varphi \quad (4.20)$$

Thus;

$$\rho = \sqrt{n^2 + m^2}$$

$$\varphi = Atan2(m, n)$$

By substituting Equations () and () in Equation

$$\cos\theta_1 * \sin\varphi - \sin\theta_1 * \cos\varphi = \frac{d_4}{\rho} \quad (4.21)$$

From the difference of angles formula

$$\sin(\varphi - \theta_1) = \frac{d_4}{\rho} \quad (4.22)$$

Hence,

$$\cos(\varphi - \theta_1) = \pm\sqrt{1 - \sin(\varphi - \theta_1)} = \pm\sqrt{1 - \left(\frac{d_4}{\rho}\right)^2} \tag{4.23}$$

Thus;

$$\varphi - \theta_1 = Atan2(\frac{d_4}{\rho}, \pm\sqrt{1 - \left(\frac{d_4}{\rho}\right)^2}) \tag{4.24}$$

Finally, the solution for the first joint angle (θ_1) is given by Equation:

$$\theta_1 = atan2(m, n) - antan2(d_4, \pm\sqrt{m^2 + n^2 - d_4^2}) \tag{4.25}$$

There are two possible solutions for θ_1, corresponding to the minus and positive sign; in other words, the shoulder configuration could be either to the 'right' or to the 'left'.

4.3.2 Solving for θ_5

By obtaining the value of the first joint angle (θ_1), the transformation matrix from frame 1 to frame 6 can now be used to solve for the fifth joint angle (θ_5), [22].

$${}^1_6T = {}^0_1T^{-1}\,{}^0_6T = {}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T{}^5_6T \tag{4.26}$$

By comparing the matrix element (3 × 3) from the right side of Equation (4.26) with the element (3 × 3) of the left side, the following relation is obtained:

$$a_x \sin\theta_1 - a_y \cos\theta_1 = \cos\theta_5 \tag{4.27}$$

From basic Trigonometric identities:

$$\sin\theta_5 = \pm\sqrt{1 - (\cos\theta_5)^2} \tag{4.28}$$

Hence,

$$\theta_5 = atan2(\sin\theta_5, \cos\theta_5) \tag{4.23}$$

Once again, there are two solutions for the fifth joint angle (θ_5), which correspond to the two different wrist configurations (either it is “in” or “out”).

4.3.3 Solving for θ_6

By comparing the matrix element (3×3) from the right side of Equation (4.26) with the element (3×3) of the left side, the following relation is obtained:

4.4 Velocity Kinematics

Velocity kinematics relates the linear and angular velocities of the robot end effect (or any other point on the robot) to the joints' velocities by determining the Jacobin matrix. Jacobin matrix is one of the essential quantities in robot analysis and control; besides encoding the relation between joints' velocities, it is used in various aspects of robotic manipulation, such as planning and executing trajectories, determining the robot's singularity configurations, and deriving the robot's dynamical equation.

The relation between the velocity of end effector to the vector of joint velocities is given by Equation ():

$$V = \begin{bmatrix} \dot{x} \\ \omega \end{bmatrix} = J_g(q)\dot{q} \tag{4.40}$$

Where:

- $\dot{x}$: The Linear velocity vector in x, y, z directions.
- ω: The Angular velocity vector about x, y, z axes.
- $J_g(q)$: Geometrical Jacobian matrix of 6×n, n: number of joints.
- $\dot{q}$: Joint velocities vector of n × 1.

Since the Detective-Vein robot (DV robot) consists only of revolute joints, the Jacobian matrix can be calculate using the matrix formula shown in Equation ().

$$J_g(q) = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{4.41}$$

$$= \begin{bmatrix} Z_0 \times (O_6 - O_0) & Z_1 \times (O_6 - O_1) & Z_2 \times (O_6 - O_2) & Z_3 \times (O_6 - O_3) & Z_4 \times (O_6 - O_4) & Z_5 \times (O_6 - O_5) \\ Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{bmatrix}$$

Where:

O_i , $i = 0,1,2 \ldots$: is the joints and end-effector origins.

Z_i , $i = 0,1,2 \ldots$: The rotation of z axis from frame i to the reference frame.

The position for the origins and rotation of z-axes for each joint and the needle tip of the Detective Vein robot are presented in Figure (), where the Equation () to () presents the vector values of Detective Vein robot's origins location and Equations from () to () presents the rotation vector of the z-axes.

- Detective Vein robot's origins

$$O_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.42}$$

$$O_1 = \begin{bmatrix} a_1 * C\theta_1 \\ a_1 * S\theta_1 \\ 0 \end{bmatrix} \tag{4.43}$$

$$O_2 = \begin{bmatrix} a_1 * C\theta_1 + L_2 * C\theta_1 * C\theta_2 \\ a_1 * S\theta_1 + L_2 * S\theta_1 * C\theta_2 \\ 0 \end{bmatrix} \tag{4.44}$$

$$O_3 = \begin{bmatrix} a_1 * C\theta_1 + L_2 * C\theta_1 * C\theta_2 + L_3 * C\theta_1 * C\theta_{23} \\ a_1 * S\theta_1 + L_2 * S\theta_1 * C\theta_2 + L_3 * S\theta_1 * C\theta_{23} \\ L_2 * S\theta_2 + L_3 * C\theta_2 * S\theta_3 + L_3 * C\theta_3 * S\theta_2 \end{bmatrix} \tag{4.45}$$

$$O_4 = \begin{bmatrix} a_1 * C\theta_1 + d_4 * S\theta_1 + L2 * C\theta_1 * C\theta_2 + L_3 * C\theta_1 * C\theta_{23} \\ a_1 * S\theta_1 - d_4 * C\theta_1 + L_2 * C\theta_2 * S\theta_1 + L_3 * S\theta_1 * C\theta_{23} \\ L_3 * S\theta_{23} + L_2 * S\theta_2 \end{bmatrix} \tag{4.46}$$

$$O_5 = \begin{bmatrix} a_1 * C\theta_1 + d_4 * S\theta_1 + L2 * C\theta_1 * C\theta_2 + d_5 * C\theta_1 * S\theta_{234} + L_3 * C\theta_1 * C\theta_{23} \\ a_1 * S\theta_1 - d_4 * C\theta_1 + L_2 * C\theta_2 * S\theta_1 + +d_5 * S\theta_1 * S\theta_{234} + L_3 * S\theta_1 * C\theta_{23} \\ L_3 * S\theta_{23} + L_2 * S\theta_2 - d_5 * C\theta_{234} \end{bmatrix} \tag{4.47}$$

$$O_6 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{4.48}$$

Where:

$$p_x = a_1 * C\theta_1 + d_4 * S\theta_1 + L_2 * C\theta_1 * C\theta_2 + L_6 * C\theta_6 * [S\theta_1 * S\theta_5 + C\theta_1 * C\theta_5 * C\theta_{234}] + d_4 * C\theta_1 * S\theta_{234} + L_3 * C\theta_1 * C\theta_2 * C\theta_3 - L_3 * C\theta_1 * S\theta_2 * S\theta_3 - L_6 * C\theta_1 * S\theta_6 * S\theta_{234} \quad (4.49)$$

$$p_y = a_1 * S\theta_1 + d_4 * C\theta_1 + L_2 * S\theta_1 * C\theta_2 - L_6 * C\theta_6 * [C\theta_1 * S\theta_5 - S\theta_1 * C\theta_5 * C\theta_{234}] + d_4 * S\theta_1 * S\theta_{234} - L_3 * S\theta_1 * S\theta_2 * S\theta_3 + L_3 * S\theta_1 * C\theta_2 * C\theta_3 - L_6 * S\theta_1 * S\theta_6 * S\theta_{234} \quad (4.50)$$

$$p_z = L_3 * S\theta_{23} + L_2 * S\theta_2 - d_5 * C\theta_{234} + L_6 * C\theta_{234} * S\theta_6 + \frac{L_6 * C(\theta_5 - \theta_6) * S\theta_{234}}{2} + \frac{L_6 * C(\theta_5 + \theta_6) * S\theta_{234}}{2} \quad (4.51)$$

- The Z_{i-1} Vectors:

$$Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.52)$$

$$Z_1 = \begin{bmatrix} S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix} \quad (4.53)$$

$$Z_2 = \begin{bmatrix} S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix} \quad (4.54)$$

$$Z_3 = \begin{bmatrix} S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix} \quad (4.55)$$

$$Z_4 = \begin{bmatrix} C\theta_1 S\theta_{234} \\ S\theta_1 S\theta_{234} \\ -C\theta_{234} \end{bmatrix} \quad (4.56)$$

$$Z_5 = \begin{bmatrix} C\theta_5\ S\theta_1 - C\theta_1\ C\theta_{234}\ S\theta_5 \\ -C\theta_5\ SC - S\theta_1 C\theta_{234} S\theta_5 \\ -S\theta_5\ S\theta_{234} \end{bmatrix} \quad (4.57)$$

The MATLAB software is used to perform the cross product and obtain the final result of geometrical Jacobian, Code is attached in Appendix:

$$J = \begin{bmatrix} v_{X1} & v_{X2} & v_{X3} & v_{X4} & v_{X5} & v_{X6} \\ v_{Y1} & v_{Y2} & v_{Y3} & v_{Y4} & v_{Y5} & v_{Y6} \\ 0 & v_{Z2} & v_{Z3} & v_{Z4} & -L_6 C\theta_6\ S\theta_5\ S\theta_{234} & L_6[C\theta_6 C\theta_{234} - C\theta_5 S\theta_6 S\theta_{234} \\ 0 & S\theta_1 & S\theta_1 & S\theta_1 & C\theta_1 S\theta_{234} & C\theta_5\ S\theta_1 - C\theta_1\ C\theta_{234}\ S\theta_5 \\ 0 & -C\theta_1 & -C\theta_1 & -C\theta_1 & S\theta_1 S\theta_{234} & -C\theta_5\ SC - S\theta_1 C\theta_{234} S\theta_5 \\ 1 & 0 & 0 & 0 & -C\theta_{234} & -S\theta_5\ S\theta_{234} \end{bmatrix} \quad (4.58)$$

Where:

$$\begin{aligned} v_{Y1} = {} & d_4 S\theta_1 + a_1 C\theta_1 + L_2 C\theta_2 C\theta_1 + L_6 C\theta_6 \\ & * [S\theta_1\ S\theta_5 + C\theta_5\ C\theta_{234}\ C\theta_1] + d_5 C\theta_1 S\theta_{234} \\ & + L_3\ C\theta_1\ C\theta_{23} - L_6\ C\theta_1\ S\theta_6\ S\theta_{234} \end{aligned} \quad (4.58)$$

$$\begin{aligned} v_{X2} = {} & -C\theta_1\ (L_3\ S\theta_{23} + L_2\ S\theta_2 - d_5\ C\theta_{234} + L_6\ C\theta_{234}\ S\theta_6 \\ & + L_6\ S\theta_{234}\ C\theta_5\ C\theta_6) \end{aligned} \quad (4.59)$$

$$\begin{aligned} v_{y2} = {} & -S\theta_1\ (L_3\ S\theta_{23} + L_2\ S\theta_2 - d_5\ C\theta_{234} + L_6\ C\theta_{234}\ S\theta_6 \\ & + L_6\ S\theta_{234}\ C\theta_5\ C\theta_6) \end{aligned} \quad (4.60)$$

$$\begin{aligned} v_{Z2} = {} & \frac{L_6\ C\theta_{234}\ C\theta_{56}}{2} + L_3\ C\theta_{23} - L_6\ S\theta_{234}\ S\theta_6 + d_5\ S\theta_{234} \\ & + L_2\ C\theta_2\ + \frac{L_6\big(\ C(\theta_{2346} - \theta_5)\big)}{4} \\ & + \frac{L_6\big(\ C(\theta_{2345} - \theta_6)\big)}{4} \end{aligned} \quad (4.61)$$

$$v_{X3} = -C\theta_1 \,(L_3\, S\theta_{23} - d_5\, C\theta_{234} + L_6\, C\theta_{234}\, S\theta_6 + L_6\; S\theta_{234}\, C\theta_5\, C\theta_6) \tag{4.62}$$

$$v_{y3} = -S\theta_1 \,(L_3\, S\theta_{23} - d_5\, C\theta_{234} + L_6\, C\theta_{234}\, S\theta_6 + L_6\; S\theta_{234}\, C\theta_5\, C\theta_6) \tag{4.63}$$

$$v_{Z3} = \frac{L_6\, C\theta_{234}\, C\theta_{56}}{2} + L_3\, C\theta_{23} - L_6\, S\theta_{234}\, S\theta_6 + d_5\, S\theta_{234} + \frac{L_6\big(\, C(\theta_{2346} - \theta_5)\big)}{4} + \frac{L_6\big(\, C(\theta_{2345} - \theta_6)\big)}{4} \tag{4.64}$$

$$v_{X4} = -C\theta_1 \,(L_3\, S\theta_{23} - d_5\, C\theta_{234} - L_3\, S\theta_{23} + L_6\, C\theta_{234}\, S\theta_6 + L_6\; S\theta_{234}\, C\theta_5\, C\theta_6) \tag{4.65}$$

$$v_{Y4} = -S\theta_1 \,(L_3\, S\theta_{23} - d_5\, C\theta_{234} - L_3 S\theta_{23} + L_6\, C\theta_{234}\, S\theta_6 + L_6\; S\theta_{234}\, C\theta_5\, C\theta_6) \tag{4.66}$$

$$v_{Z4} = d_5\, S\theta_{234} - L_6\, S\theta_6\, S\theta_{234} + L_6\, C\theta_5\, C\theta_6\, C\theta_{234} \tag{4.67}$$

$$v_{X5} = L_6\, C\theta_6[\, S\theta_1\, C\theta_5 - C\theta_{234}\, C\theta_1 S\theta_5] \tag{4.68}$$

$$v_{y5} = -L_6\, C\theta_6[\, C\theta_1\, C\theta_5 + C\theta_{234}\, S\theta_1 S\theta_5] \tag{4.69}$$

$$v_{x6} = -L_6\, S\theta_1\, S\theta_5\, S\theta_6 - L_6 C\theta_6\, C\theta_1\, S\theta_{234} - L_6\, C\theta_5\, C\theta_{234}\, C\theta_1\, S\theta_6 \tag{4.70}$$

$$v_{Y6} = L_6\, C\theta_1\, S\theta_5\, S\theta_6 - L_6\, C\theta_6\, S\theta_1\, S\theta_{234} - L_6\, C\theta_5\, C\theta_{234}\, S\theta_1\, S\theta_6 \qquad (4.71)$$

4.5 Detective Vein Robot Workspace and Singularity

There are two main scenarios that prevent a robot from moving to a target position; either the robot can't physically reach the desired point since it is out of its workspace, or the robot can't move to the target position from its current joint configuration (the robot is in a singularity configuration).

The workspace of the Detective Vein Robot, where the robot can reach easily without collision of its links, is almost a holed sphere with a diameter of approximately 0.7m. Figure (4.3) shows the workspace of Detective Vein Robot based on the angle's limitations.

However, sometimes even if the desired point is placed inside the robot workspace, the robot still can't get out of its current configuration and move to the target point, this phenomenon is known as singularity. A singularity is a particular condition where the robot loses one or more of its degrees of freedom and has to provide an almost infinite velocity to joints to get out of this particular pose. From

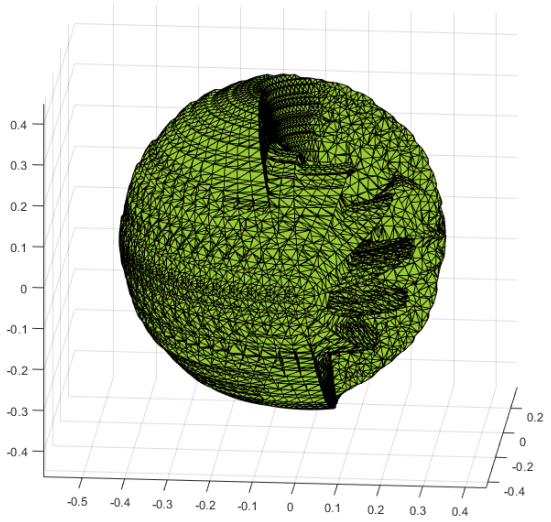


Figure 4.3: Workspace of Detective Vein Robot

a mathematical point of view, singularity occurs when the robot Jacobin matrix isn't full rank or, in other words, its determinant is equal to zero.

In order to determine the singularity poses for the Detective Vein robot, the determinant for the geometrical Jacobin calculated previously is set to Zero, as shown in Equation (4.72).

$$\det(J_g) = 0 \tag{4.72}$$

Based on this, if one of the following conditions is satisfied, the Detective Vein robot will be in a singularity pose.

- $\sin\theta_5 = 0$
- Link 2, Link 3, and Link 4 are on the same extent.
- Link3, and Link 4 are on the same extend

Figure (4.4) show some poses where the Detective Vein robot is in the singularity position

Figure 4.4: Different Detective Vein robot poses with singularity

4.6 Detective Vein Robot Dynamical Model

Robot dynamic modeling is the set of differential equations that expresses the comprehensive relation between forces/torques and the robot's motion. The robot's dynamic behavior is described in terms of joint variables and their derivatives with joint torques. Generally, two approaches are used to drive the robot dynamic equations: the Euler-Lagrange equations and the Recursive Newton-Euler formulation. However, a simpler version of the Euler-Lagrange method is used in obtaining the dynamics model of the Detective-Vein robot, where the inertia of motors and friction are neglected.

When a robot moves, it undergoes a set of different forces such as the force due to its acceleration, the Coriolis forces, etc. Thus, the dynamic equation for a N rigid bodies manipulator is given by Equation (4.73)

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau \tag{4.73}$$

Where:

- $M(q)$: is the symmetric positive definite mass inertia matrix of the system.
- $V(q,\dot{q})$: is the summation matrix of Coriolis and centrifugal terms.
- $G(q)$: is the vector of the Gravitational term.
- τ: input torques/forces vector acting of robot's body

4.6.1 Inertia Matrix

The inertia Matrix is a positive definite matrix of size n × n, where n is the number of joints. It represents the amount of inertia about each joint as a function of joint parameters, where the inertia of the link depends on the robot pose (link is stretched or not). It also reflects the coupling effect between links, where the applied torque/force by the motor in one robot joint will cause acceleration in the coupled joints. The general formula for the inertia matrix is

$$M(q) = \sum m_i \, (J_{vi})^T J_{vi} + (J_{wi})^T \, I_{ci} \, J_{wi} \tag{4.74}$$

Where:

m_i: The mass of each link, obtained using solid works software

J_{vi}: Linear part of Jacobin matrix calculated to the center mass of each link

J_{wi}: Angular part of Jacobin matrix calculated to the center mass of each link

I_{ci}: Inertia Tensor matrix for each link, obtained using solid works software

The inertia matrix of Detective Vein robot is a 6×6 matrix, and it is computed as follow:

$$\begin{aligned} M(q) = {} & (J_{V1})^T * m_1 * J_{V1} + (J_{V2})^T * m_2 * J_{V2} + (J_{V3})^T * m_3 * J_{V3} \\ & + (J_{V4})^T * m_4 * J_{V4} + (J_{V5})^T * m_5 * J_{V5} + (J_{V6})^T * m_6 \\ & * J_{V6} + (J_{w1})^T I_{c1} J_{w1} + (J_{w2})^T I_{c2} J_{w2} \\ & + (J_{w3})^T I_{c3} J_{w3} + (J_{w4})^T I_{c4} J_{w4} + (J_{w5})^T I_{c5} J_{w5} \\ & + (J_{w6})^T I_{c6} J_{w6} \end{aligned} \tag{4.75}$$

4.6.2 Coriolis and Centrifugal forces:

The robot's motion consists of one link rotating above another rotating link. Due to the motion nature of the robot, it undergoes Coriolis and centripetal forces. Coriolis force is a fictitious force that acts on a body moving within a frame that rotates with respect to an inertial frame causing the body to deflect its motion sideways. The Coriolis force acts orthogonally to the rotation axis and the body's velocity vector. Meanwhile, Centripetal force is a radial force that points toward the center, causing the body to keep moving in a circular path.

The Coriolis matrix for the Detective Vein robot has the following form:

$$C(q) = \begin{bmatrix} b_{1,11} & b_{1,22} & b_{1,33} & b_{1,44} & b_{1,55} & b_{1,66} \\ b_{2,11} & b_{2,22} & b_{2,33} & b_{2,44} & b_{2,55} & b_{2,66} \\ b_{3,11} & b_{3,22} & b_{3,33} & b_{3,44} & b_{3,55} & b_{3,66} \\ b_{4,11} & b_{4,22} & b_{4,33} & b_{4,44} & b_{4,55} & b_{4,66} \\ b_{5,11} & b_{5,22} & b_{5,33} & b_{5,44} & b_{5,55} & b_{5,66} \\ b_{6,11} & b_{6,22} & b_{6,33} & b_{6,44} & b_{6,55} & b_{6,66} \end{bmatrix} \quad (4.76)$$

While the centripetal matrix has the following form:

$$B(q)_{6\times 15} = 2 \begin{bmatrix} b_{1,12} & b_{1,13} & b_{1,14} & b_{1,15} & b_{1,16} & b_{1,23} & b_{1,24} & b_{1,25} & b_{1,26} & b_{1,34} & b_{1,35} & b_{1,36} & b_{1,45} & b_{1,46} & b_{1,56} \\ b_{2,12} & b_{2,13} & b_{2,14} & b_{2,15} & b_{2,16} & b_{2,23} & b_{2,24} & b_{2,25} & b_{2,26} & b_{2,34} & b_{2,35} & b_{2,36} & b_{2,45} & b_{2,46} & b_{2,56} \\ b_{3,12} & b_{3,13} & b_{3,14} & b_{3,15} & b_{3,16} & b_{3,23} & b_{3,24} & b_{3,25} & b_{3,26} & b_{3,34} & b_{3,35} & b_{3,36} & b_{3,45} & b_{3,46} & b_{3,56} \\ b_{4,12} & b_{4,13} & b_{4,14} & b_{4,15} & b_{4,16} & b_{4,23} & b_{4,24} & b_{4,25} & b_{4,26} & b_{4,34} & b_{4,35} & b_{4,36} & b_{4,45} & b_{4,46} & b_{4,56} \\ b_{5,12} & b_{5,13} & b_{5,14} & b_{5,15} & b_{5,16} & b_{5,23} & b_{5,24} & b_{5,25} & b_{5,26} & b_{5,34} & b_{5,35} & b_{5,36} & b_{5,45} & b_{5,46} & b_{5,56} \\ b_{6,12} & b_{6,13} & b_{6,14} & b_{6,15} & b_{6,16} & b_{6,23} & b_{6,24} & b_{6,25} & b_{6,26} & b_{6,34} & b_{6,35} & b_{6,36} & b_{6,45} & b_{6,46} & b_{6,56} \end{bmatrix}$$

(4.77)

The term b_{ijk} is calculated using Equation ()

$$b_{ijk} = 0.5\ (m_{ijk} + m_{ikj} - m_{jki}) \quad (4.78)$$

Where:

$m_{ijk} = \frac{\partial m_{ij}}{\partial q_k}$: the derivative of the i row, j column element of the inertia matrix in respect to joint parameter number k.

4.6.3 Gravity Matrix:

Gravity matrix reflects the amount of torque that the motor should supply to counteract the weight force of Robot links, it can be calculated using the following formula:

$$G(q) = \begin{bmatrix} J_{v1} & J_{v2} & J_{v3} & J_{v4} & J_{v5} & J_{v6} \end{bmatrix} \begin{bmatrix} m_1 g \\ m_2 g \\ m_3 g \\ m_4 g \\ m_5 g \\ m_6 g \end{bmatrix} \quad (4.79)$$

Where:

J_{vi}: Linear part of Jacobin matrix calculated to the center mass of each link

g: acceleration of Gravity vector, which is equal to $\begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix}$

m: mass of Link

4.7 Trajectory Planning

Trajectory planning is the foundation of motion control for robots; it consists of finding a time series of sequential joint angles that allows the movement of the robot from an initial to a desired configuration while avoiding body collision over time. Trajectory planning can be performed in the joint or the operational space. Trajectory planning designed in the joint space describes the robot trajectory as a function of joint positions, velocities, and acceleration with respect to time. Meanwhile, trajectory planning performed in the operational space/ task space describes the robot trajectory in terms of end-effector positions, orientations, and their time derivatives.

Since the motion of the Detective Vein robot depends on reaching the suitable injection point regardless of the path, point-to-point trajectory planning is used. The trajectory will be performed in the joint space; since it is easy, simple, and the singularity can be avoided. The joint angles for the initial and final pose of robot, which are necessary for joint space trajectories, are obtained using the inverse kinematics.

There are several joint space trajectory generator methods used in robotics, such as Quintic profile, 2-1-2 profile, bang-bang profile, trapezoidal profile, etc. However, two profiles will be used in generating Detective-Vein robot trajectory planning, first is Quintic profile and the second is Trapezoidal profile.

4.7.1 Quintic Profile

Quintic profile uses a fifth order polynomial to generate trajectories that move through waypoints at specific time points. Quintic profile method provides a continuous position, velocity, and acceleration with six restrictions on the initial and final value for position, velocity, and acceleration, thus making it one of the most commonly used method in robotics. However, the Quintic profile has a drawback presented by its lack of constrictions on maximum acceleration

The Quintic trajectory form is:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{4.80}$$

The six constrains required to solve for the a's parameters are:

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4$$

$$\alpha_0 = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4$$

$$\alpha_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3$$

Where:

q_0: values of the joint angles at the robot initial configuration, obtained using inverse kinematics

q_f: values of the joint angles at the robot final configuration, obtained using inverse kinematics

v_0: joint velocity at time equal t_0

v_f: joint velocity at time equal t_f

α_0: joint acceleration at time equal t_0

α_f: joint acceleration at time equal t_0

4.7.2 Trapezoidal Profile

The trapezoidal profile gets its name from the trapezoid shape of the velocity profile, where the profile breaks the total move time into three sections; acceleration phase, constant velocity phase, and deceleration phase, with the ability to determine the duration for each phase (more control over the maximum velocity and acceleration). The trapezoidal profile is continuous in position and velocity, however in terms of acceleration it lost its continuity.

The Figure (4.5) presents the general desired Trapezoidal Profile for each link of the robot arm, where the whole process should be performed in 4.5 seconds. All links will have the same acceleration phase time of about 1.25 seconds, as well as the deceleration phase, which will also be 1.25 second.

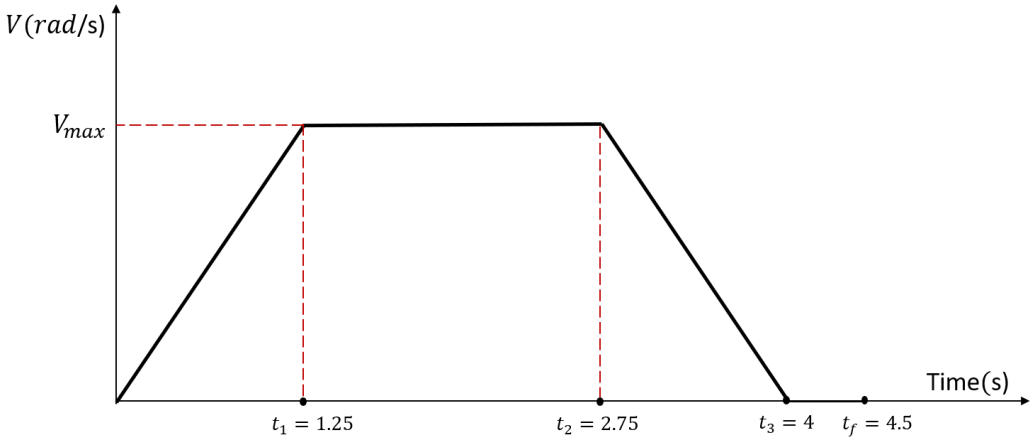


Figure 4.5: Velocity profile used in trapezoidal trajectory of Detective Vein robot

The maximum value of velocity can be determined using Equation (4.81)

$$\theta_f - \theta_i = 0.5 * V_{max} * (t_1 - t_0) + V_{max} * (t_2 - t_1) + 0.5 * V_{max} * (t_3 - t_2) \quad (4.81)$$

And maximum acceleration and deceleration values are calculated using Equation (4.82) and (4.83)

$$acc = \frac{V_{max}}{t_1 - t_0} \quad (4.82)$$

$$dec = -\frac{V_{max}}{t_3 - t_2} \quad (4.83)$$

Where:

θ_f: Joint angle at the final position in rad

θ_i: Joint angle at the initial position in rad

t_0: Starting time in seconds

t_1: End time of acceleration phase in seconds

t_2: Start time of deceleration phase in seconds

t_f: Final time in seconds

4.7.3 Quintic Profile and Trapezoidal profile graphs

The home configuration of the Detective Vein robot is shown in Figure (4.6). However, the goal position for the robot isn't known since it depends on the selected injection point, which mainly relies on the patient's arm and the location of the suitable vein for injection. A comparison between the Quintic and Trapezoidal profiles are made in Figure (4.7), when first joint moved from home configuration to the same final pose, during the same travel time.

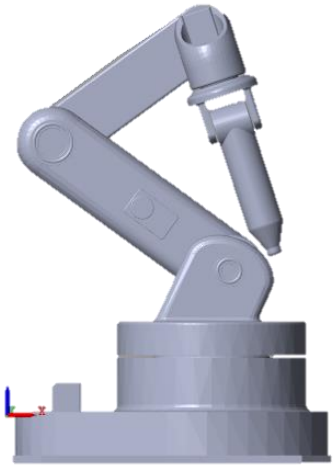
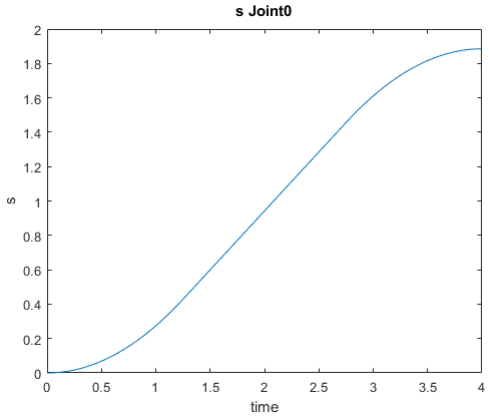
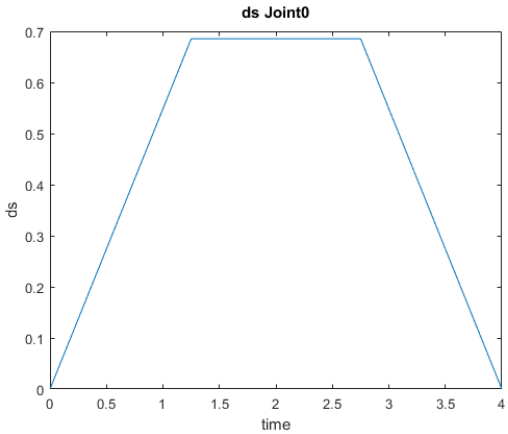


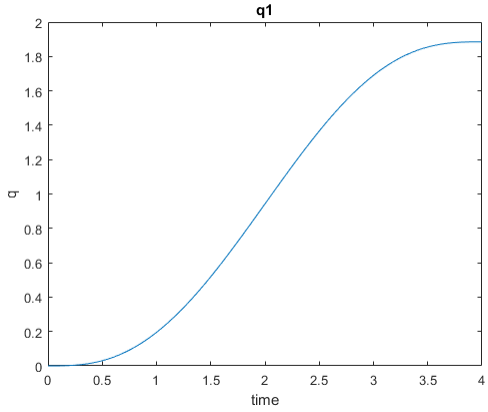
Figure 4.6: Home configuration of Robot

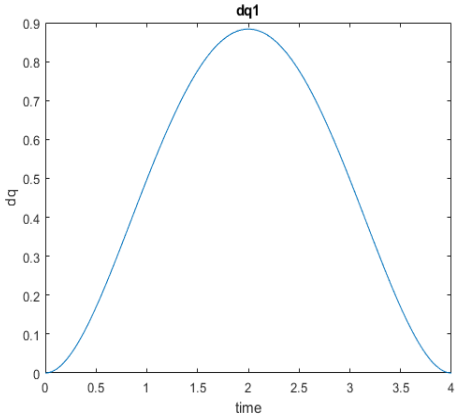
Trapezoidal Profile





Quintic Profile





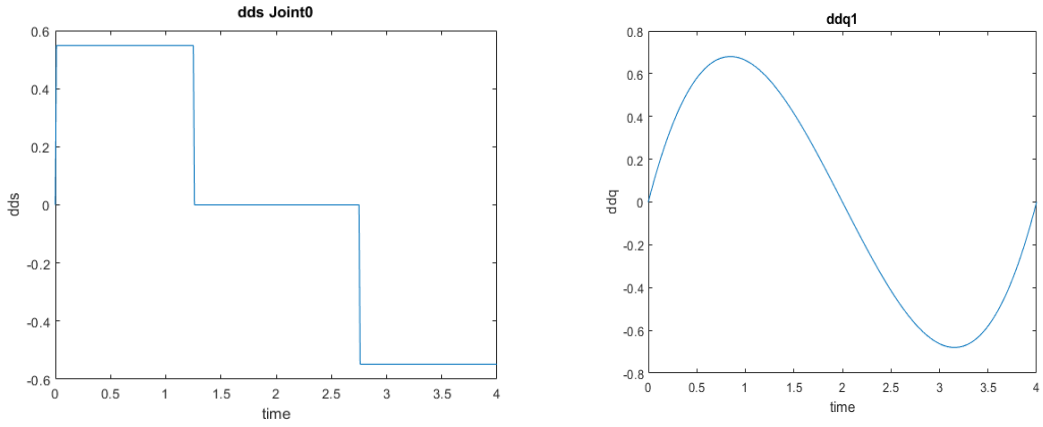


Figure 4.7: Angle, velocity, and acceleration profiles of First joint using Trapezoidal and Quintic method

By comparing the results of the two profile, it can be seen that the maximum velocity and acceleration values in case of trapezoidal are less than that of the Quintic profile, even though the traveled time and distance are the same.

4.8 Motor Selection

The Robot needs a total of 6 servo motors to be able to perform the designed movement. The selection of motors depends on calculating the maximum and root mean square (rms) values of the torque required to move the robot joint, as well as the maximum angular velocity required to perform the robot movement based on the designed trajectory. If the system has any additional gear transmission system, such as in the first, second and third joints of the Detective Vein Robot, it should be taken into consideration before selecting the motor, where a pulley and Timing system of gear ratio 2 is used.

However, as mentioned before, the final position of the robot arm isn't known, thus to determine the maximum torque and rms values required, a set of worst-case scenarios for the robot movement is applied, and the obtained results were used in motor selecting.

Table 4.1: First scenario: computed maximum and rms torques due to the applied motion using polynomial profile

Joints	J0	J1	J2	J3	J4	J5
From -To (angle)	0 → pi	0 → pi	0 → pi/2	0 → pi/2	0 → pi/2	0 → 0
Maximum Torque (LSS)	0.0965	2.2458	0.821701	0.11013	0.0178099	0.0212652
RMS Torque (LSS)	0.05181	1.683	0.5447	0.0637	0.0081	0.0157
Velocity (rad/s)	1.473	1.473	0.736	0.736	0.736	0
Maximum Torque (HSS)	0.0259	1.1229	0.41085	0.0555	0.0089	0.01063
RMS Torque (LSS)	0.7365	0.8415	0.27235	0.0318	0.00405	0.00785
Velocity (rad/s)	2.946	2.946	1.472	1.472	1.472	0

Table 4.1: First scenario: computed maximum and rms torques due to the applied motion

Joints	J0	J1	J2	J3	J4	J5
From -To (angle)	0 → 0.6pi	0 → 0.8pi	-0.6pi → -0.22pi	-0.1pi → pi/2	0 → pi/2	0 → 0

Maximum Torque (LSS)	0.03785	1.96715	0.83315	0.106274	0.0104	0.0185
RMS Torque (LSS)	0.02112	1.37955	0.65683	0.076148	0.004154	0.00815
Velocity (rad/s)	0.8836	1.178	-0.5596	1.129	1.472	1.473
Maximum Torque (HSS)	0.018925	0.98352	0.41657	0.053137	0.0052	0.00925
RMS Torque (LSS)	0.01056	0.68977	0.27235	0.038074	0.002077	0.004075
Velocity (rad/s)	1.7672	2.356	-1.1192	2.258	2.944	2.946

Another trail was performed to find the torque required to lift the fully extended arm 90°, and maximum torque required for joint 2 is 2.35244 with angular speed of 0.736.

Based on the obtained values a set of dynamixel motors are selected:

- XM430-W350-T
- XC330-T288-T
- XL330-M288-T

4.9 Control of Robot

A controller is the brain of the robot arm; it aims to determine the generalized torques that the actuators must apply to the robot joints in order for the robot to execute the assigned task while satisfying both steady-state and transient response requirements. The robot controller can be developed in the joint space or task space. The joint space method is used in controlling the Detective Vein robot, where this control strategy is based on developing a separate controller for each one of the joint parameters.

The main challenge in developing a controller for robotic arm consists in its non-linear mathematical model. To go over this, feedback linearization method is used to convert the robot from non-linear into linear system. This is done by applying input torque equals to the non-linear part of robot dynamic model. Equation (4.73) presents the dynamic model of robot arm

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau \tag{4.73}$$

To convert the non-linear system into a linear system, the input torque is selected as follow

$$\tau = \alpha\, \tau' + \beta \tag{4.84}$$

Where:

$\alpha = M(q), inertia\ matrix$

$\beta = V(q,\dot{q}) + G(q)$

τ': Trajectory Tracking control law, which is given by Equation (4.85)

$$f' = \ddot{x}_d - k'_v(\dot{x} - \dot{x}_d) - k'_p(x - x_d) \tag{4.85}$$

Where:

$\ddot{x}_d$: Desired acceleration

k'_v: Unit mass derivate gain

$\dot{x}$: actual velocity

$\dot{x}_d$: desired velocity

k'_p: Unit mass proportional gain

x:actual position

x_d: desired position

By substituting Equation 4.84 in Equation 4.73, a unit mass system is obtained as follow:

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \alpha\,\tau' + \beta \tag{4.86}$$

Thus,

$$\ddot{q} = \tau'$$

Hence the system is converted to a simple linear system. Therefore, the linear system methods used to determine gains of control. Control scheme shows in Figure is implemented to build Detective Vein robot.

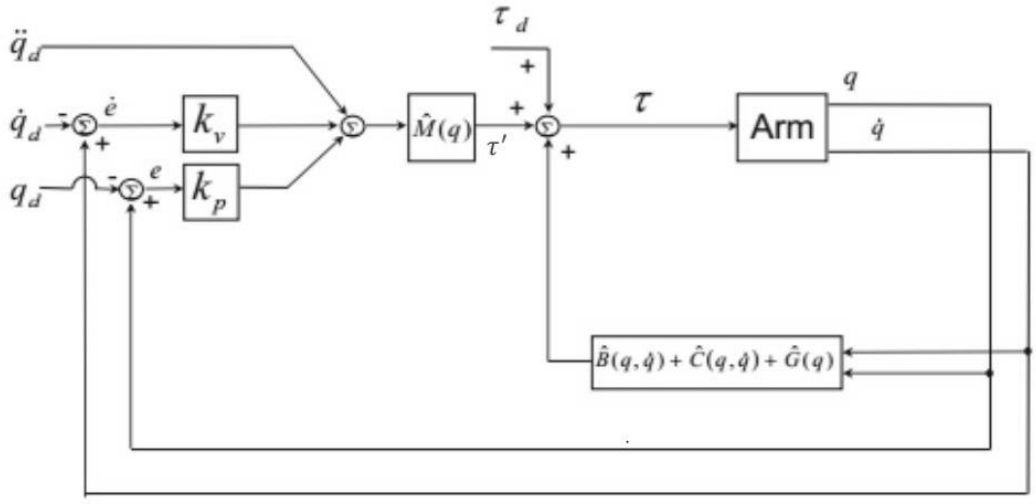


Figure 4.8: PD control scheme for controlling robot Detective Vein robot

CHAPTER 5: IMAGE PROCESSING

The operation of blood drawing depends directly on locating the targeted vein within the required area, which in our case study is the human arm. In the human eye, the light passes through the cornea, which is shaped like a dome, before entering the lens. Both of them work to focus the incoming light correctly on the retina. When the incoming light hits the retina, it will be transformed into electrical signals by the use of special cells called photoreceptors [24].

The same operation can happen in digital cameras: the lens is used to focus the incoming light, and an array of millions of tiny light photocells are used to transform the light photons into an electrical signal and store them in digital form. The digital image can be defined as a two-dimensional image made of a grid of tiny points, each assigned a specific color and intensity, stored in binary format in many formats such as PNG, JPEG, etc., so they can operate on digital devices [25].

The digital image can be displayed on more than one scale, as the targeted output will be. The main scale is the Red, Green, and Blue (RGB) scale, which is a combination of three colors: red, green, and blue. By combining the intensity of these colors, any other color can be created by using three channels. The values of the color range (0–255) are written as the following (255,255,255), which produce the white color. Any change in the numerical numbers will change the color and its intensity [26].

To manipulate the image to locate the veins, the RGB scale will not be enough. Due to the nature of the IR light and the way it interacts with the CO2 in the blood, the veins will be marked with black. The RGB scale cannot display the veins enough because of the variety of colors and intensities in the scale, which introduced the next stage in the image processing process, the grayscale, which is a single-channel scale that represents the intensity of the light in a black-and-white image without considering color information [27]. The output image of the scale will range from white color passing by the gray ending with the black color. More simply, it’s a derivative of a gray color. The gray scale can give a clear view of the veins, which introduces the third stage in the binary scale's image processing process. The binary

image can be obtained using the threshold algorithm, which uses a reference value of pixel intensity where any pixel assigned a value below the targeted intensity will be automatically set to zero, which represents the white color. On the other hand, any other pixel assigned a value equal to the targeted intensity or higher will be assigned a value of 1, meaning a black color [28], [29].

Through this process, the binary image can be formed by using an input image, turning the input image into a grayscale, and then applying the threshold algorithm to transform the image into the binary scale to locate the target vein. To obtain such an output, a sequence of algorithms and functions must be used, starting with:

5.1 GAUSSIAN low-pass filter

A low-pass filter is used to reduce the level of noise in the image while preserving the edges. The low-pass filter blocks the high-frequency signals, allowing only the low-frequency components to pass, which smooths the image by working as a filter kernel, which means that the filter weights are calculated based on a Gaussian distribution. The kernel used is a three-by-three, which can be seen in Figure (5.1):

A low-pass filter is used to reduce the level of noise in the image while preserving the edges. The low-pass filter blocks the high-frequency signals, allowing only the low-frequency components to pass, which smooths the image by working as a filter kernel, which means that the filter weights are calculated based on a Gaussian distribution. The kernel used is a three-by-three, which can be seen below.

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 5.1: Three by three kernel matrix

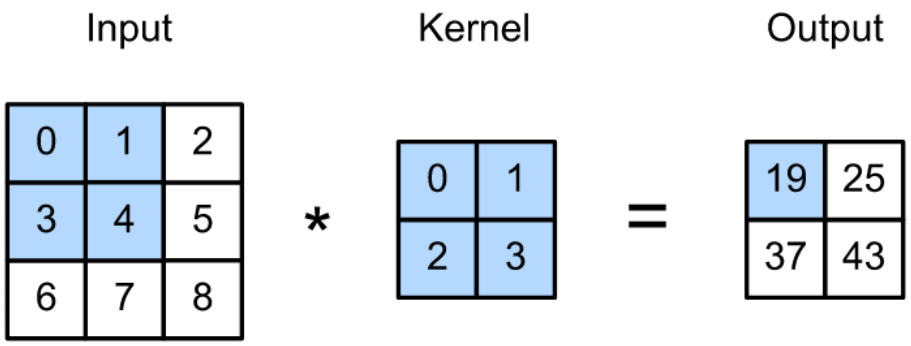


Figure 5.2: Kernel math

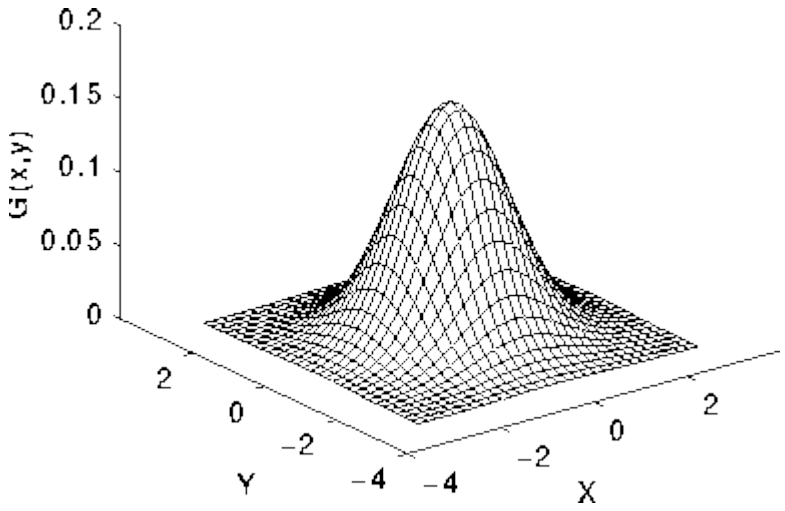


Figure 5.3: Gaussian distribution of an image in the spatial domain

The kernel size is (2×2) matrix and the picture are (3×3) matrix the kernel will start from the top left of the image, the pixels will be multiplied and the sum will be the new intensity value because of the nature of the Gaussian distribution the distribution will be normally distributed, as shown in Figure (5.3):

The code can be implemented can be seen below:

cv2.GaussianBlur(Input, k-size, sigma)

Where:

- Input: is the input image

- k-size: is the size of the kernel which is (3×3) matrix
- sigma: is the standard deviation of the Gaussian kernel, where the default is two

For more filtration, a masking image is used with a bounding range starting from 15 to 250 so the black color is neglected to perform high threshold to the image.

For more enhancement in the filtration process, we set another bad-pass filter to filter the non-ranged pixels and set their values into a white color to enhance the threshold process accuracy, the bounding range starts from (15 to 250).

5.2 Adaptive Threshold

The use of Python programing language is to construct a working algorithm that can detect the veins after processing the incoming live video from the camera. The veins hold a color that is not equal to the skin color which allows the human eye to locate them if the veins are prominent enough due to color differences, the human eye can locate them at the same time can be applied in programming using the threshold method.

Thresholding is a technique used in digital image processing to form a separation between the regions in the image by changing pixels intensity, it can also be used to separate between the objects and the background in the image, making the edges more defined in the image. Many techniques can be applied such as a global, adaptive, multi-level, edge-based, and color-based threshold. In our case study, the adaptive threshold will be used because it calculates the threshold value which is calculated for each pixel locally referenced to the surrounding pixels.

The adaptive method depends on the neighborhood of the pixel allowing a more accurate threshold in the variable lighting conditions, a kernel with a chosen size is applied to the image for each pixel to calculate the local threshold value for each pixel. Based on this calculation, the Gaussian Mean, and Sauvola methods are applied. The Gaussian Mean method was applied which calculates the mean of the intensity of surrounding pixels, mathematically the calculations can be expressed as follows:

$$T_{(X,Y)} = mean(X, Y) - C \text{ [30]}$$

Where:

T: represents the value of the threshold calculated by the mean method.

The constant C is used to adjust the threshold level which is inversely proportional to the threshold. As a Python code, the adaptive threshold can be described as follows:

cv2.adaptiveThreshold(source, maxVal, adaptiveMethod, thresholdType, block size, Constant).

Where:

- source: is the input image and it must be in the grayscale.
- maxVal: represents the maximum value assigned to pixels that exceed the local threshold value.
- adaptiveMethod: decides which method to use by the useable methods.
- thresholdType: decides the color of the background and the foreground.
- blocksize: defines the size of the kernel and the constant is the C constant which affects the level of threshold.

The output of the threshold function is a binary image consisting of two colors only the white and black color, to acquire the veins only the boundaries of the arm that are produced because of the change in light intensity must be removed to only acquire the veins this can be done in two steps, firstly be locating the outer counters of the image then masking them into a new image, after the masking is done a subtraction of images can be done the masked image only contains the outer boundaries as well as the threshold image. Subtracting these images will produce a new image containing the veins only.

To locate the image boundaries the following code can be used:

cv2.drawContours(image, contours, contourIdx, color, thickness, lineType, hierarchy, maxLevel, offset)

Where:

- Image: represents the input image.
- Contours: This is a list of contours to draw.
- contourIdx: a parameter indicating a contour to draw the negative value means all the contours to be drawn.
- Color parameter: decides the color of the contours.
- Thickness: changes the thickness of the line as the user wants.
- lineType: type of line used to draw contours.
- Hierarchy: is an optional input vector containing information about the image topology. It is required for drawing hierarchical contours.
- maxLevel: is the maximal level for drawn contours. If 0, only the contour itself is drawn. If 1, the contour and all its direct children are drawn. If 2, the contour and all its children are drawn
- offset optional parameters to offset the contours.

After the threshold image is obtained a counter-founder algorithm will be used to locate the outer boundaries of the hand to perform the rest of the image processing algorithm.

5.3 Contours Finder

The following function relies on image segmentation and contour detection criteria, by taking the binary image as an input (in our case the input will be the threshold image) the function will start extracting the contours of the image by locating the contours regions of the white pixels. The function can be called using the following formula:

contours, hierarchy = cv2.findContours(Image, Mode, Method, [offset])

Where:

- Image: represents the input image which is in binary formatting.

- Mode: The parameter specifies the retrieval method of the contours, in our case we used cv2.RETR_EXTERNAL because in the case study, the external contours are the targeted contours neglecting the rest.
- Method: a parameter specifies the approximation method of the contours, in our case, we used (cv2. CHAIN_APPROX_SIMPLE) because it expresses the lines with their endpoints [31].

Then a subtraction operation takes place so the threshold image and contoured image are subtracted from each other deleting the non-filtered noise and removing the outer boundaries leaving only the detected veins.

In the end, morphological operations take place to enhance separation. Calculating the number of connected components, and the opening and closing operation is used because the opening operation is a combination of erosion followed by dilation removing noise from the image, and smoothing the edges. Where the closing operation is a combination of a dilation followed by erosion to fill gaps in the contours of objects and smooth the boundaries.

The time of each stage was calculated to be as follows:

To obtain the input image, the image processing function can be put in use, the feed of the camera must be obtained from the raspberry pi due to the type of the camera which only connects to Raspberry Pi to obtain such connection with the targeted device which is in or case of study a PC a Real-Time Streaming Protocol (RTSP). RTSP protocol is a network communication method that allows the connection between two computers to create communication between them by providing secure access for users.

RTSP relies on a client-server model, meaning that the client connects to the RTSP server by assigning a public key to verify the identity of the RTSP server; alternatively, a username and password can be used to log in to the server. In the Raspberry Pi module, the RTSP can be easily enabled, but in the window, the client and the server must be installed on the device, and you must manually enable the RTSP server to open the server [32].

The feed will be passed to the Python compiler, which is PyCharm with Python version 3.7.0. The RTSP opens a server connection between the Raspberry Pi and the host device using the VideoLAN media player (python-vlc) library. When the feed is obtained, the image processing function can start to locate the desired vein through a sequence of the following steps:

After the desired vein is chosen, the robot mechanism will move to the desired location with the desired trajectory. To ensure that the robot moves at the right angle relative to the position of the hand, the position and orientation of the hand must be calculated or measured. The X position and Y position of the hand can be measured from the image processing function mentioned above; the Z position will be assumed to be of constant length for the orientation of the hand in the three axes. The MPU-6050 sensor will be used.

The working principle of such a sensor is that a 3-axis Accelerometer and 3-axis Gyroscope are embedded into the MPU-6050 to measure acceleration, velocity, orientation, and displacement. With the help of a digital motion processor, the MPU-6050 can operate these complex calculations.

The MEMS accelerometer is used to measure the linear acceleration of the target object reference to a mobile frame where the frame changes concerning the moving object. Applying the principle of a mass on a spring which can be expressed as, when the target object starts to accelerate due to its inertia the mass will tend to fight the change in motion causing the spring to be stretched or compressed, creating a force which is detected and corresponds to the applied acceleration.

MEMS accelerometers can achieve precise linear acceleration detection in two orthogonal axes using a pair of silicon MEMS detectors formed by spring masses. When the sensor is subjected to a linear acceleration along its sensitive axis, the proof mass will start to resist motion due to its inertia. The displacement of the mass induces a differential capacitance between the moving and fixed silicon fingers, which is proportional to the applied acceleration. The change in capacitance is measured with a high-resolution ADC, and the acceleration is obtained by the rate of change in capacitance

The MEMS Gyroscope measures the Coriolis Effect. The Coriolis Effect states the following: "When a mass moves in a particular direction with velocity and an external angular motion is applied to it, a force is generated and that causes a perpendicular displacement of the mass." The generated force is called the Coriolis Force, and this phenomenon is known as the Coriolis Effect. The rate of displacement will be directly related to the angular motion applied.

The MEMS Gyroscope contains a set of four proof masses. When an angular motion is applied due to the Coriolis Effect, it will cause a change in capacitance between the masses depending on the axis of the angular movement. This change in capacitance is sensed as an electrical signal, and with proper calibration, the values are obtained as numerical values [33].

The MPU6050 module has a total of 8 pins, as shown in Figure (5.6). In which at least 4 pins are necessary for the interfacing.

Where:

- VCC: Provides the power to the MPU-6050 from the microcontroller or an external power source with the proper voltage value.
- GND: Ground Connected to the Ground pin of the microcontroller or an external power source with the proper voltage value.
- SCL: Serial Clock Used for providing clock pulse for I2C Communication.
- SDA Serial Data Used for transferring Data through I2C communication.
- XDA Auxiliary Serial Data - Can be used to interface other I2C modules with MPU6050.
- XCL Auxiliary Serial Clock - Can be used to interface other I2C modules with MPU6050.
- ADD/ADO Address select pin if multiple MPU6050 modules are used.
- INT: Interrupt pin to indicate that data is available for MCU to read.

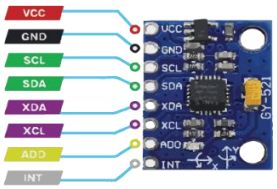


Figure 5.6: MPU-6050 Module PINOUT [33]

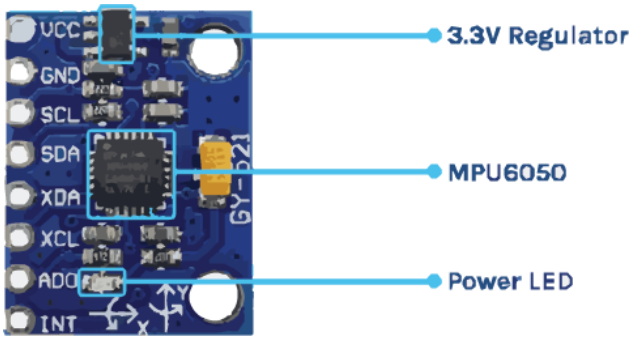


Figure 5.7: MPU 6050 Module Parts [33]

MPU6050 module consists of an MPU6050 IMU chip, AP2112K 3.3V regulator, I2C pull-up resistors, bypass capacitors, and, a power led which indicates the power status of the module as shown in Figure (5.7).

After calculating the position of the veins and determining the hand's orientation, the robot may move to the appropriate position and then compute the angles needed to complete the process. The Sympy library, a Python library for symbolic mathematics, can perform the specified mathematical operations, while the standard Numpy library is unable to.

Finally, the position and orientation of the human hand are obtained using the MPU-6050 and the image processing algorithm, but the PU-6050 is designed to interface with the Arduino or any other microcontroller, but the Arduino can’t automatically be connected with PyCharm, which is a Python compiler, so the Pyserial library is used. The Pyserial library is a Python library that provides

interfaces for serial communication between several devices, such as microcontrollers, allowing the user to send and receive data through the serial port.

The operation can be simply summarized in the following figure as following:

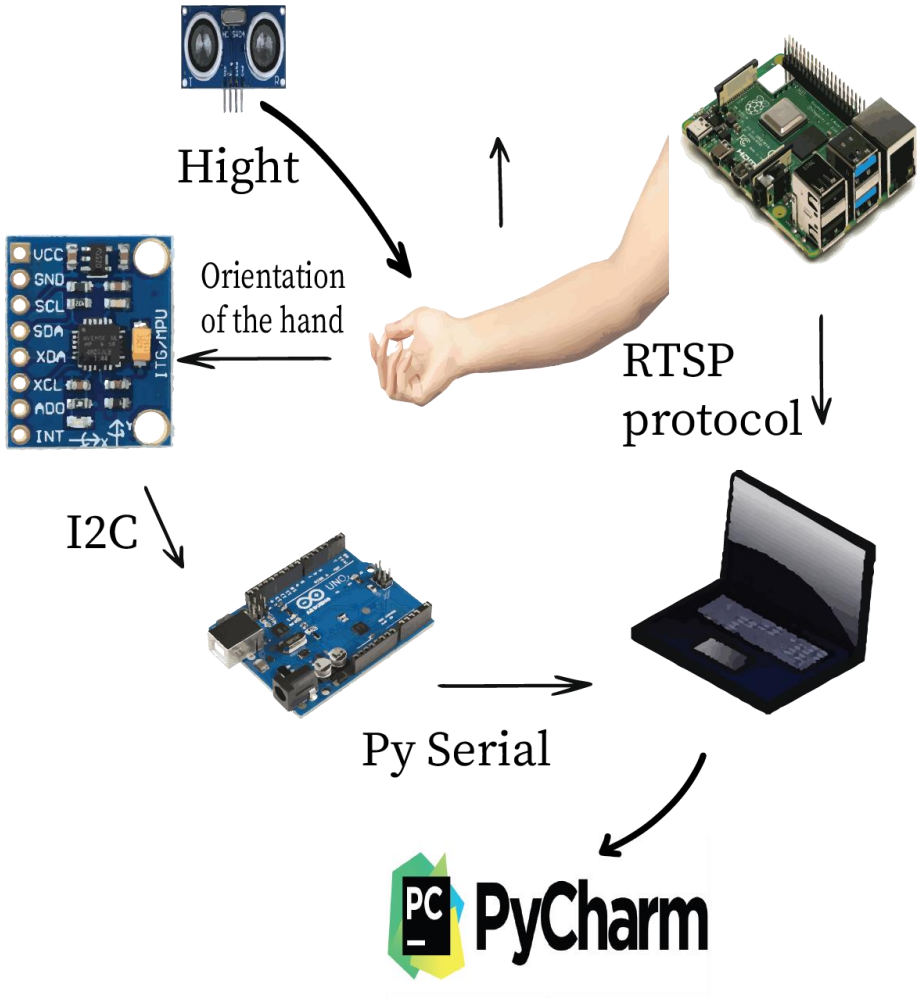


Figure 5.8: Image processing Flow Chart

5.4 Results

Figure 5.9: Input image

Figure 5.10: Filtered image

Figure 5.11: Threshold image

Fig 5.12: Contoured image

Figure 5.13: Final image

REFERENCES

[1] C. Ialongo and S. Bernardini, "Phlebotomy, a bridge between laboratory and patient," Biochem Med (Zagreb), vol. 26, pp. 17-33, 2016.

[2] T. Surgery. (2018, January 31). Intravenous Cannulation (16 ed.). Available: https://teachmesurgery.com/skills/clinical/cannulation/

[3] M. Lamperti and M. Pittiruti, "II. Difficult peripheral veins: turn on the lights,"Br J Anaesth, vol. 110, pp. 888-91, Jun 2013.

[4] (2019). Available: https://www.istockphoto.com/photo/nurse-takes-a-blood- sample-from-arm-vein-performing-a-venipuncture-gm1181857164-331645182

[5] (2020). IV cannulation procedure for nurses - How to place a peripheral IV catheter (PIVC). Available: {$plugin.tx_news.opengraph.site_name}

[6] C. T. Pan, M. D. Francisco, C. K. Yen, S. Y. Wang, and Y. L. Shiue, "Vein Pattern Locating Technology for Cannulation: A Review of the Low-Cost Vein Finder Prototypes Utilizing near Infrared (NIR) Light to Improve Peripheral Subcutaneous Vein Selection for Phlebotomy," Sensors (Basel), vol. 19, Aug 16 2019.

[7] A. Madrid García and P. R. Horche, "Light source optimizing in a biphotonic vein finder device: Experimental and theoretical analysis," Results in Physics, vol. 11, pp. 975-983, 2018/12/01/ 2018.

[8] M. Cengiz, P. Ulker, H. J. Meiselman, and O. K. Baskurt, "Influence of tourniquet application on venous blood sampling for serum chemistry, hematological parameters, leukocyte activation and erythrocyte mechanical properties," Clin Chem Lab Med, vol. 47, pp. 769-76, 2009.

[9] E. O. MBChB. (January 19th). Median cubital vein. Available: https://www.kenhub.com/en/library/anatomy/median-cubital-vein

[10] L. Finlayson, I. R. M. Barnard, L. McMillan, S. H. Ibbotson, C. T. A. Brown, E. Eadie, et al., "Depth Penetration of Light into Skin as a Function of Wavelength from 200 to 1000 nm," Photochemistry and Photobiology, vol. 98, pp. 974-981, 2022.

[11] Skin: Layers,Structure and Function. Available: https://my.clevelandclinic.org/health/articles/10978-skin

[12] (2022). Near Infrared Measurements – How Do They Work? Available: https://www.kpmanalytics.com/articles-insights/near-infrared-measurements- how-do-they-work

[13] A. Sakudo, "Near-infrared spectroscopy for medical applications: Current status and future perspectives," Clinica Chimica Acta, vol. 455, pp. 181-188, 2016/04/01/ 2016.

[14] C. Ash, M. Dubec, K. Donne, and T. Bashford, "Effect of wavelength and beam width on penetration in light-tissue interaction using computational methods," Lasers Med Sci, vol. 32, pp. 1909-1918, Nov 2017.

[15] F. Chandra, A. Wahyudianto, and M. Yasin, "Design of vein finder with multi tuning wavelength using RGB LED," Journal of Physics: Conference Series, vol. 853, p. 012019, 2017/05/01 2017.

[16] M. D. Francisco, W.-F. Chen, C.-T. Pan, M.-C. Lin, Z.-H. Wen, C.-F. Liao, et al., "Competitive Real-Time Near Infrared (NIR) Vein Finder Imaging Device to Improve Peripheral Subcutaneous Vein Selection in Venipuncture for Clinical Laboratory Testing," Micromachines, vol. 12, p. 373, 2021.

[17] M. Dhakshayani and M. Sikkandar, Economically Affordable and Clinically Reliable Vein Finder, 2015.

[18] M. Wadhwani, A. Sharma, A. Pillai, N. Pisal, and M. Bhowmick, "Vein Detection System using Infrared Light," 2016.

[19] A. Chen, K. Nikitczuk, J. Nikitczuk, T. Maguire, and M. Yarmush, "Portable robot for autonomous venipuncture using 3D near infrared image guidance," Technology (Singap World Sci), vol. 1, pp. 72-87, Sep 2013.

[20] e.-c. systems. (21st January). 5MP Monochrome USB camera. Available: https://www.e-consystems.com/ar0521-monochrome-usb-nir-camera.asp

[21] Villalobos, J., et al. (2022). "Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot." Robotics 11(6): 137.

[22] Hawkins, K. P. (2013). "Analytic inverse kinematics for the universal robots UR-5/UR-10 arms." Georgia Institute of Technology, Tech. Rep.

[23]

[24] research., T. n. s. l. i. v. (2022). "How the Eyes Work." Retrieved 16-May-2023, from https://www.nei.nih.gov/learn-about-eye-health/healthy-vision/how-eyes-work.

[25] Woodford, C. (2022). "Digital cameras." from https://www.explainthatstuff.com/digitalcameras.html.

[26] Services, U. I. T. (2021). "What are the RGB values of some common colors?". 16-May-2023, from https://kb.iu.edu/d/aetf.

[27] Prakash, S. and J. Yeom (2014). Nanofluidics and microfluidics: systems and applications, William Andrew.

[28] Bovik, A. C. (2005). Handbook of Image and Video Processing (Communications, Networking, and Multimedia). Inc.6277 Sea Harbor Drive Orlando, FLUnited States, Academic Press.

[29] Dana Harry Ballard, C. M. B. (1982). Binary Image Analysis. Computer Vision, Prentice Hall Professional Technical Reference.

[30] Gonzalez, R. C. and R. E. Woods (2008). Digital image processing. Upper Saddle River, N.J., Prentice Hall.

[31] OpenCV. "What are contours?". Retrieved 5-August-2022, from https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.

[32] Erik, a. (2019). "Raspberry Pi: Stream video to VLC player, using rtsp protocol."23-July-2023,from https://helloraspberrypi.blogspot.com/2019/02/raspberry-pi-stream-video-to-vlc-player.html.

[33] Joseph, J. (2022). " How Does the MPU6050 Accelerometer & Gyroscope Sensor Work and Interfacing It With Arduino." from https://circuitdigest.com/microcontroller-projects/interfacing-mpu6050-module-with-arduino.

*

APPENDIX A: MATLAB SOURCE CODES

```
%% Veins Detector
clc
clear all
%%

syms  Theta1 Theta2 Theta3 Theta4 Theta5 Theta6 a1 L2 L3 d4 d5 L6
d1 real
%%
a1 = 0.012;
d4 = 0.013;
L3 = 0.12;
L2 = 0.15;
d5 = 0.065;
L6 = 0.095;

%}

%% Forward Kinematics

%% DH_ Parameter


DH = [ pi/2   a1   0    Theta1; %1-2
       0      L2   0    Theta2; %2-3
       0      L3   0    Theta3; %3-4
       pi/2   0   d4    Theta4; %4-5
      -pi/2   0   d5    Theta5; %5-6
        0     L6   0    Theta6]; %6-end

%% Transformation Matrix From 0 to 1
i=1;

T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                  0
1               ];

%% Transformation Matrix From 1 to 2
i=2;

T2 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
```

```
       0                       0                                   0
1               ];

%% Transformation Matrix From 2 to 3
i=3;

T3 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                   0
1               ];

%% Transformation Matrix From 3 to 4
i=4;

T4 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                   0
1               ];

%% Transformation Matrix From 4 to 5
i=5;

T5 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                   0
1               ];


%% Transformation Matrix From 5 to 6
i=6;

T6 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                   0
1               ];



%% Transformation Matrices
T01 = T1;
```

```
T02 = T1*T2;
T03 = T1*T2*T3;
T04 = simplify(T1*T2*T3*T4);
T05 = simplify (T04*T5);
T06 = simplify(T1*T2*T3*T4*T5*T6);
%T07 = simplify(T1*T2*T3*T4*T5*T6*T7) %Tip Point of End Effector
%% Jacobin Matrix
O0 = [0;0;0];
O1= [T01(1:3,4)];
O2= [T02(1:3,4)];
O3= [T03(1:3,4)];
O4= [T04(1:3,4)];
O5= [T05(1:3,4)];
O6= [T06(1:3,4)];
Z0 = [0;0;1];
JV_1= cross(Z0, O6);
JV_2= cross(T01(1:3,3), (O6-O1));
JV_3= cross(T02(1:3,3), (O6-O2));
JV_4= cross(T03(1:3,3), (O6-O3));
JV_5= cross(T04(1:3,3), (O6-O4));
JV_6 = cross (T05(1:3,3), (O6-O5));
Jv= simplify ([JV_1,JV_2,JV_3,JV_4,JV_5,JV_6]);
%Jv= simplify(jacobian(T06(1:3,4),[Theta1,Theta2,Theta3, Theta4,
Theta5, Theta6 ]));
Jw = simplify([[0;0;1],T01(1:3,3), T02(1:3,3) , T03(1:3,3),
T04(1:3,3), T05(1:3,3)]);

%% Singularity
J = simplify([Jv;Jw]);
Theta = sym('Theta',[1 6]);
theta = sym('theta',[1 6]);
J = subs(J,Theta,theta); % use theta to make the output use greek
letter symbols
assume(theta,'real'); % the angles are real, so let's assume that
to be the case
D(theta) = simplify(det(J),100) %

%% Position of Robot
%{
x = [T01(1,4) T02(1,4) T03(1,4) T04(1,4) T05(1,4) T06(1,4)];
y = [T01(2,4) T02(2,4) T03(2,4) T04(2,4) T05(2,4) T06(2,4) ];
z = [T01(3,4) T02(3,4) T03(3,4) T04(3,4) T05(3,4) T06(3,4) ];


plot3(x(1),y(1),z(1),'o')
hold on
plot3(x(2),y(2),z(2),'o')
hold on
plot3(x(3),y(3),z(3),'o')
hold on
plot3(x(4),y(4),z(4),'o')
hold on
plot3(x,y,z)
grid on
%}
```

```
%% Dynamical Model of Robot
%% Center of Mass Dimensions

syms a1_CM L2_CM L3_CM d4_CM d5_CM L6_CM real
syms m1 m2 m3 m4 m5 m6  IC1 IC2 IC3 IC4 IC5 IC6


a1_CM = 0.02963;
L2_CM = 0.07239;
L3_CM = 0.05777;
d4_CM = 0.02757;
d5_CM = 0.0229;
L6_CM = 0.01653;
%}
%% Mass of Links (Kg)

m1 = 0.55069 + 0.102;
m2 = 0.37143 + 0.102;
m3 = 0.2724 + 0.102;
m4 = 0.06677 + 0.102;
m5 = 0.03629 + 0.102;
m6 = 0.01653 + 0.102;

%% Inertia matrix
IC1 =  (m1/0.55069)*[1.5070285 *10^-3     0                  0;
                          0          1.2562684*10^-3        0;
                          0               0
7.5836252*10^-4];

IC2 = (m2/0.37143)*[ 1.0827013*10^-4      0                   0;
                          0          1.19223728*10^-3        0;
                          0               0
1.2303038*10^-3];

IC3 = (m3/0.2724)*[ 6.094787*10^-5    -3.477810^-8          0;
                   -3.477810^-8    6.7472225*10^-4         0;
                       0               0
6.8987888*10^-4];

IC4 = (m4/0.06677)* [ 3.082979*10^-5        0                   0;
                          0             2.788223*10^-5          0;
                          0                0
1.44917*10^-6];

IC5 = (m5/0.03629)*[ 8.59701*10^-6        0                 -
0.03*10^-9;
                          0             9.29722*10^-6           0;
                   -0.03*10^-9             0
6.73427*10^-6];

IC6 = (m6/0.01653)* [ 2.279309*10^-5        0                   0;
                          0             2.269576*10^-5          0;
                          0                0
2.81447*10^-6];
%}
```

```
%% DH_ Parameter ( to the center of mass)


DH = [ pi/2    a1_CM    0     Theta1; %1-2
       0       L2       0     Theta2; %2-3
       0       L3       0     Theta3; %3-4
       pi/2    0        d4    Theta4; %4-5
      -pi/2    0        d5    Theta5; %5-6
       0       L6       0     Theta6]; %6-end


i=1;

T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                  0
1               ]

Jv1 = jacobian(T1(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%%
DH =[ pi/2     a1     0     Theta1; %1-2
       0    L2_CM     0     Theta2; %2-3
       0       L3     0     Theta3; %3-4
       pi/2    0      d4    Theta4; %4-5
      -pi/2    0      d5    Theta5; %5-6
       0       L6     0     Theta6]; %6-end

i=1;
%From 1 to center mass of Link 1
T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                  0
1               ];

i=2;
%From 2 to center mass of Link 2
T2 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                  0
1               ];

T02= T1*T2;
```

```
Jv2 = jacobian(T02(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%%
DH = [ pi/2    a1    0     Theta1; %1-2
       0       L2    0     Theta2; %2-3
       0    L3_CM    0     Theta3; %3-4
       pi/2    0     d4    Theta4; %4-5
      -pi/2    0     d5    Theta5; %5-6
        0      L6    0     Theta6]; %6-end

i=1;
%From 1 to center mass of Link 1
T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                        0                                0
1               ];

i=2;
%From 2 to center mass of Link 2
T2 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                        0                                0
1               ];
i=3;
%From 3 to center mass of Link 3
T3 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                        0                                0
1               ];

T03= T1*T2*T3;

Jv3 = jacobian(T03(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%%
DH = [ pi/2    a1    0      Theta1; %1-2
       0       L2    0      Theta2; %2-3
       0       L3    0      Theta3; %3-4
       pi/2    0   d4_CM    Theta4; %4-5
      -pi/2    0    d5      Theta5; %5-6
        0      L6    0      Theta6]; %6-end
```

```
i=1;
%From 1 to center mass of Link 1
T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];

i=2;
%From 2 to center mass of Link 2
T2 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];
i=3;
%From 3 to center mass of Link 3
T3 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];
i=4;
%From 4 to center mass of Link 4
T4 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];

T04= T1*T2*T3*T4;

Jv4 = jacobian(T04(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%%
DH = [ pi/2    a1    0       Theta1; %1-2
       0       L2    0       Theta2; %2-3
       0       L3    0       Theta3; %3-4
       pi/2    0    d4       Theta4; %4-5
      -pi/2    0   d5_CM     Theta5; %5-6
        0      L6    0       Theta6]; %6-end
```

```
i=1;
%From 1 to center mass of Link 1
T1 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                0
1               ];

i=2;
%From 2 to center mass of Link 2
T2 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                0
1               ];
i=3;
%From 3 to center mass of Link 3
T3 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                0
1               ];
i=4;
%From 4 to center mass of Link 4
T4 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                0
1               ];

i=5;
%From 5 to center mass of Link 5
T5 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                0
1               ];

T05= T1*T2*T3*T4*T5;
```

```
Jv5 = jacobian(T05(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%%
DH = [ pi/2    a1    0       Theta1; %1-2
       0       L2    0       Theta2; %2-3
       0       L3    0       Theta3; %3-4
       pi/2    0    d4       Theta4; %4-5
      -pi/2    0    d5       Theta5; %5-6
       0    L6_CM   0       Theta6]; %6-end

i=1;
%From 1 to center mass of Link 1
T1 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];

i=2;
%From 2 to center mass of Link 2
T2 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];
i=3;
%From 3 to center mass of Link 3
T3 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];
i=4;
%From 4 to center mass of Link 4
T4 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))       cos(DH(i,4))*cos(DH(i,1))          -
cos(DH(i,4))*sin(DH(i,1))              DH(i,2)*sin(DH(i,4));
       0                  sin(DH(i,1))
cos(DH(i,1))                           DH(i,3);
       0                       0                                0
1               ];
i=5;
%From 5 to center mass of Link 5
T5 = [ cos(DH(i,4))       -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))              DH(i,2)*cos(DH(i,4));
```

```
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))         -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                  0
1               ];

i=6;
%From 5 to center mass of Link 5
T6 = [ cos(DH(i,4))      -sin(DH(i,4))*cos(DH(i,1))
sin(DH(i,4))*sin(DH(i,1))             DH(i,2)*cos(DH(i,4));
       sin(DH(i,4))      cos(DH(i,4))*cos(DH(i,1))         -
cos(DH(i,4))*sin(DH(i,1))             DH(i,2)*sin(DH(i,4));
       0                 sin(DH(i,1))
cos(DH(i,1))                          DH(i,3);
       0                      0                                  0
1               ];

T06= T1*T2*T3*T4*T5*T6;

Jv6 = jacobian(T06(1:3,4),[Theta1,Theta2,Theta3,
Theta4,Theta5,Theta6]);

%% Find JW

JW1  = [[0;0;1] [0;0;0] [0;0;0] [0;0;0] [0;0;0] [0;0;0]] ;
JW2  = [[0;0;1] T1(1:3,3) [0;0;0]    [0;0;0]    [0;0;0]
[0;0;0]];
JW3  = [[0;0;1] T1(1:3,3) T02(1:3,3) [0;0;0]    [0;0;0]
[0;0;0]];
JW4  = [[0;0;1] T1(1:3,3) T02(1:3,3) T03(1:3,3) [0;0;0]
[0;0;0]];
JW5  = [[0;0;1] T1(1:3,3) T02(1:3,3) T03(1:3,3) T04(1:3,3)
[0;0;0]];
JW6  = [[0;0;1] T1(1:3,3) T02(1:3,3) T03(1:3,3) T04(1:3,3)
T05(1:3,3)];

%% Mass Matrix
o1 = vpa((m1*Jv1'*Jv1),6);
o2 = vpa((JW1'*IC1*JW1),6);
o3 = vpa((m2*Jv2'*Jv2),6);
o4 = vpa((JW2'*IC2*JW2),6);
o5 = vpa((m3*Jv3'*Jv3),6);
o6 = vpa((JW3'*IC3*JW3),6);
o7 = vpa((m4*Jv4'*Jv4),6);
o8 = vpa((JW4'*IC4*JW4),6);
o9 = vpa((m5*Jv5'*Jv5),6);
o10 = vpa((JW5'*IC5*JW5),6);
o11 = vpa ((m6*Jv6'*Jv6),6);
o12 = vpa ((JW6'*IC6*JW6),6);



%%
digits(6)
M = o1+ o2 +o3+ o4 + o5+ o6 + o7+ o8 + o9+ o10 + o11+ o12;
```

```
%% Coriolis and centripetal Forces
q_1 = [Theta1;Theta2;Theta3;Theta4;Theta5;Theta6];


for i = 1:6
    for j=1:6
        for k = 1: 6

            bijk = 0.5*(diff(M(i,j),q_1(k))+diff(M(i,k),q_1(j))-
diff(M(j,k),q_1(i)));
            if j==k
                c(i,j) =  bijk;
            end

            if (j==1 &k>1)
                b1(i,(k-1))= 2*bijk;
            end
            if (j==2 &k>2)
                b2(i,(k-2))= 2*bijk;
            end
            if (j==3 &k>3)
               b3(i,(k-3))= 2*bijk;
            end

            if (j==4 &k>4)
               b4(i,(k-4))= 2*bijk;
            end

            if (j==5 &k>5)
               b5(i,(k-5))= 2*bijk;
            end


        end
    end
end


c= c;
b= [b1 b2 b3 b4 b5];


%% Gravity Matrix
g = [0;0;9.81];
G = -
[Jv1'*g*m1+Jv2'*g*m2+Jv3'*g*m3+Jv4'*g*m4+Jv5'*g*m5+Jv6'*g*m6];
G = simplify(G)
%% trajectory for motor selection
%%  joint 1 trajectory

t0   = 0;  % sec
tf   = 4 ;  % sec
q0   = 0;  % inetial possition
dq0  = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
```

```
qf    = 0;  % final possition
dqf   = 0 ;  % final speed
ddqf = 0 ;  % final accelaration


%% Quintic Polynomial Trajectories

 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];

 q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
 a = inv(A)*q;

 %%

i = 1;
for t = 0:0.01:tf
    q_1(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_1(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_1(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
    time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_1)
title('q1');
ylabel('q');
xlabel('time');
figure
plot(time, dq_1)
title('dq1');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_1)
title('ddq1');
ylabel('ddq');
xlabel('time');

%%  joint 2 trajectory

t0    = 0;  % sec
tf    = 4 ;  % sec
q0    = 0;  % inetial possition
dq0   = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
qf    = pi/2;  % final possition
dqf   = 0 ;  % final speed
ddqf = 0 ;  % final accelaration

%% Quintic Polynomial Trajectories
```

```
 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];

 q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
 a = inv(A)*q;

%%

i = 1;
for t = 0:0.01:tf
    q_2(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_2(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_2(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
   time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_2)
title('q2');
ylabel('q');
xlabel('time');
figure
plot(time, dq_2)
title('dq2');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_2)
title('ddq2');
ylabel('ddq');
xlabel('time');

%%  joint 3 trajectory
t0   = 0;  % sec
tf   = 4 ;  % sec
q0   = 0;  % inetial possition
dq0  = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
qf   = 0;  % final possition
dqf  = 0 ;  % final speed
ddqf = 0 ;  % final accelaration

%% Quintic Polynomial Trajectories

 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];
```

```
q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
a = inv(A)*q;



%%

i = 1;
for t = 0:0.01:tf
    q_3(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_3(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_3(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
   time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_3)
title('q3');
ylabel('q');
xlabel('time');
figure
plot(time, dq_3)
title('dq3');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_3)
title('ddq3');
ylabel('ddq');
xlabel('time');

%%  joint 4 trajectory
t0   = 0;  % sec
tf   = 4 ;  % sec
q0   = pi/2;  % inetial possition
dq0  = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
qf   = pi/2;  % final possition
dqf  = 0 ;  % final speed
ddqf = 0 ;  % final accelaration

%% Quintic Polynomial Trajectories

 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];
```

```
q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
a = inv(A)*q;



%%

i = 1;
for t = 0:0.01:tf
    q_4(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_4(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_4(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
   time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_4)
title('q4');
ylabel('q');
xlabel('time');
figure
plot(time, dq_4)
title('dq4');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_4)
title('ddq4');
ylabel('ddq');
xlabel('time');


%%  joint 5 trajectory
t0   = 0;  % sec
tf   = 4 ;  % sec
q0   = 0;  % inetial possition
dq0  = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
qf   = 0;  % final possition
dqf  = 0 ;  % final speed
ddqf = 0 ;  % final accelaration

%% Quintic Polynomial Trajectories

 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];



q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
a = inv(A)*q;
```

```
%%

i = 1;
for t = 0:0.01:tf
    q_5(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_5(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_5(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
   time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_5)
title('q5');
ylabel('q');
xlabel('time');
figure
plot(time, dq_5)
title('dq5');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_5)
title('ddq5');
ylabel('ddq');
xlabel('time');

%%  joint 6 trajectory
t0   = 0;  % sec
tf   = 4 ;  % sec
q0   = -pi/2;  % inetial possition
dq0  = 0 ;  % inetial speed
ddq0 = 0 ;  % inetial accelaration
qf   = -pi/2;  % final possition
dqf  = 0 ;  % final speed
ddqf = 0 ;  % final accelaration

%% Quintic Polynomial Trajectories

 A = [1 t0 t0^2  t0^3   t0^4   t0^5 ;
      0 1  2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0    2   6*t0  12*t0^2 20*t0^3;
      1 tf tf^2  tf^3   tf^4   tf^5 ;
      0 1  2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0    2   6*tf  12*tf^2 20*tf^3];



q = [ q0 dq0 ddq0 qf dqf ddqf].' ;
a = inv(A)*q;
```

```
%%

i = 1;
for t = 0:0.01:tf
    q_6(i)=a(1)+a(2)*t+a(3)* t^2 + a(4)*t^3+ a(5)*t^4 + a(6)*t^5;
    dq_6(i)=a(2)+2*a(3)* t + 3*a(4)*t^2+ 4*a(5)*t^3 + 5*a(6)*t^4;
    ddq_6(i)=2*a(3) + 6*a(4)*t+ 12*a(5)*t^2 + 20*a(6)*t^3;
   time(i) = t;
    i = i+1;
end
%% plots
figure
plot(time, q_6)
title('q6');
ylabel('q');
xlabel('time');
figure
plot(time, dq_6)
title('dq6');
ylabel('dq');
xlabel('time');
figure
plot(time, ddq_6)
title('ddq6');
ylabel('ddq');
xlabel('time');

%% motors torques profiles
i = 1;
for t = 0:0.01:tf
acc = [ddq_1(i);ddq_2(i);ddq_3(i);ddq_4(i);ddq_5(i);ddq_6(i)];
V_ = [dq_1(i);dq_2(i);dq_3(i);dq_4(i);dq_5(i);dq_6(i)];
V_V = [V_(1)*V_(2);
V_(1)*V_(3);V_(1)*V_(4);V_(1)*V_(5);V_(1)*V_(6);V_(2)*V_(3);V_(2)*
V_(4);V_(2)*V_(5);V_(2)*V_(6);V_(3)*V_(4);V_(3)*V_(5);V_(3)*V_(6);
V_(4)*V_(5);V_(4)*V_(6);V_(5)*V_(6)];


torques= M*acc+c*V_.^2 + b*V_V+ G;


torques_(i,:) = vpa(subs(torques,[Theta1, Theta2,
Theta3,Theta4,Theta5,Theta6],[q_1(i),q_2(i),q_3(i),q_4(i),q_5(i),q
_6(i)]));
i = i+1;

end
max_torque = max(abs (torques_))
rms_torque = rms(torques_)

figure
plot(time,torques_(:,1))
title('Torque q1');
ylabel('Torque ');
xlabel('time');
figure
```

```
plot(time,torques_(:,2))
title('Torque q2');
ylabel('Torque ');
xlabel('time');
figure
plot(time,torques_(:,3))
title('Torque q3');
ylabel('Torque ');
xlabel('time');
figure
plot(time,torques_(:,4))
title('Torque q4');
ylabel('Torque ');
xlabel('time');
figure
plot(time,torques_(:,5))
title('Torque q5');
ylabel('Torque ');
xlabel('time');
figure
plot(time,torques_(:,6))
title('Torque q6');
ylabel('Torque ');
xlabel('time');
```

APPENDIX B: PYTHON SOURCE CODES

```
# Used librarys:
import cv2
import av

from PIL import ImageTk, Image

import os

import time
from datetime import datetime
import pytz

import customtkinter

import numpy as np
import sympy


from sympy import symbols, cos, sin, trigsimp

import math

import matplotlib.pyplot as plt

import serial

# Global variables:
Patient_hand = None
Final_result = None
Patient_file = None
path = None

X = None
Y = None
Z = None

Roll = None
Pitch = None
Yaw = None

X_inCartesian = None
Y_inCartesian = None

time_zone = pytz.timezone('Asia/Jerusalem')

arduino = serial.Serial('COM4', 115200)


# User interface functions:

# Ending function:
def End_fun():
```

```
    image = cv2.imread('N_2020_logo_final.png')
    cv2.namedWindow("N_2020 Software", cv2.WINDOW_FULLSCREEN)
    cv2.imshow("N_2020 Software", image)

    key = cv2.waitKeyEx(5000)
    cv2.destroyAllWindows()

# Time and Data functions:
def Time():
    # Get the current time in your timezone
    now = datetime.now(time_zone)
    current_time = now.strftime("%H:%M")
    Time_label = customtkinter.CTkLabel(master=frame,
text=current_time, font=("Times new roman", 17),
                                        text_color='white')
    Time_label.place(relx=0.1, rely=0.1, anchor="center")
    root.after(1000, Time)


def Date():
    current_date = datetime.now().strftime("%A, %B %Y")
    Time_label = customtkinter.CTkLabel(master=frame,
text=current_date, font=("Times new roman", 17),
                                        text_color='white')
    Time_label.place(relx=0.4, rely=0.1, anchor="center")
    root.after(1000, Time)


# Error functions:
# Naming error function:
def Naming_error(window):
    global Patient_file, Final_result, path
    Alert_image = Image.open("Readflage.png")
    Alert_photo = ImageTk.PhotoImage(Alert_image)

    error_label = customtkinter.CTkLabel(master=frame, text="The
ID has been used, the file will be over written",
                                         font=("Times new roman",
20), text_color='red')
    error_label.place(relx=0.52, rely=0.6, anchor="center")
    error_label.text = ""

    Alert_label = customtkinter.CTkLabel(master=frame, text=" ",
image=Alert_photo)

    Alert_label.place(relx=0.08, rely=0.59, anchor="center")
    window.after(3000, Reset_function, error_label, Alert_label)

    directory = Patient_file

    parent_dir = "C:\\N_2020/"

    path = os.path.join(parent_dir, directory, directory + ".png")
    return
```

```
def show_error_message(window):
    Alert_image = Image.open("Readflage.png")
    Alert_photo = ImageTk.PhotoImage(Alert_image)

    error_label = customtkinter.CTkLabel(master=frame,
text="Please enter your ID numbers only",
                                         font=("Times new roman",
20), text_color='red')
    error_label.place(relx=0.5, rely=0.59, anchor="center")
    error_label.text = ""

    Alert_label = customtkinter.CTkLabel(master=frame, text=" ",
image=Alert_photo)
    Alert_label.place(relx=0.18, rely=0.59, anchor="center")
    window.after(3000, Reset_function, error_label, Alert_label)
    return


def number_function_error(window):
    Alert_image = Image.open("Readflage.png")
    Alert_photo = ImageTk.PhotoImage(Alert_image)

    error_label1 = customtkinter.CTkLabel(master=frame,
text="Invalid ID",
                                          font=("Times new roman",
20), text_color='red')
    error_label1.place(relx=0.5, rely=0.59, anchor="center")
    error_label1.text = ""

    Alert_label = customtkinter.CTkLabel(master=frame, text=" ",
image=Alert_photo)
    Alert_label.place(relx=0.35, rely=0.59, anchor="center")
    window.after(3000, Reset_function, error_label1, Alert_label)
    return



# Error Timer:
def Reset_function(label1, label2):
    label1.destroy()
    label2.destroy()


# Closing function:
def Exit_all():
    root.destroy()
    plt.close()


def close_frame():
    global frame2, root2
    frame2.destroy()
    root2.destroy()
    cv2.destroyAllWindows()
    plt.close()
    End_fun()
```

```
    Orientations_gyro()


def exit_frame():
    global frame2, root2
    frame2.destroy()
    root2.destroy()
    cv2.destroyAllWindows()
    plt.close()
    End_fun()

# Window closer:
def Left_close():
    global frame3, root3
    root3.destroy()
    frame3.destroy()


def Close_leftright():
    root3.destroy()
    frame3.destroy()


# Confirmation function:
def Confirmation_fun():
    global frame2, root2, X_inCartesian
    customtkinter.set_appearance_mode("dark")
    customtkinter.set_default_color_theme("green")
    root2 = customtkinter.CTkToplevel()
    root2.title("Confirmation window")
    root2.grab_set()
    root2.geometry("10,10")

    frame2 = customtkinter.CTkFrame(master=root2)
    frame2.pack(pady=10, padx=10, fill='both')
    Start_button = customtkinter.CTkButton(master=root2,
text="Press to confirm the points",
                                           font=("Times new
roman", 15), command=close_frame)
    Start_button.place(relx=0.5, rely=0.4, anchor="center")
    Start_button = customtkinter.CTkButton(master=root2,
text="Back",
                                           font=("Times new
roman", 15), command=exit_frame)
    Start_button.place(relx=0.5, rely=0.6, anchor="center")

    label = customtkinter.CTkLabel(master=frame,
text=X_inCartesian, font=("Times new roman", 40))
    label.place(relx=0.5, rely=0.32, anchor="center")

    root2.mainloop()


# Data handling function:
# Storing function:
```

```
def storing_fun(event, entry):
    global Patient_file, Final_result, path
    Patient_file = entry.get()

    string_length = len(Patient_file)

    if string_length < 6:
        number_function_error(frame)

    if Patient_file.isdigit() == True and string_length >= 6:

        directory = Patient_file
        parent_dir = "C:\\N_2020/"
        path = os.path.join(parent_dir, directory, directory +
".png")  # specify the file name and extension

        try:
            os.mkdir(os.path.join(parent_dir, directory))
        except FileExistsError:
            Naming_error(frame)
        else:
            print("The created file is: " + str(Patient_file))

    if not Patient_file.isdigit():
        show_error_message(frame)

        return


# Arduino function:
def End_effector_fun():
    with open("C:\\N_2020/Data.txt", "r") as file:
        X_Coordinates = file.readline().strip()
        Y_Coordinates = file.readline().strip()

        data = X_Coordinates + ',' + Y_Coordinates

        arduino.write(data.encode())


# Coordinates generator:
def get_point_coordinates(image):

    fig, ax = plt.subplots(facecolor='gray')
    ax.imshow(image, cmap='gray', aspect='auto', origin='upper',
extent=[0, 1920, 0, 1080])
    ax.set_facecolor('white')
    plt.gca().set_facecolor('white')
    points = []

    def onclick(event):
        global X_inCartesian, Y_inCartesian, Yaw, Z
        nonlocal points
        point = [event.ydata, event.xdata]
        points.append(point)
        ax.plot(point[1], point[0], marker='o', color='red')
```

```
        print(f"Clicked point coordinates: y={point[0]},
x={point[1]} ")

        Spatial_X = point[1]
        Spatial_Y = point[0]

        Real_distance_X = (0.40 * Z) /0.15
        Real_distance_Y = (0.20 * Z) /0.15

        Mapping_x = (Spatial_X / 1920)
        Mapping_y = (Spatial_Y / 1080)

        X_inCartesian = Mapping_x * Real_distance_X * cos(Pitch)
        Y_inCartesian = Mapping_y * Real_distance_Y * cos(Pitch)

        Confirmation_fun()

    cid = fig.canvas.mpl_connect('button_press_event', onclick)
    plt.show()
    return points


# Image Processing Function:
def Image_processing_fun():
    if Patient_file.isdigit():
        counter = 0
        Start_time = time.time()
        global Patient_hand, Final_result, X, Y
        X = 1280
        Y = 720

        rtsp_url = "rtsp://172.19.3.134:8000/"
        container = av.open(rtsp_url)

        for frame in container.decode(video=0):
            Input_feed = frame.to_ndarray(format='bgr24')

            Input = cv2.cvtColor(Input_feed, cv2.COLOR_BGR2GRAY)
            Start_filtering = time.time()

            ksize = (3, 3)
            sigma = 1.0
            filtered_img = cv2.GaussianBlur(Input, ksize, sigma)

            Low_Intensity = 0
            High_Intensity = 215

            replace_color = 255

            filtered_img[(filtered_img < Low_Intensity) |
(filtered_img > High_Intensity)] = replace_color
            Finish_filtering = time.time()

            Start_thresolding = time.time()
            sigma_thres = 3.5 * sigma
```

```
            Thresolded_image = cv2.adaptiveThreshold(filtered_img,
255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,
                                                     17,
                                                     sigma_thres)
            Finish_thresolding = time.time()

            Start_Finding_Contours = time.time()
            _, contours, hierarchy =
cv2.findContours(Thresolded_image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            contour_img = np.zeros_like(Input)
            cv2.drawContours(contour_img, contours, -1, (255, 255,
255), cv2.FILLED)
            result = cv2.absdiff(Thresolded_image, contour_img)

            Finish_Finding_Contours = time.time()

            Patient_hand = cv2.bitwise_not(result)

            connected_componnets_start = time.time()
            kernel = np.ones((3, 3), np.uint8)

            binary_image = cv2.morphologyEx(Patient_hand,
cv2.MORPH_OPEN, kernel, iterations=1)
            binary_image = cv2.morphologyEx(Patient_hand,
cv2.MORPH_CLOSE, kernel, iterations=1)
            num_labels, labels, stats, centroids =
cv2.connectedComponentsWithStats(binary_image)

            for i in range(1, num_labels):
                component_image = np.zeros_like(binary_image)
                component_image[labels == i] = 255

            connected_componnets_end = time.time()

            cv2.imshow('Original Image', Input)
            cv2.imshow('Filtered Image', filtered_img)
            cv2.imshow('Thresholded Image', Thresolded_image)
            cv2.imshow('Pre-final Result', Patient_hand)
            cv2.imshow('Contours', contour_img)
            cv2.imshow('Bitwise', result)
            cv2.imshow('Result', component_image)

            cv2.imwrite(path, component_image)

            Final_time = time.time()

            Filtering = Finish_filtering - Start_filtering
            Thresolding = Finish_thresolding - Start_thresolding
            Findig_Countors = Finish_Finding_Contours -
Start_Finding_Contours
            connected_componnets = connected_componnets_end -
connected_componnets_start
            Total_time = Final_time - Start_time

            print("Filtering time:", Filtering)
```

```
            print("Thresolding time:", Thresolding)
            print("Countors finding time:", Findig_Countors)
            print("Connected components finding time:",
connected_componnets)
            print("Total time:", Total_time)

            if counter == 0:
                get_point_coordinates(component_image)
                counter += 1
            else:
                pass
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    else:
        pass


def Orientations_gyro():
    counter = 0
    counter2 = 0
    global Roll, Pitch, Yaw, Z

    while True:
        data = arduino.readline().decode().strip()
        counter +=1
        if counter > 10:
            Roll, Pitch, Yaw, Z = data.split(',')

            print("Roll:", Roll, "degrees")
            print("Pitch:", Pitch, "degrees")
            print("Yaw:", Yaw, "degrees")
            print("High:", Z, "Meter")

        if counter2 == 2:
            arduino.close()
            Angels_calculator()
            break


def Angels_calculator():
    Start_time = time.time()
    global X_inCartesian, Y_inCartesian, Roll, Pitch, Yaw

    # Symbolic variables
    Theta1, Theta2, Theta3, Theta4, Theta5, Theta6, m6, n6 =
symbols('Theta1 Theta2 Theta3 Theta4 Theta5 Theta6 m6 n6')
    nx, ny, nz, ox, oy, oz, ax, ay, az, px, py, pz = symbols('nx
ny nz ox oy oz ax ay az px py pz')

    a1 = 0.012
    d4 = 0.013
    L3 = 0.12
    L2 = 0.15
    d5 = 0.065
    L6 = 0.095
```

```
    nx, ox, ax, px = symbols('nx ox ax px')
    ny, oy, ay, py = symbols('ny oy ay py')
    nz, oz, az, pz = symbols('nz oz az pz')

    T = sympy.Matrix([
        [nx, ox, ax, px],
        [ny, oy, ay, py],
        [nz, oz, az, pz],
        [0, 0, 0, 1]
    ])

    # Given values
    ax = -0.2939
    ay = 0.4045
    az = 0.866
    ox = -0.03633
    oy = 0.9007
    oz = -0.433
    nx = 0.9551
    ny = 0.1587
    nz = 0.25
    px = 0.1
    py = 0.02439
    pz = 0.3189

    m = L6 * ny - py
    n = L6 * nx - px

    # Theta 1
    th1_2 = sympy.atan2(m, n) - sympy.atan2(d4, -sympy.sqrt(m ** 2
+ n ** 2 - d4 ** 2))

    th1_2_value = th1_2.evalf()

    # Theta 1 in rad
    print("th1(2):", th1_2_value)
    print()

    # Theta 5

    th5 = sympy.acos(ax * sympy.sin(th1_2) - ay *
sympy.cos(th1_2))

    th5_value = th5.evalf()

    # Theta 5 in rad

    print("th5:", th5_value)

    print()

    # Theta 6
    th1 = sympy.symbols('th1:3')  # Array of theta_1_1 and
theta_1_2
```

```
    m6_2 = nx * sympy.sin(th1_2) - ny * sympy.cos(th1_2)
    n6_2 = -(ox * sympy.sin(th1_2) - oy * sympy.cos(th1_2))
    print(m6_2)
    print(n6_2)

    th6_1 = -1 * (sympy.atan2(m6_2, n6_2) -
sympy.atan2(sympy.sin(th5), -1 *

sympy.sqrt(

round(m6_2 ** 2 + n6_2 ** 2 - sympy.sin(th5) ** 2, 3))))

    print(round(m6_2 ** 2 + n6_2 ** 2 - sympy.sin(th5) ** 2, 3))
    th6_1_value = th6_1.evalf()

    # Theta 6 in rad
    print("th6(1):", th6_1_value)
    print()

    # Theta star_ theta 2 +theta 3 + theta 4

    s234 = -(sympy.sin(th6_1) * (nx * sympy.cos(th1_2) + ny *
sympy.sin(th1_2)) + sympy.cos(th6_1) * (
                ox * sympy.cos(th1_2) + oy * sympy.sin(th1_2)))

    c234 = oz * sympy.cos(th6_1) + nz * sympy.sin(th6_1)
    th234 = sympy.atan2(s234, c234)

    # Theta 3

    m3_1 = (px - L6 * nx) * sympy.cos(th1_2) + (py - L6 * ny) *
sympy.sin(th1_2) - a1 + d5 * sympy.cos(th1_2) * (
            ox * sympy.cos(th6_1) + nx * sin(th6_1)) + d5 *
sin(th1_2) * (
                    oy * cos(th6_1) + ny * sin(th6_1))

    n3_1 = pz - L6 * nz + d5 * (oz * sympy.cos(th6_1) + nz *
sympy.sin(th6_1))

    if ((m3_1 ** 2 + n3_1 ** 2 - L2 ** 2 - L3 ** 2) / (2 * L3 *
L2) >= 0):
        th3 = sympy.acos((m3_1 ** 2 + n3_1 ** 2 - L2 ** 2 - L3 **
2) / (2 * L3 * L2))
    else:
        th3 = -1 * sympy.acos(round((m3_1 ** 2 + n3_1 ** 2 - L2 **
2 - L3 ** 2) / (2 * L3 * L2), 3))

    s2 = ((L3 * sympy.cos(th3) + L2) * n3_1 - (L3 * sympy.sin(th3)
* m3_1)) / (
            L2 ** 2 + L3 ** 2 + (2 * L2 * L3 * sympy.cos(th3)))
    c2 = (m3_1 + (L3 * sympy.sin(th3) * s2)) / (L3 *
sympy.cos(th3) + L2)
    th2 = sympy.atan2(s2, c2)
    th4 = th234 - th3 - th2

    th3_value = th3.evalf()
```

```
    print("th3_value:", th3_value)
    print()
    th2_value = th2.evalf()
    print("th2_value:", th2_value)
    print()

    print('th4:', th4)


# User Interface:
customtkinter.set_appearance_mode("dark")
customtkinter.set_default_color_theme("green")
root = customtkinter.CTk()
root.geometry("1920,1080")
root.title('N_2020 Software')

# Get screen dimensions
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

# Load image
image = Image.open("N_2020_bg.png")
image = image.resize((1920, 1080), Image.ANTIALIAS)
photo = ImageTk.PhotoImage(image)

# Create label widget and set image
label = customtkinter.CTkLabel(master=root, image=photo,
width=screen_width, height=screen_height)
label.pack(fill=customtkinter.BOTH, expand=True)

frame = customtkinter.CTkFrame(master=root, width=500, height=400)
frame.place(relx=0.5, rely=0.45, anchor="center")

label = customtkinter.CTkLabel(master=frame, text="System login",
font=("Times new roman", 40))
label.place(relx=0.5, rely=0.32, anchor="center")

entery_1 = customtkinter.CTkEntry(master=frame,
placeholder_text="Patient ID", font=("Times new roman", 20),
width=300,
                                  justify="center")
entery_1.place(relx=0.5, rely=0.49, anchor="center")
entery_1.bind("<Return>", lambda event: storing_fun(event,
entery_1))

Start_button = customtkinter.CTkButton(master=root, text="Press to
start", font=("Times new roman", 15),

command=Image_processing_fun)
Start_button.place(relx=0.5, rely=0.55, anchor="center")

close_button = customtkinter.CTkButton(master=root, text="Exit",
font=("Times new roman", 15), command=Exit_all)
close_button.place(relx=0.5, rely=0.6, anchor="center")

Time()
```

```
Date()
root.mainloop()
```

APPENDIX C: ARDUINO SOURCE CODE

```
 #include <MPU6050_tockn.h>
#include <Wire.h>

int IN_A0 = A0;
int trigger = 8;
int echo = 9;


MPU6050 mpu6050(Wire);

float Pitch = 0;
float  Roll = 0;
float  Yaw = 0;

void setup() {

  pinMode (IN_A0, INPUT);
  pinMode (echo, INPUT);
  pinMode (trigger, OUTPUT);


  Serial.begin(115200);

  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
  Pitch = ReadAngle_Y();
  Roll = ReadAngle_Z();
  Yaw = ReadAngle_X();


}

void loop() {
  static unsigned long Start_time = 0;
  unsigned long T = millis();
  unsigned long Real_time = T - Start_time;
  unsigned long Period = 3000;
  long duration, Z;
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  duration = pulseIn(echo, HIGH);



  Pitch = ReadAngle_Y();
  Roll = ReadAngle_Z();
  Yaw = ReadAngle_X();
  Z =  microsecondsToCentimeters(duration)

  if (Real_time >= Period) {
```

```
    if (isnan(Pitch) || isnan(Roll) || isnan(Yaw)) || isnan(Z)) {
      Serial.print("NAN condition.. ");

    }
    else {

      Serial.print(Pitch);
      Serial.print(",");
      Serial.print(Roll);
      Serial.print(",");
      Serial.println(Yaw);
      Serial.print(",");
      Serial.println(Z);

    }
  }
  delay(500);
}




float ReadAngle_Y() {

  mpu6050.update();
  return mpu6050.getAngleY();

}
float ReadAngle_X() {

  mpu6050.update();
  return mpu6050.getAngleX();

}
float ReadAngle_Z() {

  mpu6050.update();
  return mpu6050.getAngleZ();

}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```