Beni-Suef University

Faculty of computer and Artificial Intelligence

"Vein Detector"

Under Supervision:

Dr. Amr Abdelaziz

Kareem Mohamed **Karen Wafik**
Martina Mohsen **Fadia Magdy**

2022/2023

Beni-Suef University

Faculty of computer and Artificial Intelligence

Medical Informatics

Graduation Project

Vein Detector

Under Supervision:

Dr. Amr Abdelaziz

Assistant:

Eng\Abanop Gerges

2022/2023

Acknowledgement

First and foremost, we have to thank our research supervisor, **Dr. Amr Abdelaziz**, Professor at Faculty of computer and Artificial Intelligence at Beni-Suef University. Without his assistance and dedicated involvement in every step throughout the process, this project would have never been accomplished, so we would like to express our deepest appreciation to him.

We would also like to show gratitude to **Eng\Abanop Gerges** for his support and understanding over these past four years.

Finally, we would like to thank the dean of faculty of computer and Artificial Intelligence of Beni-Suef University for his permission to use all the available means that help us to progress our project.

Abstract:

One of the most common procedures used when making the initial preparations for the initial operations is the procedures for drawing blood or injecting medications, where the nurse is required to access the patient's veins. And therefore, it is very important to reduce as much as possible the pain resulting from the useless investigation of the vein, as these pains result mainly from repeated pricking operations. However, certain problems with access to veins, especially in children and obese patients, make it very difficult to achieve successful access within a short time. Where a camera takes pictures of the infrared illuminated veins and a device re-project the image to locate on the same location of the skin after applying the processing algorithm and image optimization algorithms so as to achieve a real-time system. Each vein in the image of the vein detector appears with different contrast relative to the skin that was taken as a reference, ie as the background image, which can be of great value affecting the properties of the vein during the investigation process.

The main objective in this project is to quantitatively determine the effect of the physical properties of veins such as depth and diameter on their visibility in the device image. Comparing infrared imaging technology with imaging technology spectrophotometry, and the results of this study can be interpreted to understand the biological phenomena that affect the image quality of the Vein detector

Through the expansion of this project, significant progress can be made in hardware or software, which is particularly beneficial for doctors and nurses, and which ultimately leads to relieving patients' pain by achieving quick and comfortable access to the intended blood vessels.

Table of contents

Acknowledgement……..……I
Abstract.....I.I
Chapter One: Introduction
1.1 Introduction.....2
Chapter Two: Literature Review
2.1 Veins…….....6
2.2 Vein detector.....6
Chapter Three: Requirements, Diagrams
3.1Requirments…….....11
3.2 Diagrams.....12
Chapter Four: Materials and Methods
4.1 Materials and Methods.....16
4.2 Raspberry pi 4 model B……..……..17
4.3 SD card…….....19
4.4 Raspberrypi camera V2…….....20
4.5 Raspberry Pi Professional Camera (Night Vision - Adjustable Focus)……..……..…..22
4.6 Raspberry pi official USB-C Power Supply ……....26
4.7 Hardware components images……..27
Chapter Five: Software
5.1 Open CV-Python…….....29
5.2 Step 1: Set Up RaspberryPi.....29
5.3 Step 2: enable the camera…….....32
5.4 Step 3: Installing OpenCV ……....34
5.5 Step 4: install pi camera module……34
5.6 Step 5: How to capture a frame from real-time camera video using OpenCV…….……..41
5.7 Step 6: analysis and implementation of vein image processing……47
Chapter Six: Results
6.1 Software results……55
Chapter Seven: Appendices
7.1 Code 1……...……..…59
7.2 Code 2……61
Chapter Eight: Conclusions
8.1 Conclusion……..…...66

References……..63

List of Figures

Figure [2.1] main veins in the arm …….....6
Figure [2.2] hemoglobin and deoxyhemoglobin.....8
Figure [2.3] Light penetration into the skin at different wavelengths…….....8
Figure [3.1] Requirements…….....11
Figure [3.2] Use case diagram……12
Figure [3.3] Sequence diagram……13
Figure [3.4] steps of software……..13
Figure [3.5] steps of image processing……14
Figure [4.1] Material and methods…….....16
Figure [4.2] Raspberry pi 4 model B……...……..17
Figure [4.3] Physical_specifications of raspberrypi…….....19
Figure [4.4] SD card……19
Figure [4.5] Raspberrypi camera V2……..20
Figure [4.6] physical of Raspberrypi camera V2……21
Figure [4.7] Raspberry Pi Professional Camera (Night Vision - Adjustable Focus)……..22
Figure [4.8] Installation of infrared led……...24
Figure [4.9] physical of IR camera……..25
Figure [4.10] Raspberry pi official USB-C Power Supply …….……26
Figure [4.11] Hardware components images……...27
Figure [5.1] Set Up RaspberryPi.....30
Figure [5.2] Step to get space……..31
Figure [5.3] Steps to enable the camera……..32
Figure [5.4] Example……...36
Figure [5.5] the dimension of this matrix……...39
Figure [5.6] Ex: Convert color image to black and white …….....40
Figure [5.7] Steps of Image Processing……...47
Figure [5.8] Frame……...48
Figure [5.9] ROI…….....48
Figure [5.10] grey image……49
Figure [5.11] HBE Image……49
Figure [5.12] CLAHE Image……...49
Figure [5.13] Adaptive threshold……50
Figure [5.14] Opening, Closing Image……....51
Figure [5.15] Trackbar…….....52
Figure [6.1] Software result using Raspberrypi camera V2…….....55
Figure [6.1] Software result using IR camera……..…56
Figure [7.1] Code 1……..59
Figure [7.2] Code 2……..61

Chapter One

Introduction

1.1 Introduction

Recently, Near-Infrared (750–1400 nm) Imaging (NIR) has significantly attracted the attention of many medical scientists and practitioners as a non-invasive method of acquiring images of veins near the surface of skin efficiently. It has been scientifically proven to be useful assistance in phlebotomy procedure performed on patients and a remarkable tool in medical diagnostic. Hence, this device aids more support in the vein tracking process of the phlebotomists because vein puncture is an essential part of clinical testing, donation, or medical research.

Why is this work important?

In many medical situations, diagnosis and treatment of medical conditions require blood draw, drug and fluid administration which can be achieved through proper venous access in the patients. However, problems involved in accessing veins in patients whose veins are not obviously found make it difficult to access the veins directly. Among the patients, many are obese, geriatric, and pediatric, their veins are not palpable or visible for the needle stick. The application of near-infrared light (NIR) is an effective way to help locate the vein, since the light can penetrate up to several centimeters into tissues and be absorbed by the vein, which enables discrimination between blood vessels and surrounding tissues. Based on the theory, several devices have been developed recently to aid physicians, phlebotomists, and surgeons in locating veins to avoid unnecessary sticks. The approach in this paper is proposed to solve the problem of vein display and it is easy to accomplish at the same time. There is only an IR light source, an IR camera or raspberrypi camera V2. The original image is captured by an IR or V2 camera first, then we can get the enhanced vein image by image processing

Difficult things facing patients, doctors and nurses:

Locating the veins is very difficult either by sight or by touch. The fatty tissue often grows to take the shape and appearance of a vein, causing an unsuccessful blood draw in that area. This pricking in the wrong place can be prevented and avoided if the nurse or phlebotomist is trained to differentiate between the veins and the surrounding adipose tissue. Geriatric patients have weak veins that are easy to collapse due to their loss of elasticity, while pediatric patients have veins that are tight, but delicate and of small size. Locating the vein in children is a major challenge because most children are uncooperative with the doctor and face the needle pricks with great nervousness. Psychological restriction and anxiety lead to high blood pressure in patients and thus narrowing of the veins. These narrowed veins are an important problem, especially in children whose veins are of small size in the normal state. And the repeated pricking of the vein leads to its harm, so it is necessary to know the veins that are exposed to several punctures in order to avoid them. One of the main reasons for the failure of attempts to puncture the veins, as the vein moves away from the place of the insertion needle and leads to a useless puncture.

We can consider that the main reasons for the failure of venous positioning in patients are:

1.body mass.

2. Properties of vein 1D.

3. The structure of the site.

4. Fearlessness.

5. Patient anxiety. In addition to factors related to the patient himself. The phlebotomy nurse or phlebotomist plays a vital role in successfully locating the vein.

How does the vein detector work?

Vein Detector that shows the map of the veins and allows to find the required vein effortlessly, It uses infrared technology, which turns the color of blood into a dark color, while the color of the rest of the body's tissues is light. For example, the light emitted by the device is directed to the hand in the area from which the blood sample is taken, and the device will display the map of green veins in the desired area by the projector, and allow the vein to be punctured successfully the first time.

Chapter Two

Literature Review

2.1 Veins

They are flexible blood channels that transport oxygen-poor blood and carbon dioxide-laden blood from different parts of the body to the right atrium in the heart. The vein wall consists of three layers like the artery wall, but it is less durable and less thick than the layers of the artery wall. The color of the veins is bluish red.

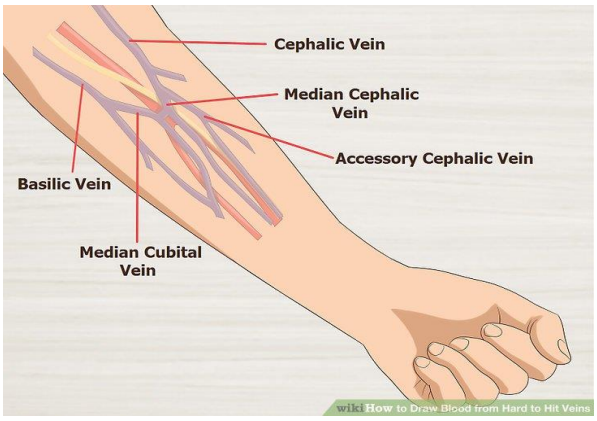


Fig [2.1] main veins in the arm

2.2 Vein detector

Vein detector: is an infrared imaging device for contrast enhancement of subcutaneous veins (in the basal layer) designed for the clinical benefit of physicists and phlebotomists. The vein detector system allows displaying contrast-enhanced images of the venous system in the human body in real time. In our project, the focus was on building this system, taking into account the real-time display due to the material cost and achieving its practical goal without the commercial goal, as the images were displayed on computer screen. Where the veins appear dark in relation to the background light in the image. The images help physicists find the location and direction of the vein. Clinical studies have been carried out on pediatric patients and it has been proven that the device it can be a clinical tool for phlebotomists and patients. Since the device is based on the interaction of light with the hemoglobin in the blood, it is necessary for us to understand behavior of light in the layers of the skin and within the veins.

Optical properties of the skin:

Studying the phenomenon of light transmission in tissues will give us a better understanding of the working mechanism of the vein detector system

The scattering of light within human tissues is that the beam of light falling on the skin undergoes absorption, scattering and reflection due to different optical properties of tissue layers at different depths. The properties of light diffusion differ from one layer to another within the skin. The reflection of light at the surface of the skin is called specular reflection. This reflection blocks the process of light diffusion into the tissues, and thus can cause the image of the skin and veins to glow. |

We will consider the skin as a model that contains three layers: the epidermis, the dermis, and the base. Part of the light is absorbed by the epidermis layer, and the light is transmitted through it to the tissue beneath it, after it is scattered. Most of the scattering occurs within the dermis before it spreads to the base layer while some of the light is absorbed. The fat scatters a large part of the light and absorbs a small part. Part of the light that reaches the blood is absorbed into the vessels by the hemoglobin present in them. While a large part of it is scattered in the direction of diffusion, due to the huge size of red blood cells.

The light absorbance is different as shown in Figure 2.2 Both type's exhibit the same absorbance properties up to a wavelength of 600 nm

While the light absorption in veins is higher than in arteries between wavelengths 600 nm to 800 nm, it decreases

The curve is accelerating for deoxyhemoglobin while rising slightly and then decreasing for hemoglobin.

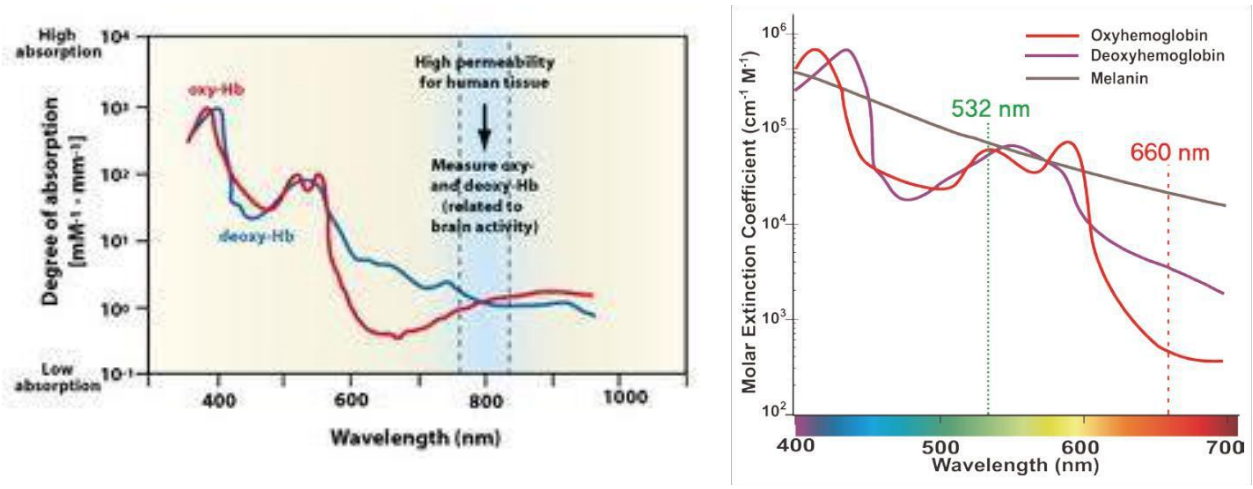


Fig [2.2] shows the absorbance properties of both hemoglobin and deoxyhemoglobin depending on the wavelength of light falling on the skin.

The different wavelengths reach different depths during their transmission through the skin, as shown in Fig [2.2]. The beams in the figure indicate the extent to which light is transmitted in all layers of the skin at different wavelengths. The visible wavelengths range from in all layers of the skin at different wavelengths, visible wavelengths range from 400 nm to 700 nm, while infrared wavelengths range from 700 nm to 10nm. Wavelengths 300 and 400 nm reach only the epidermis and certain sections of the image, which do not contain any veins. Near-infrared light with wavelengths between 700 nm and 1000 nm is less absorbed by tissues and reaches the blood vessels in the base layer.

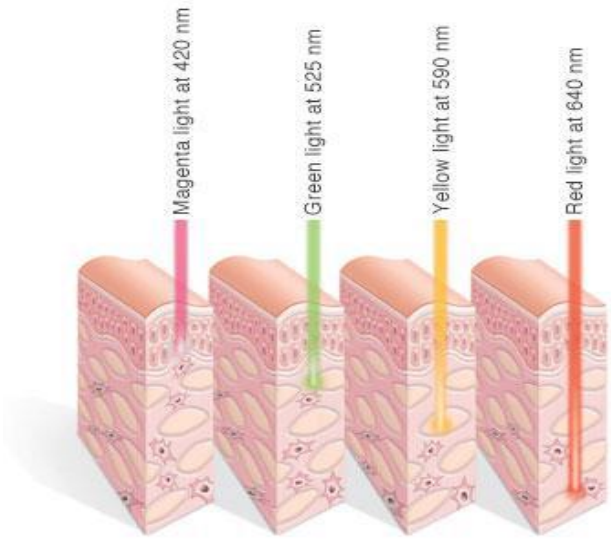


Fig [2.3] Light penetration into the skin at different wavelengths

Our device employs this phenomenon to display veins, which cannot be seen by visible light that is used in spectroscopic imaging, as we will compare its research results with these results. However, the strength of the spectroscopic vein identification method lies in the image processing technology used, which gave satisfactory results.

Chapter Three

Requirements, Diagrams

3.1 Requirements

VEIN DETECTOR PROJECT

ID	Requirements	Date	Related Features	Related Stakeholders	Type (Functional / Non-Functional
1	Accuracy	10 th Dec, 2022	Vein detection	• Healthcare professional • patients	Functional
2	Safety	10 th Dec, 2022	Measures to prevent injury or discomfort	• Healthcare professional • patients	Non-Functional
3	portability	10 th Dec, 2022	Lightweight, compact design	• Healthcare professionals	Non-Functional
4	Imaging Technology	12 th Dec, 2022	Infrared or near-infrared imaging	• Healthcare professional. • patients.	Functional
5	Ease of Use	12 th Dec, 2022	Intuitive controls, simple interface	• Healthcare professionals	Non-Functional
6	Real-time Imaging	12 th Dec, 2022	Live visualization of veins	• Healthcare professionals	Functional
7	Power source	12 th Dec, 2022	Extended power source	• Healthcare professionals	Functional
8	Durability	12 th Dec, 2022	Robust construction, resistance to damage	• Healthcare professionals	Non-Functional
9	Cost-Effectiveness	12 th Dec, 2022	Affordable pricing	• Healthcare facilities	Non-Functional
10	User feedback	25 th Dec, 2022	Satisfaction of the user	• Healthcare professional • patients	Non-Functional
11	Image processing software	25 th Dec, 2022	Using enhancement filters for good result	• Healthcare professionals	Non-Functional
12	A lens and camera system	25 th Dec, 2022	IR night vision camera and changeable lens	• Healthcare professionals	Functional

Fig [3.1] requirements

3.2 Diagrams

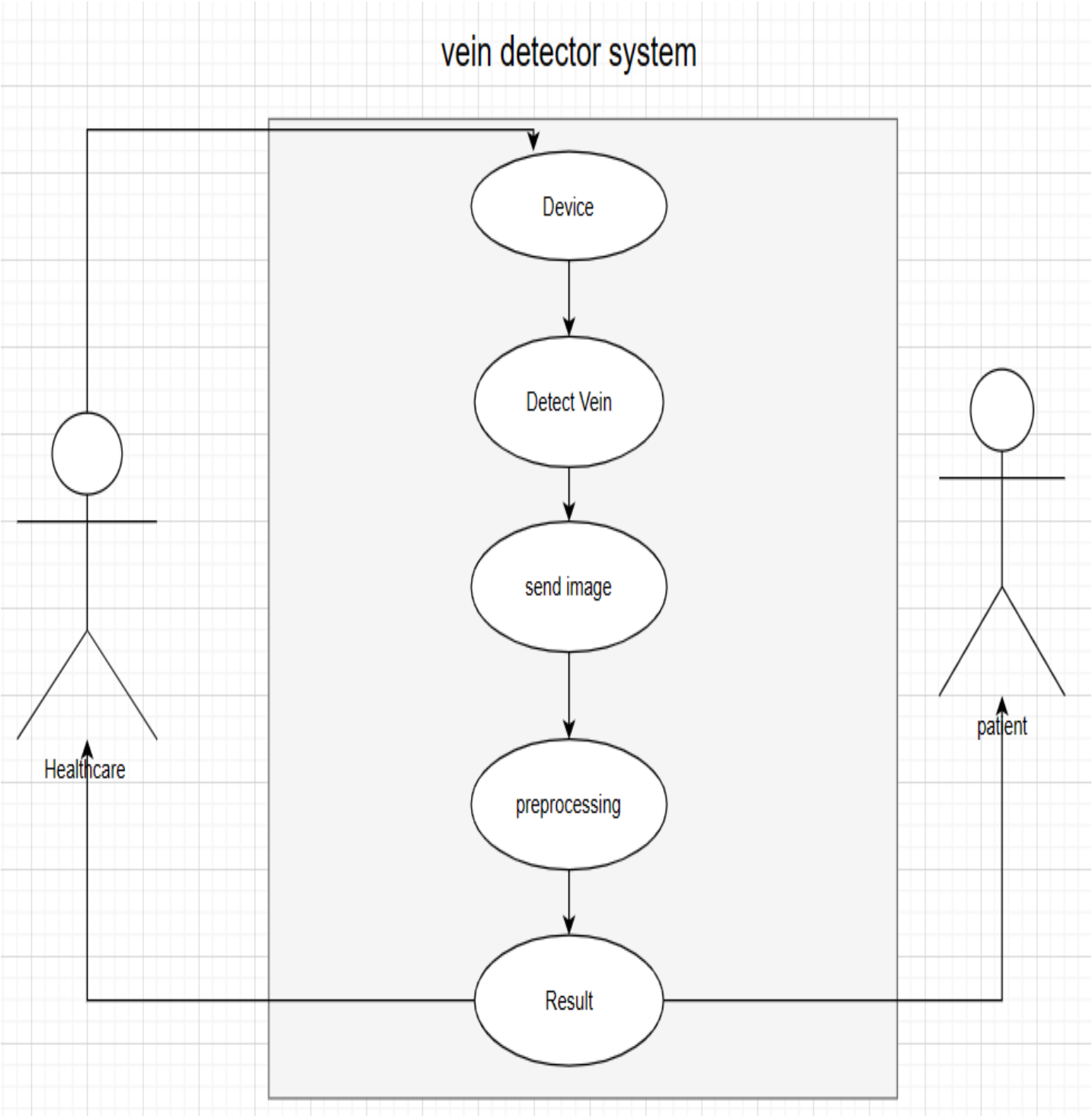


Fig [3.2] Use case diagram

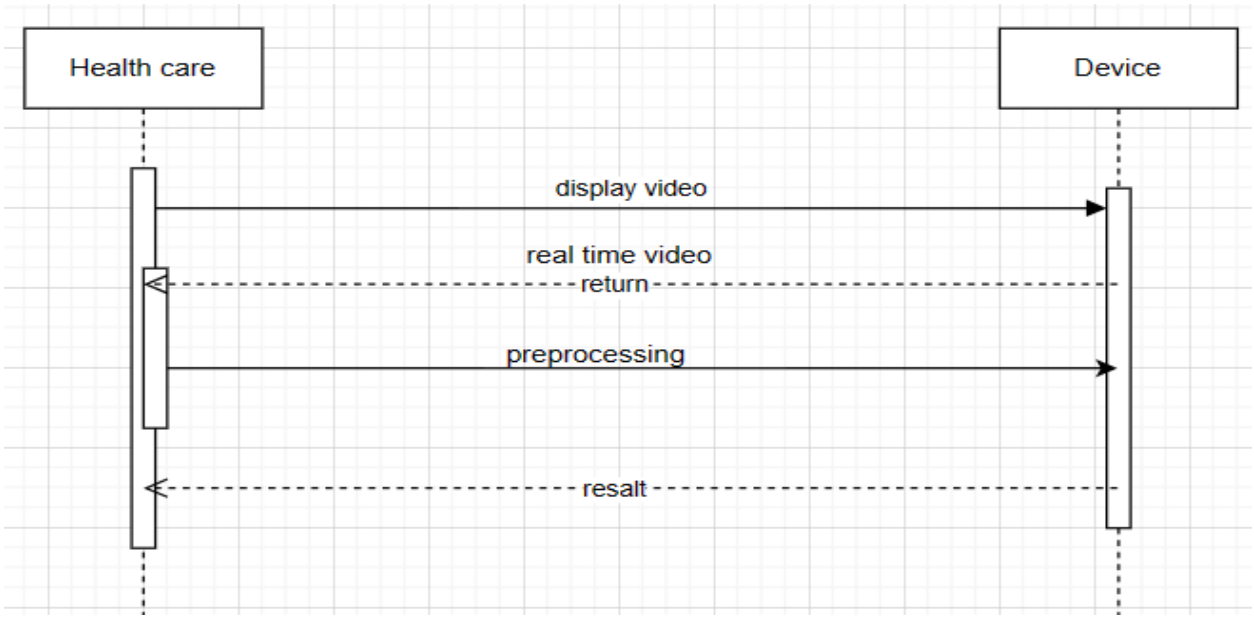


Fig [3.3] Sequence diagram

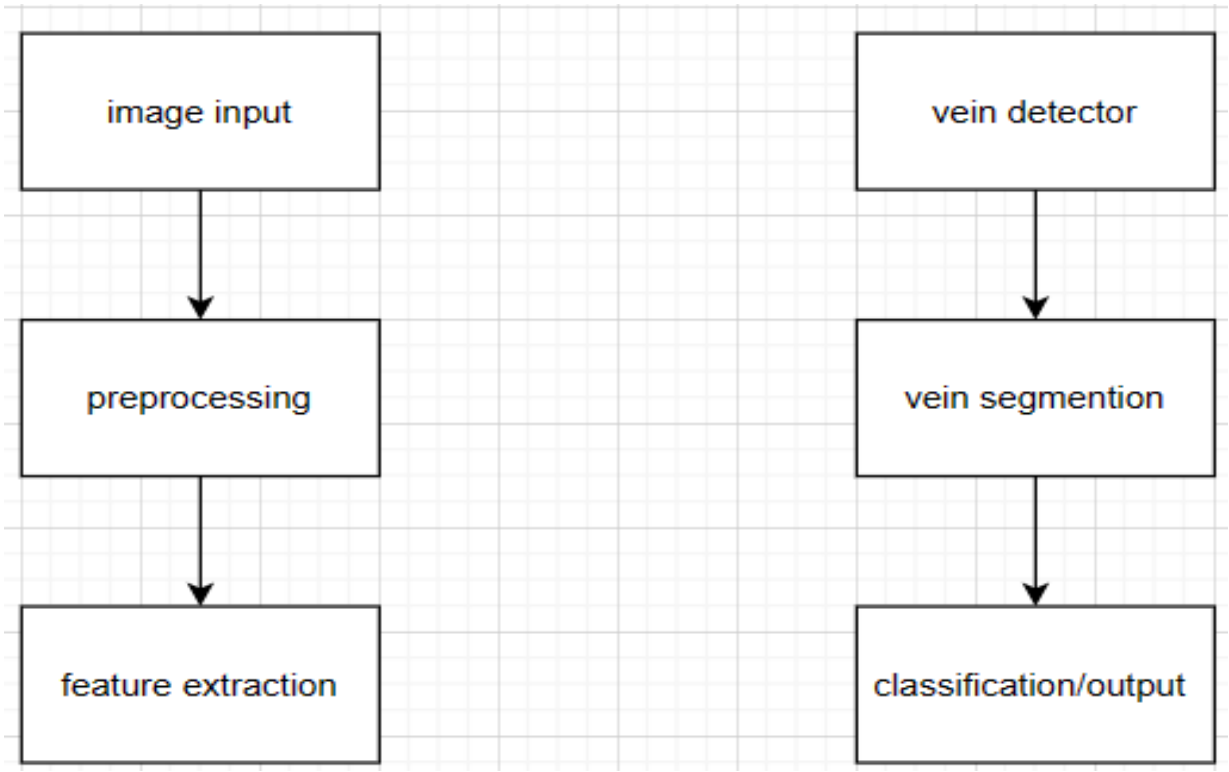


Fig [3.4] steps of software

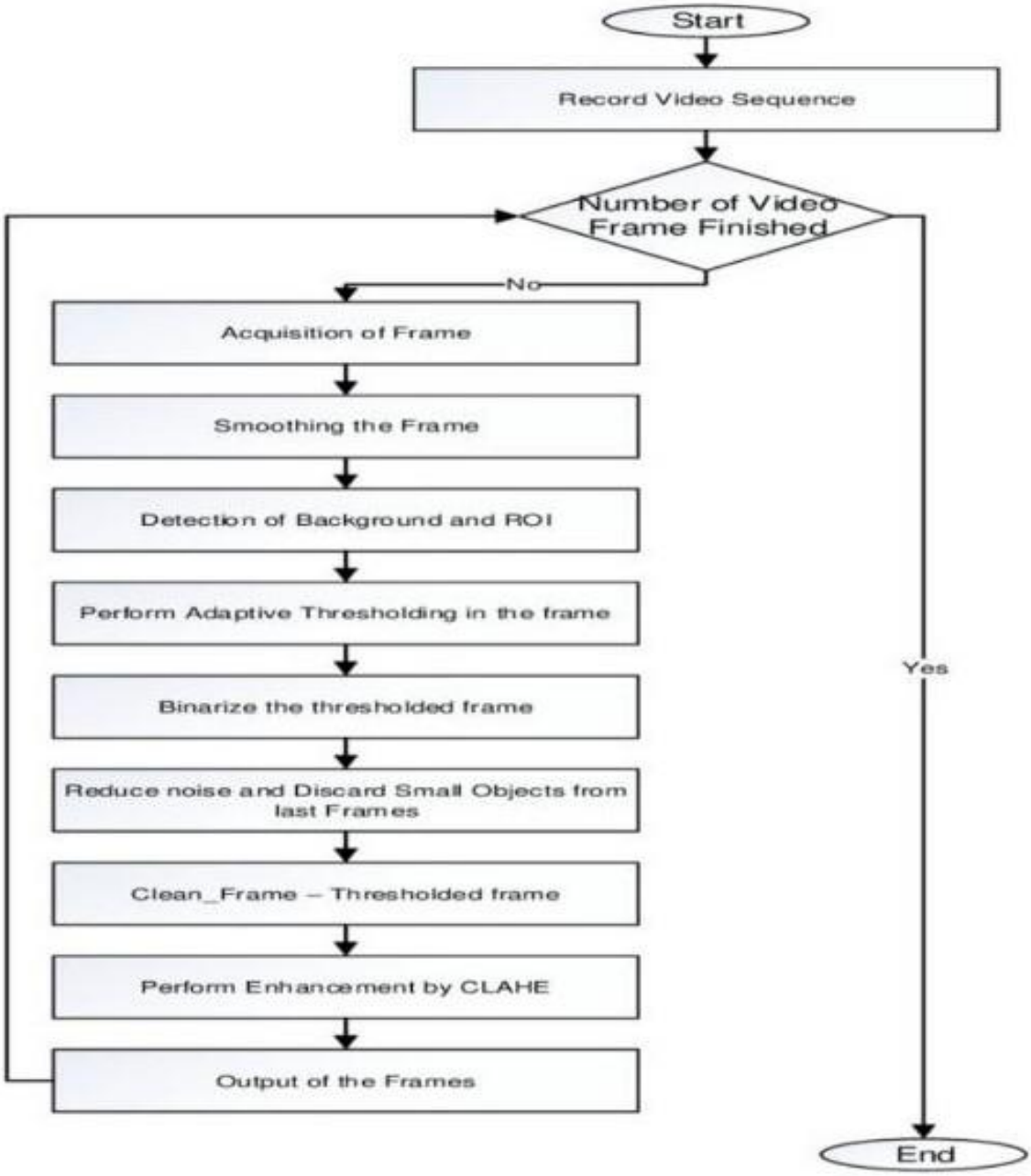


Fig [3.5] steps of image processing

Chapter Four

Materials and Methods

4.1 Materials and Methods

The method of vein imaging, which relies on the variation of light absorption parameters between venous blood and surrounding tissue to capture the image of the venous system non-invasively.

Together with some digital image enhancement techniques a system is designed with a NIR light source, IR or V2 camera to observe images, Raspberry Pi 4 Model B to receive the image captured by the IR or V2 camera for performing image processing and segmenting procedure and sends them to the computer screen.

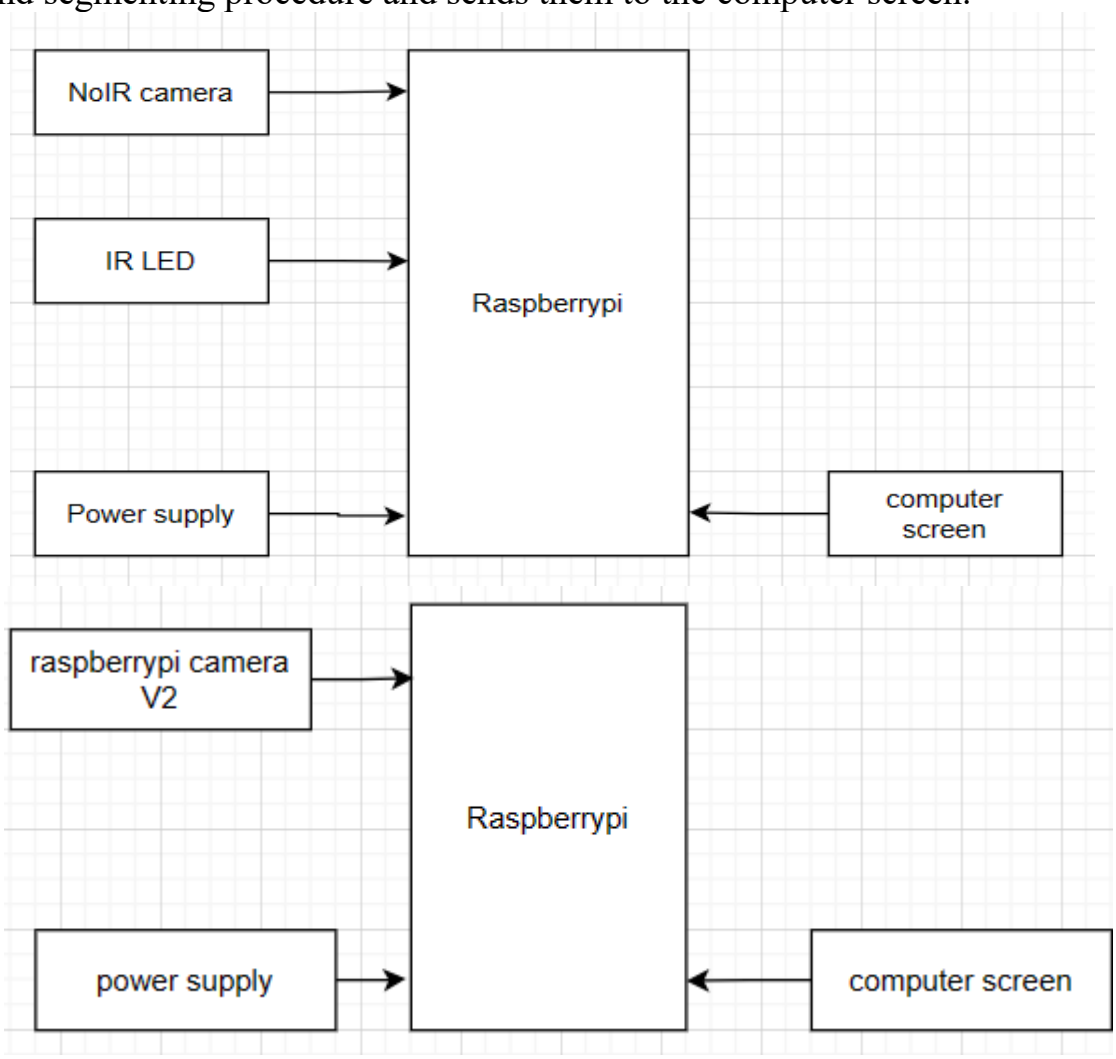


Fig [4.1] Material and methods

4.2 Raspberry pi 4 model B

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries. The original model became more popular than anticipated, selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of the HDMI and USB standards.

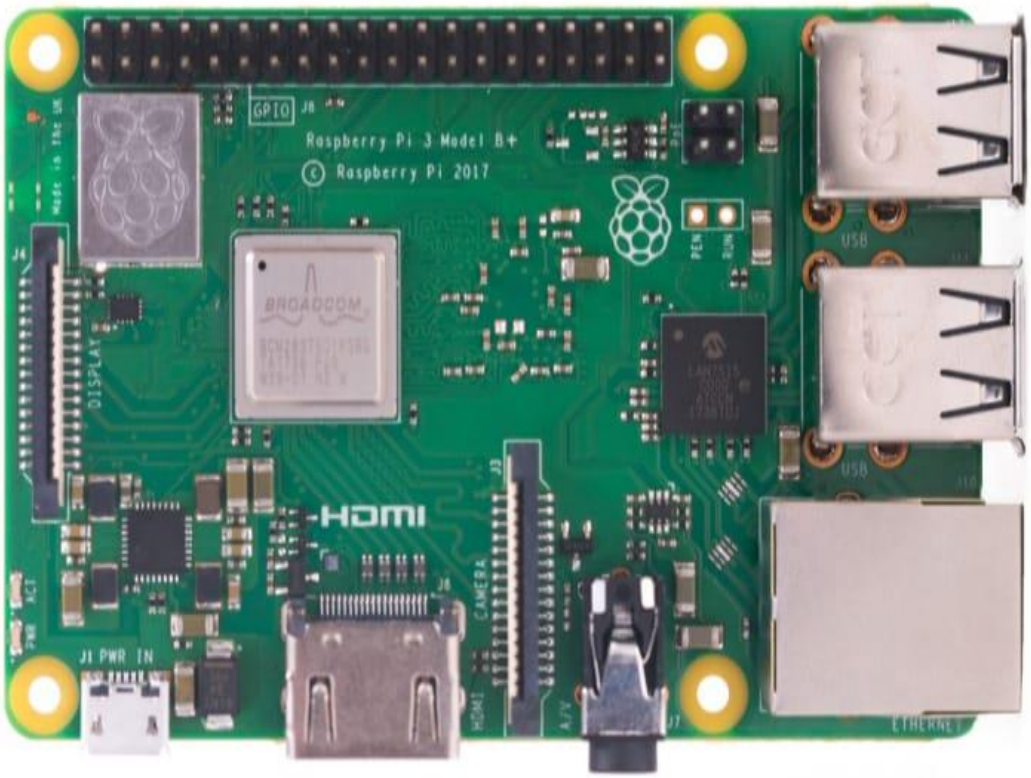


Fig [4.2] Raspberry pi 4 model B

Specifications:

Processor: Broadcom the BCM2711, 64-bit quad-core cortex-A72 processor @ 1.5GHz

Memory: 4GB LPDDR4 RAM

Connectivity:

- 802.11.b/g/n/ac wireless
- Bluetooth 5.0
- Gigabit Ethernet port
- 2 USB 2.0 ports
- 2 USB 3.0 ports
- PoE-capable (requires PoE HAT,sold separately)
- 5V/3A USB-C power supply required (sold separately)

Access: Extended 40-pin GPIO header.

Video & sound: 1 × full size HDMI
MIPI DSI display port
MIPI CSI camera port
4 pole stereo output and composite video port

SD card support: Micro SD format for loading operating system and data storage

Input power:

- 5V/3A DC via micro USB connector
- 5V DC via GPIO header
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)

Environment: Operating temperature, 0–50°C

Physical specifications:

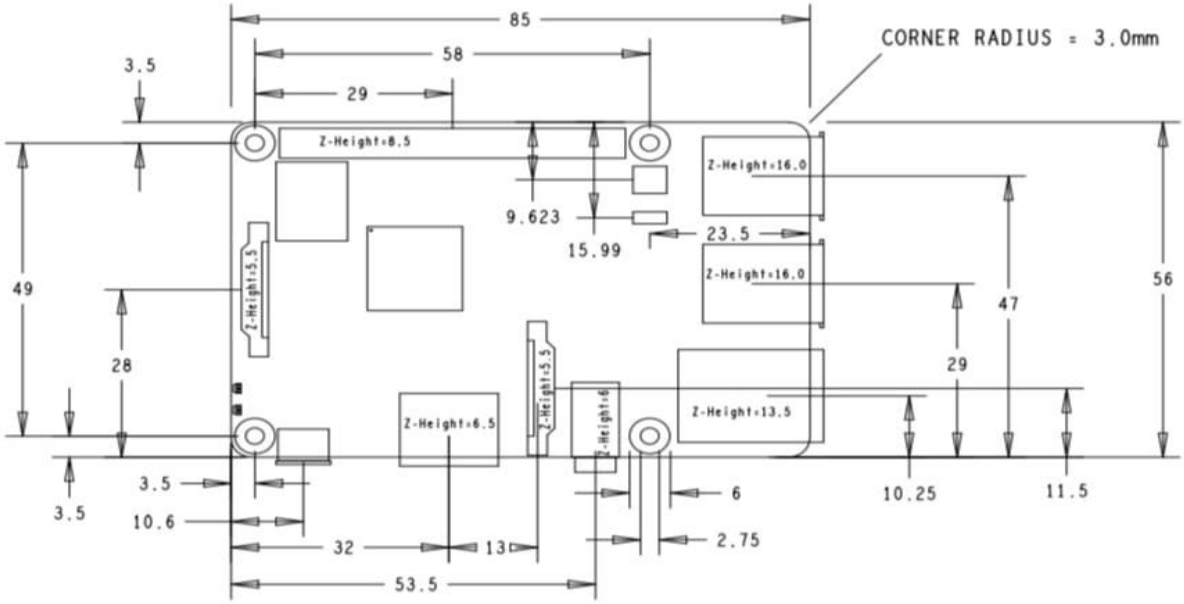


Fig [4.3] Physical specifications of raspberrypi

4.3 SD card

Operating systems:

Various operating systems for the Raspberry Pi can be installed on a Micro SD or SD card, depending on the board and available adapters; seen here is the Micro SD slot located on the bottom of a Raspberry Pi 2 board.

Fig [4.4] SD card

The Raspberry Pi Foundation provides Raspberry Pi OS (formerly called Raspbian), a Debian-based Linux distribution for download, as well as third-party Ubuntu,

Windows 10 IOT Core, RISC OS, Libre ELEC (specialized media Centre distribution) and specialized distributions for the Kodi media Centre and classroom

management. It promotes Python and Scratch as the main programming languages, with support for many other languages. The default firmware is closed source, while unofficial open source is available. Many other operating systems can also run on the Raspberry Pi. The formally verified microkernel seL4 is also supported. There are several ways of installing multiple operating systems on one SD card.

4.4 Raspberrypi camera V2

The Camera Module 2 can be used to take high-definition video, as well as stills photographs.

Size: Around 25 × 24 × 9 mm
Still resolution: 8 Megapixels
Video modes: 1080p47, 1640 × 1232p41 and 640 × 480p206
Sensor: Sony IMX219
Sensor image area: 3.68 x 2.76 mm (4.6 mm diagonal)
Sensor resolution: 3280 × 2464 pixels

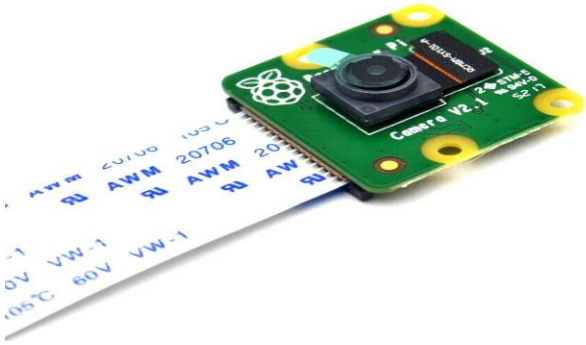


Fig [4.5] Raspberrypi camera V2

Physical of Raspberrypi camera V2

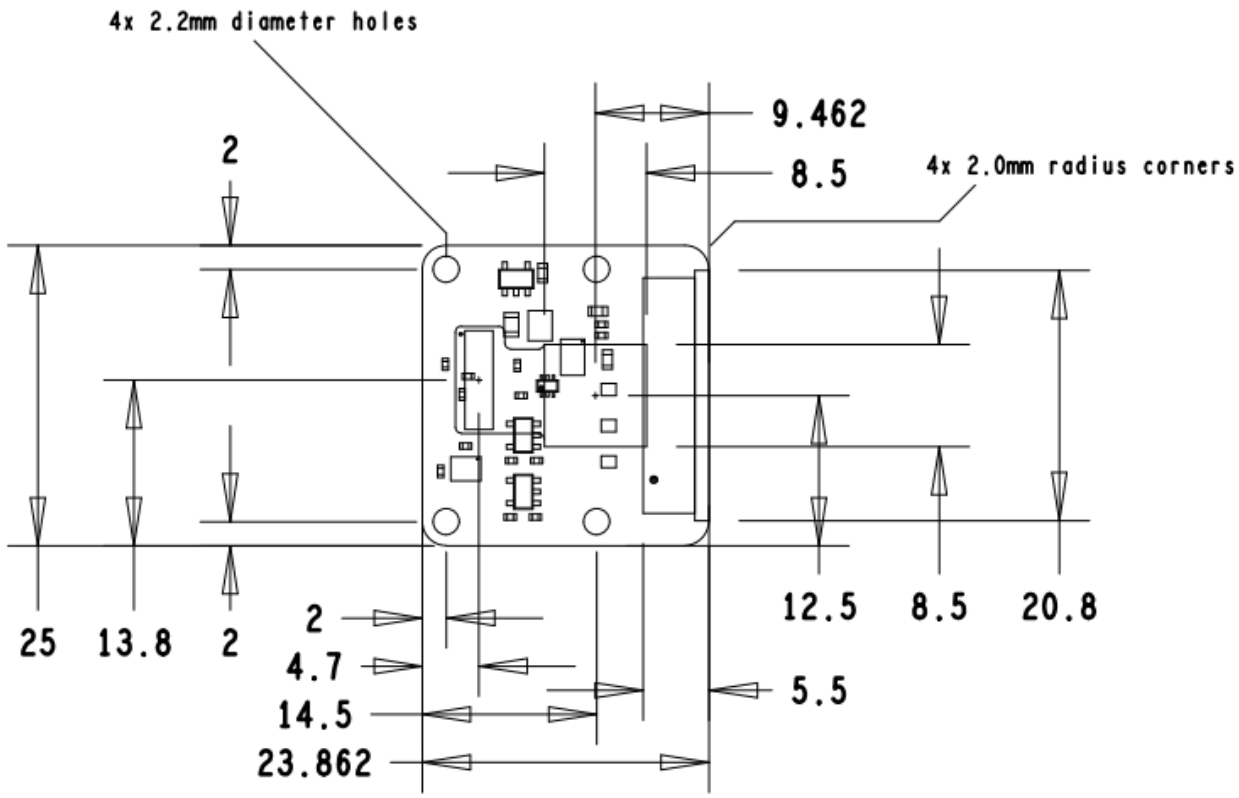


Fig [4.6] physical of camera V2

When using the Raspberry Pi V2 camera for a vein detection device, there are a few potential disadvantages to consider:

1. Image Quality: The Raspberry Pi V2 camera has a fixed focus lens, which means it may not provide the optimal image quality for close-up vein detection. The fixed focus lens is designed for general-purpose use and may not be optimized for capturing fine details required for vein detection.
2. Limited Resolution: The Raspberry Pi V2 camera has a maximum resolution of 8 megapixels (3280 x 2464 pixels). While this resolution may be sufficient for many applications, it may not provide the level of detail required for accurate and precise vein detection, especially if you need to capture small or subtle veins.

3. Limited Frame Rate: The Raspberry Pi V2 camera is capable of capturing video at a maximum frame rate of 30 frames per second (fps) at 1080p resolution. If you require a higher frame rate for real-time vein detection, the Raspberry Pi V2 camera may not meet your needs.
4. Lack of Advanced Features: The Raspberry Pi V2 camera is a basic camera module and lacks some of the advanced features that can enhance vein detection accuracy. For example, it may not have built-in infrared (IR) illumination, which is often used in vein detection devices to improve contrast and visibility of veins beneath the skin.
5. Integration Challenges: The Raspberry Pi V2 camera is designed for use with Raspberry Pi boards and may require additional effort to integrate it into a vein detection device. You may need to design and implement the necessary hardware and software interfaces to connect the camera to your device, which can add complexity to the development process.

If high image quality, resolution, frame rate, or advanced features are critical for accurate vein detection, you may need to consider alternative camera options with better specifications.
Like Raspberry Pi Professional Camera (Night Vision - Adjustable Focus)

4.5 Raspberry Pi Professional Camera (Night Vision - Adjustable Focus)

This is professional high resolution day/ night vision Raspberry Pi camera with fisheye lens for wider field of view and adjustable focal lens. This camera can produced high definition (HD) videos and also high quality picture snapshot. The camera comes with two infrared LEDs of power 1Watt each and 850 nm wavelength for excellent night vision. Each one of the two infrared LED boards have photo resistor for detecting darkness (night) and triggering the infrared LED.

Fig [4.7] IR Night Vision Camera

The camera is equipped with Omnivision5 megapixel OV5647 high resolution camera sensor. In terms of still images, the camera is capable of 2592 x 1944 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. The camera module is good for both professional and beginner users. It can be used for security applications, slow motion, image recognition, time laps and many other countless applications.

Features

Omni vision OV5647 sensor

5MPixel sensor

Still picture resolution: 2592 x 1944

Max video resolution: 1080p

Max frame rate: 30fps

Connection by flat ribbon cable to 15-pin MIPI Camera Serial Interface (CSI) connector S5 on Raspberry Pi computer board

Two IR 850 1W LED light for Night Vision

Light detector sensor for triggering IR LED

Adjustable focal Lens

16cm ribbon cable

Fisheye Lens for wide field of view

Field of View: 160degree (where for normal lens it is 72)

How to Use the infrared led module

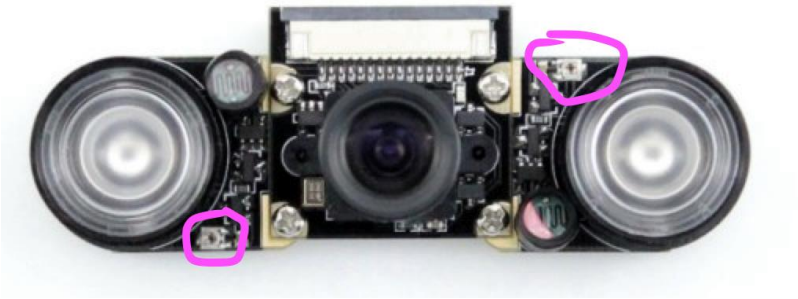


Fig [4.8] Installation of infrared led

Adjust the adjustable resistor to set a proper ambient light threshold, which toggles the infrared LED automatically. When ambient light is lower than threshold value, the infrared LED is on and vice versa.

Our Raspberry Pi Night Vision Camera plugs directly into the CSI connector on the Raspberry Pi (require esan adapter for use with a Pi Zero), and features two high intensity Infrared LED spotlights for nighttime recording! The IR LED's are powered directly from port, and are capable the CSI of lighting an area at a distance of up to 8 m! In testing, the best images were captured at a distance of 3m to 5m. The camera also features an adjustable 3.6mm focal length lens and 75.7 degree viewing angle.

This Raspberry Pi night vision camera uses the same OV5647 as the standard Raspberry Pi camera, and is therefore able to deliver a crystal clear5MP resolution image, or 1080p HD video recording at 30fps!

The Raspberry Pi Zero now comes complete with a camera port! Using the new Raspberry Pi Zero Camera Adapter, you can now use a Raspberry Pi camera to your Zero.

The main difference between a 'normal' camera and an IR one is that normal cameras have a physical filter behind of the lens that cuts out some frequencies of ultraviolet and infrared before it hits the surface of the sensor.

By removal of this filter, the sensor can then detect the IR wavelengths and create an image based on this rather than human visible light. Sometimes they are also in a different place relative to the lens as the focus point of IR is different to normal light - hence some IR cameras produce slightly fuzzy images as they have had the filter removed but nothing done to alter the focal point. You can adjust how the brightness of the IR LEDs responds to the light sensors using these tiny adjustment screws. You're still going to have some IR artifacts in the images since the camera still doesn't have an IR filter. If you have access to light room (or other software), you might be able to create an automated script to remove the IR artifacts (I'm definitely a novice at this but this might be helpful.

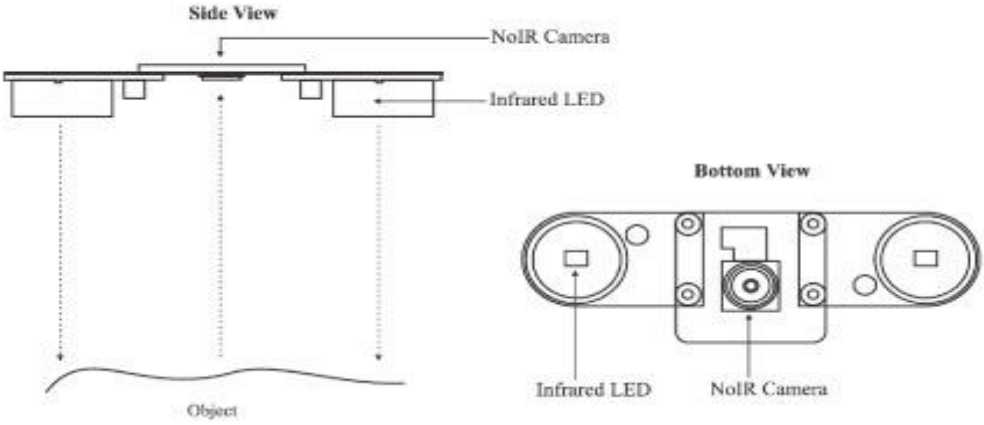


Fig [4.9] physical of IR camera

4.6 Raspberry pi official USB-C Power Supply

Specification

Output

Output voltage: +5.1V DC Minimum load current: 0.0A

Nominal load current: 3.0A Maximum power: 15.0W Load regulation: ±5%

Line regulation: ±2%

Fig [4.10] USB-C Power Supply

Ripple & noise: 120mVp-p

Rise time: 100ms maximum to regulation limits for DC outputs

Turn-on delay: 3000ms maximum at nominal input AC voltage and full load

Protection: Short circuit protection, overcurrent protection, over temperature

Protection Efficiency: 81% minimum (output current from 100%, 75%, 50%, 25%)

Output cable: 1.5m 18AWG

Output connector: USB Type-C

Input

Voltage range: 100–240Vac (rated) 96–264Vac (operating)

Frequency: 50/60Hz ±3Hz Current: 0.5A maximum

Power consumption (no load): 0.075W maximum

Inrush current: No damage shall occur and the input fuse shall not blow

4.7 Hardware components images

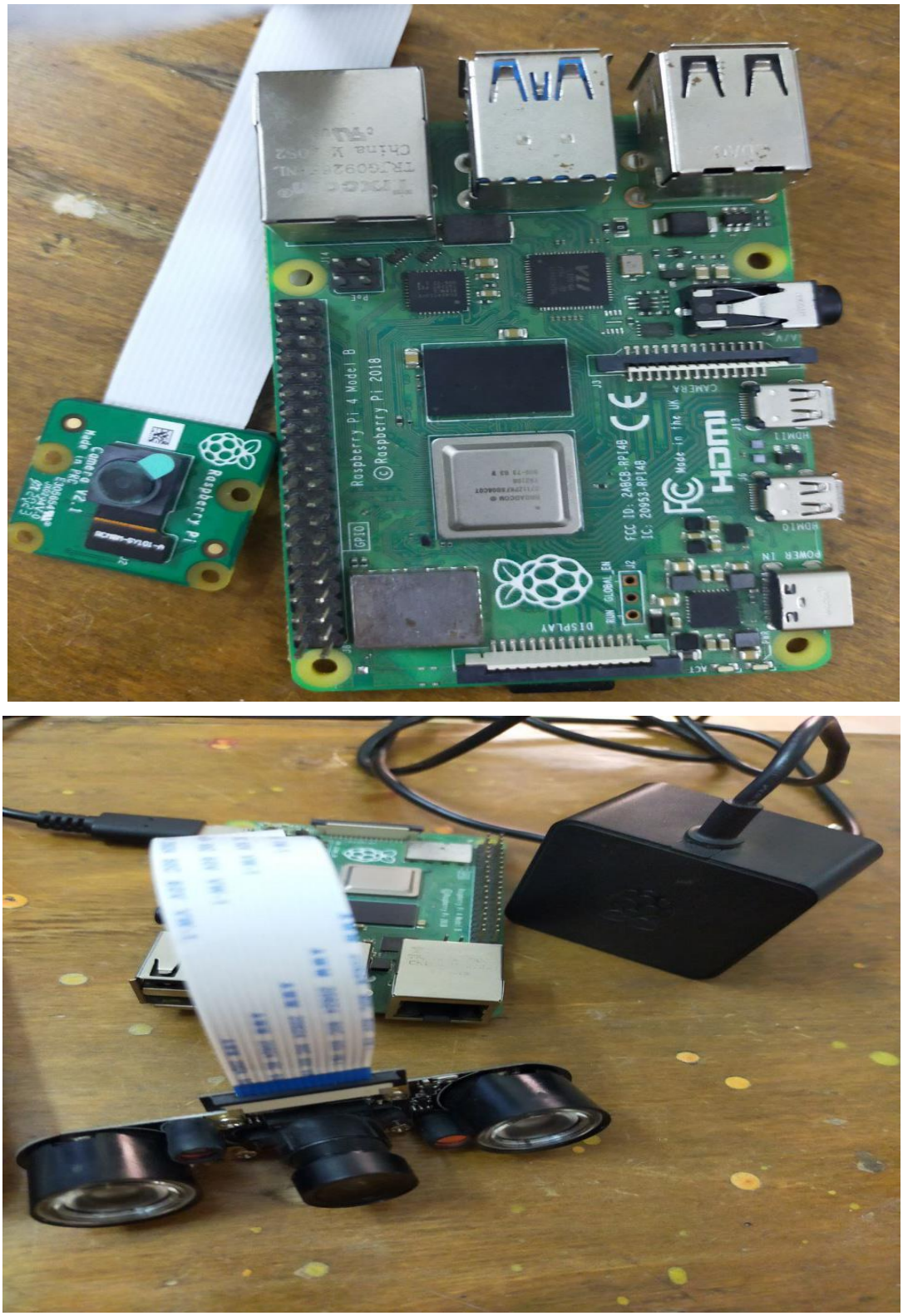


Fig [4.11] Hardware components

Chapter Five

Software

5.1 Open CV-Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by **Guido van Rossum** that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

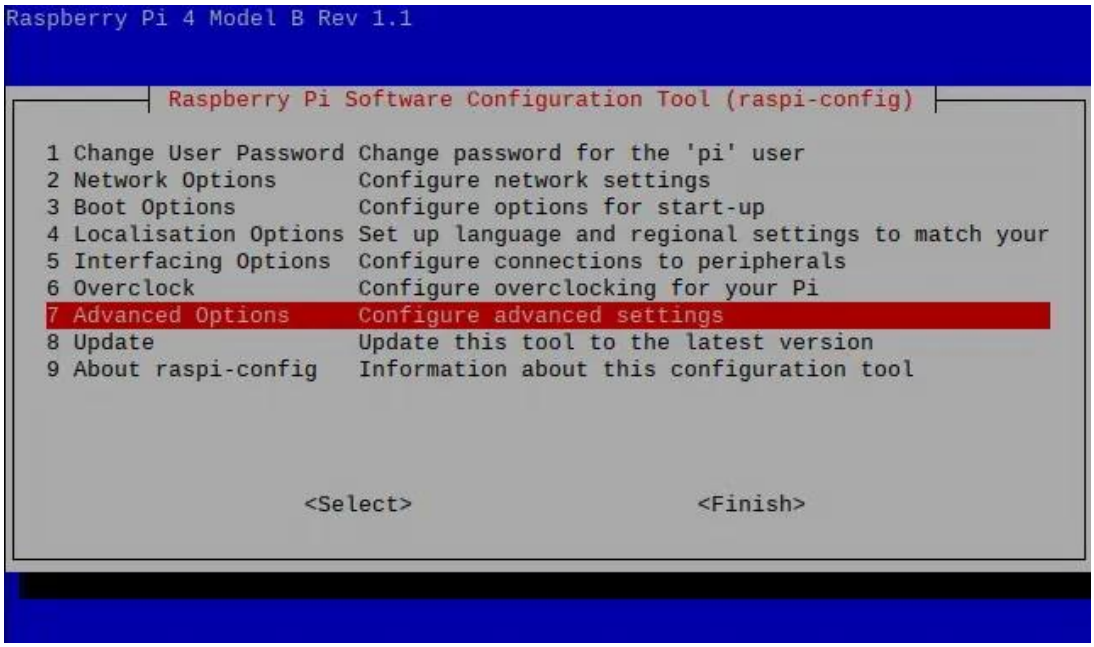
Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of **Numpy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

5.2 Step 1: Set Up Raspberry Pi

Simply insert the micro SD card into your board after upload the operating system and then power it ON. The first thing you should do is make sure your Pi is using the full SD card. It's easy enough to do:

```
sudo raspi-config
```



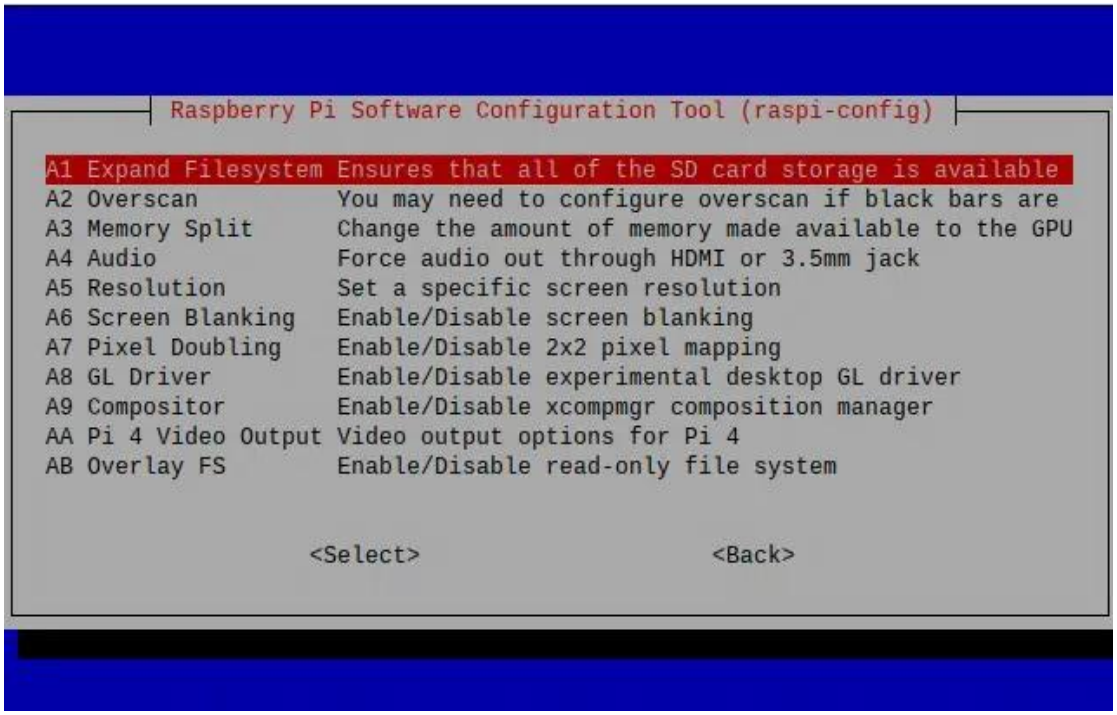


Fig [5.1] Set Up Raspberrypi

After that reboot the raspberry pi.

Now you can check to make sure you've got the space:

Cool, we're ready. Let's update the system:

```
sudo apt-get update && sudo apt-get upgrade
```

This is all you'll need to do the

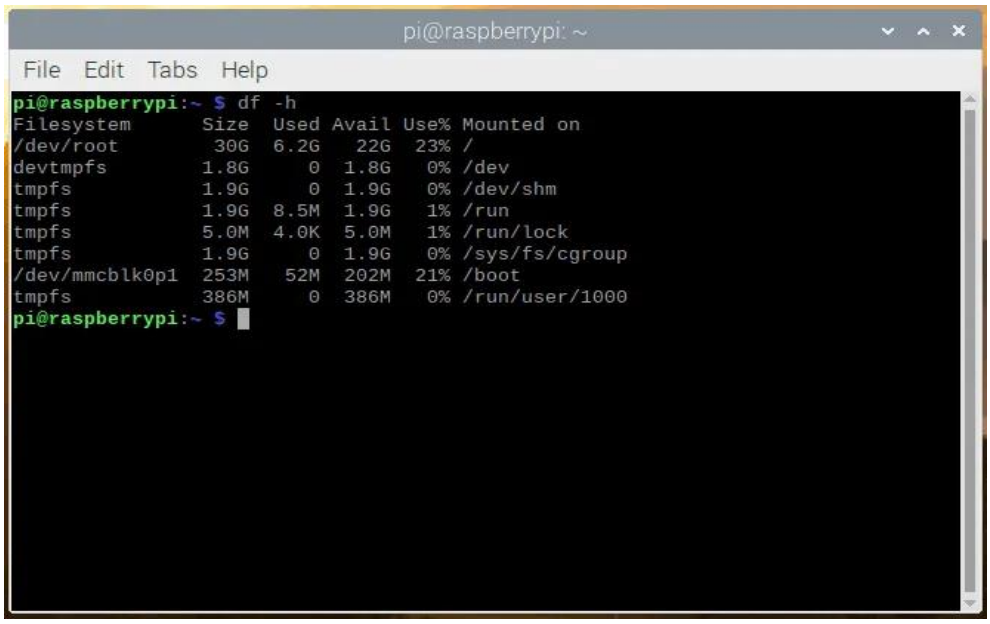
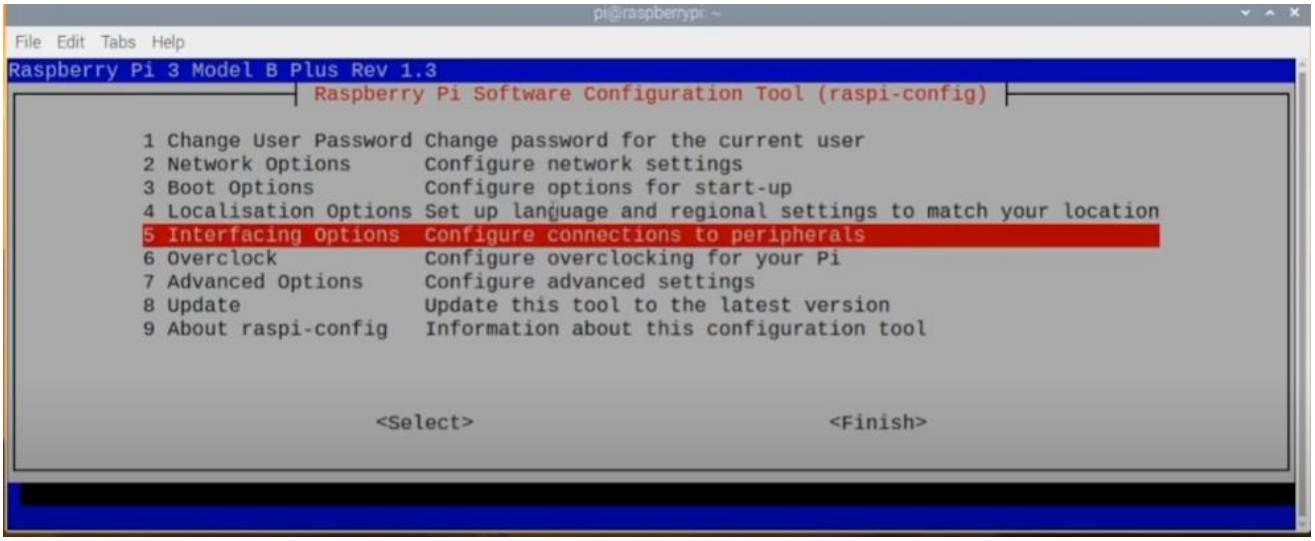
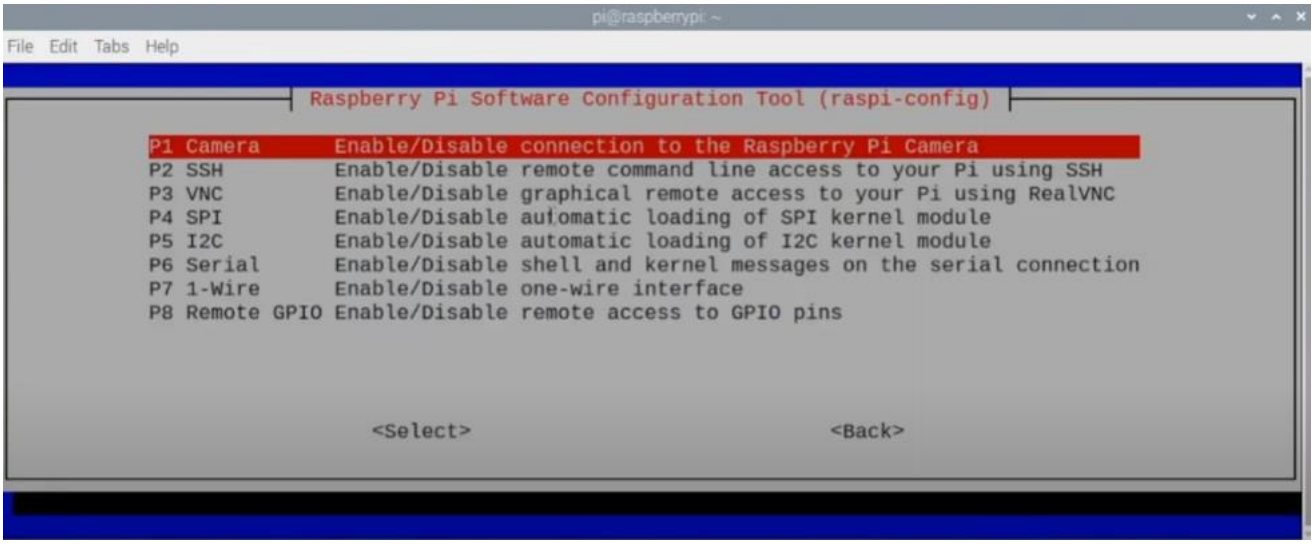


Fig [5.2] Step to get space

5.3 Step 2: enable the camera





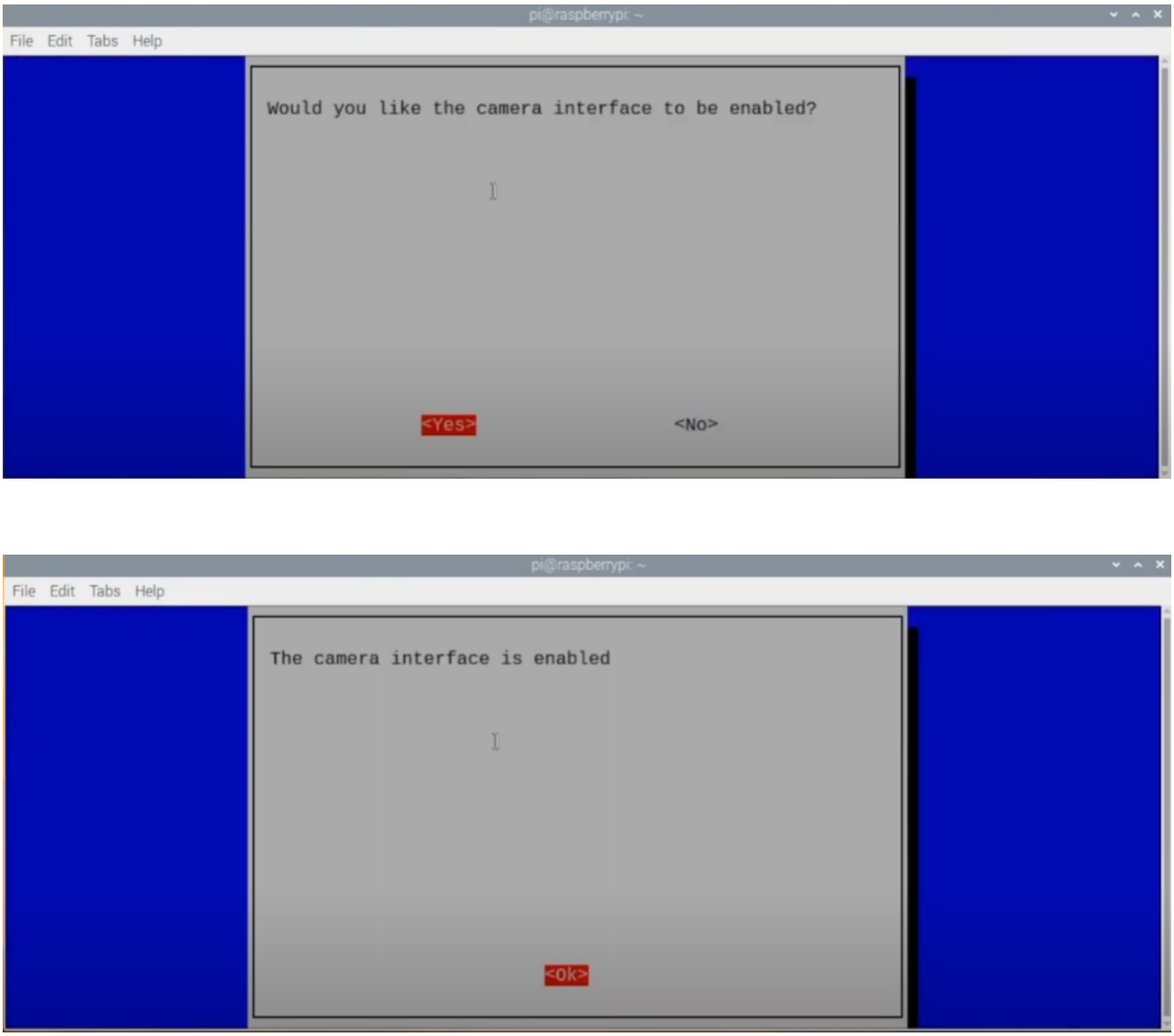


Fig [5.3] Steps to enable the camera

And then reboot the raspberry.

5.4 Step 3: Installing OpenCV

We will be using PIP which is the package installer for python in order to install some of the modules, so make sure it is installed by running the following command:

- Sudo apt install python3-pip

Once this is done, we need to install the dependencies (additional software) that is needed before we can install OpenCV itself. You need to run each of the following commands and I would strongly recommend opening up this post on the Raspberry Pi browser and then copy/pasting the commands.

- sudo apt install libatlas-base-dev -y
- sudo apt install libjasper-dev -y
- sudo apt install libqtgui4 -y
- sudo apt install python3-pyqt5 -y
- sudo apt install libqt4-test -y
- sudo apt install libhdf5-dev libhdf5-serial-dev -y
- sudo pip3 install opencv-contrib-python==4.1.0.25

5.5 Step 4: install pi camera module

This will install OpenCV for us. Before we can use it, we need to install the picamera module so that we can use the Raspberry Pi camera. This can be done by running the following command:

- pip3 install picamera[array]

What is Image Processing?

Image processing involves performing some operations on an image, in order to get an enhanced image or to extract some useful information from it.
The field of image processing is upcoming and advancing rapidly. It enables object detection in images that has applications ranging from self-driving cars to tumor detection in the field of medical science.

What are images?

What you see as an image is actually a 2D matrix for computer.

A digital image is stored as a combination of pixels. Each pixel further contains a different number of channels. If it a grayscale image, it has only one pixel, whereas a colored image contains three channels: red, green, and blue.

Each channel of each pixel has a value between 0 and 255. With the combination of red, green and blue in different proportions we can create any colour.

2. Programming to Read images:

To read an image using OpenCV, use the following line of code.

```
img = cv2.imread('image_path')
```

Now the variable **img** will be a matrix of pixel values. We can print it and see the RGB values.

The image we are using for this example is:

Fig [5.4] Example

To print the matrix use:

```
print(img)
```

3. Complete Implementation to Read Images in Python using OpenCV:

```
import cv2

#read
img = cv2.imread('sample_image.jpg')

#show
print(img)
```

Output:

```
[[[203 201 170]
  [202 200 169]
  [202 200 169]
  ...
  [175 173 138]
  [175 173 138]
  [174 172 137]]

 [[203 201 170]
  [202 200 169]
  [202 200 169]
  ...
  [175 173 138]
  [175 173 138]
  [174 172 137]]

 [[203 201 170]
  [202 200 169]
  [202 200 169]
  ...
  [175 173 138]
  [175 173 138]
  [174 172 137]]

 ...
```

Fig [5.4] Example

Display Images Using Python OpenCV

To show the image using OpenCV use the following line:

```
ccv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds.

The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a keystroke.

The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a keystroke.

Python Implementation:

The complete code for displaying the image is:

```
import cv2
#read
img = cv2.imread('sample_image.jpg')
#show
```

```
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Manipulating images using Python OpenCV:

There are a lot of functionalities in OpenCV that allow you to manipulate an image. We will look at how to turn an image into grayscale. A grayscale image means that each pixel will only have one channel with values between 0 to 255.

The code to do that is:

```
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Implementing the Python code:

The complete code for turning an image to grayscale is :

```
#read
img = cv2.imread('sample_image.jpg')
#to grayscale
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#show
print(gray_image)
cv2.imshow('image',gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
[[192 191 191 ... 163 163 162]
 [192 191 191 ... 163 163 162]
 [192 191 191 ... 163 163 162]
 ...
 [104 108 114 ... 148 148 148]
 [111 114 119 ... 149 149 149]
 [114 117 120 ... 149 150 150]]
```

Fig [5.5] the dimension of matrix

You can see that the dimension of this matrix is different from that of a colored image.

Save the manipulated image file:

To save an image after manipulation use the following line of code:

```
cv2.imwrite('sample_grayscale.jpg',gray_image)
```

Here, the first argument is the name you want to give to the file, the second argument is the variable that contains the image you want to save. We are saving the grayscale image we created above.

Complete Python code for turning a color image greyscale:

The complete code for saving an image is:

```
import cv2


#read
img = cv2.imread('sample_image.jpg')
```

```
#to grayscale
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#show
print(gray_image)
#save
cv2.imwrite('sample_grayscale.jpg',gray_image)
```

Convert color image to black and white

In the following example, we will read the following color image using cv2.imread() as grey scale image and then apply cv2.threshold() function on the image array.

Input Image

Fig [5.6] Ex: Convert color image to black and white

```
import cv2
#read image
img_grey = cv2.imread('D:/original.png', cv2.IMREAD_GRAYSCALE)
# define a threshold, 128 is the middle of black and white in grey scale
thresh = 128
# threshold the image
img_binary = cv2.threshold(img_grey, thresh, 255, cv2.THRESH_BINARY)[1]
#save image
cv2.imwrite('D:/black-and-white.png',img_binary)
```

The output black-and-white.png is as shown below.

Output Binary Image

Fig [5.6] Ex: Convert color image to black and white

5.6 Step 5: How to capture a frame from real-time camera video using OpenCV

What is a frame?

A **frame** is an image that forms a single instance of a video. A video consists of a lot of frames running per second (also known as frames per second). For example, a video streaming at 30 fps means that it displays 30 images in a second.

Steps:

We will be using Python language for this purpose, but the OpenCV library is available in most of the high-level languages, e.g., C++, Java, Kotlin, etc.

1. As we have to use OpenCV methods, they have to be included in our code file. Use the following line of code to include OpenCV library functions in Python import cv2.

```
import cv2
```

2. The above code will include all the libraries defined in cv2 library (OpenCV library in Python). Now, the next step is to create a variable that will hold the video capturing object. In this case, the variable is vidcap, which will hold the reference returned by the VideoCapture function of the cv2 library. The below code demonstrates this:

```
vidcap = cv2.VideoCapture(0)
```

3. The vidcap object created will now have the reference to the integrated webcam (for laptops) or any other portable webcam attached via input ports. The next step is to check if cv2.VideoCapture is successfully used.
   Use vidcap.isOpened(). It will return true if the connection with the camera was successful. Otherwise, false will be returned. If true is returned, we will continue. Otherwise, we will print the error message.

```
if vidcap.isOpened():

 #do something

else:
   print("Cannot open camera")
```

4. If our connection with the camera was successful, we'll capture a frame from the live video. For that, use the following line of code inside the if block from the previous step.

```
if vidcap.isOpened():
  ret, frame = vidcap.read()
else:
  print("cannot open camera")
```

The read () function returns a tuple of size 2. The first element of the tuple is a true or false, depending on whether the frame was captured successfully, and the second element of the tuple contains the actual video frame if it was successfully captured.

After this statement, another if block will be added to the previous if block that will check whether the frame was successfully captured.

```
if vidcap.isOpened():
  ret, frame = vidcap.read()
  if ret:
    #do something
  else:
     print("Error : Failed to capture frame")
```

```
else:
  print("cannot open camera")
```

5. If the frame was successfully captured, we will display the captured frame by using cv2.imshow(windname, frame). The captured frame will be displayed in a new window. The windname argument is a string for the name you want to give to your frame that will be displayed with the frame window and the frame attribute is an object that holds the frame captured from the previous step. Now, it will look something like this:

```
if vidcap.isOpened():
  ret, frame = vidcap.read()
  if ret:
    cv2.imshow('Frame', frame)
  else:
    print("frame not captured")
else:
  print("cannot open camera")
```

If the code runs successfully, a new window is opened and it then immediately closes. To keep the frame in persistent view, we need to use a loop with a condition to break out of the loop. The following lines of code inside the second if block caters to it:

```
while(True):
    cv2.imshow(windowname,frame)
    if cv2.waitKey(1) & 0xFF==ord('q'):
      break
```

After adding this code, the window containing the video frame will open, but the program will wait for the 'q' key to be pressed to break out of the loop. Once 'q' is pressed, the window will close.

Below is the complete Python code for the task discussed above.

```
import cv2   #include opencv library functions in python
#Create an object to hold reference to camera video capturing
vidcap = cv2.VideoCapture(0)
#check if connection with camera is successfully
if vidcap.isOpened():
    ret, frame = vidcap.read()  #capture a frame from live video
    #check whether frame is successfully captured
    if ret:
        # continue to display window until 'q' is pressed
        while(True):
            cv2.imshow("Frame",frame)   #show captured frame
```

```
        #press 'q' to break out of the loop
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    #print error if frame capturing was unsuccessful
    else:
        print("Error : Failed to capture frame")


# print error if the connection with camera is unsuccessful
else:
    print("Cannot open camera")
```

5.7 Step 6: analysis and implementation of vein image processing

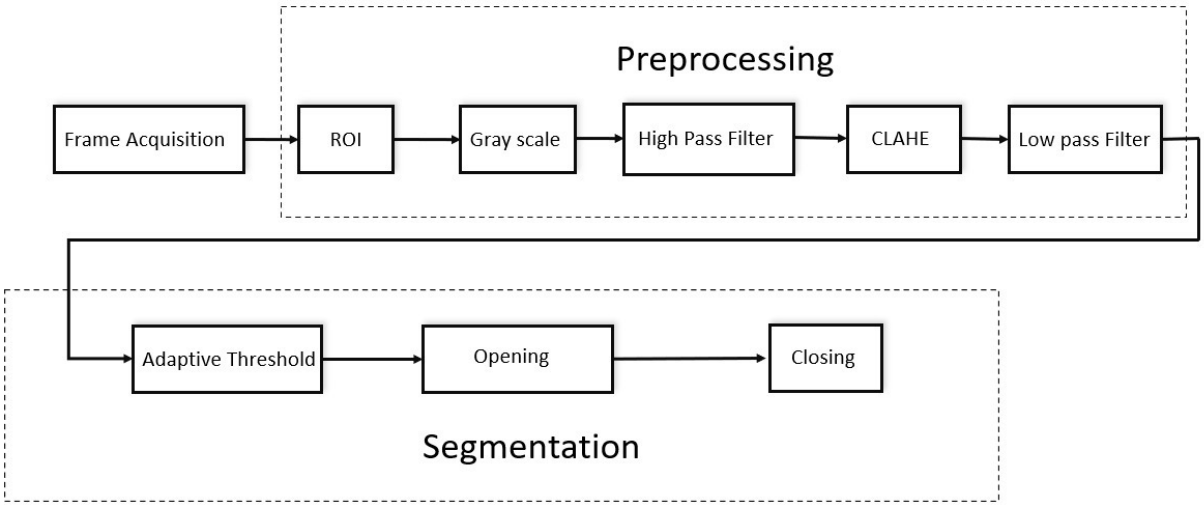


Fig [5.7] steps of image processing

Implementation of image processing in this research is divided into two stages:

1. Preprocessing.
2. Segmentation.

Preprocessing aims to create a clean image and strengthen the difference between the vein pattern and the skin. Furthermore, the segmentation process aims to take the pattern of the veins and remove the parts that are not vein.

To perform preprocessing on the image that has been obtained, here are some steps that must be done. First is to determine the coordinate area of the image to be processed by setting the region of interest (ROI). The obtained image has a low information about vein vessel. To clarify the difference between the veins and the skin, the image enhancement such as histogram equalization and high boost filter need to be performed. By performing image enhancement, the veins will become darker and the skin will become brighter. Once the difference between the skin and

the veins looks clear, the next step is performing the segmentation process to obtain the vein pattern.

The images obtained by the camera are determined to be 640 x 480-pixel to avoid heavy computation because it uses a single board computer. In this case, the ROI is placed on the center of the frame because it has relevance to the projection method that will be developed.

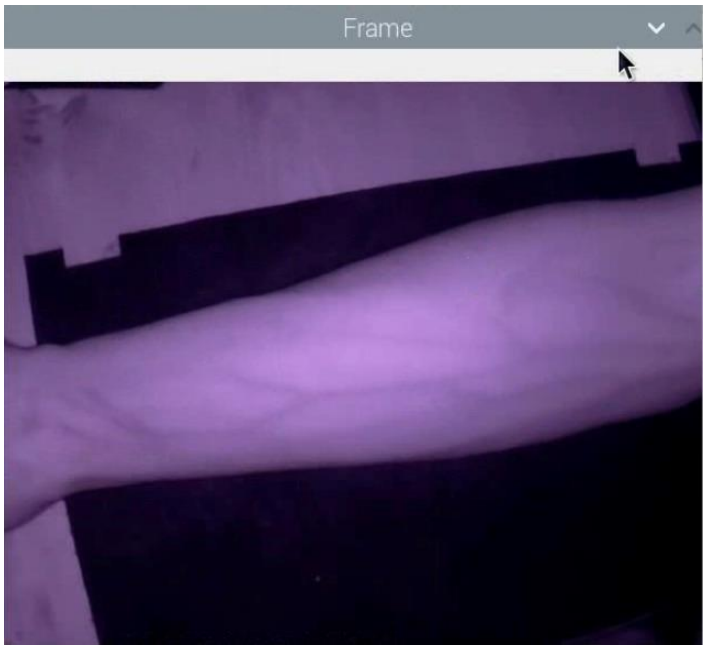


Fig [5.8] Frame

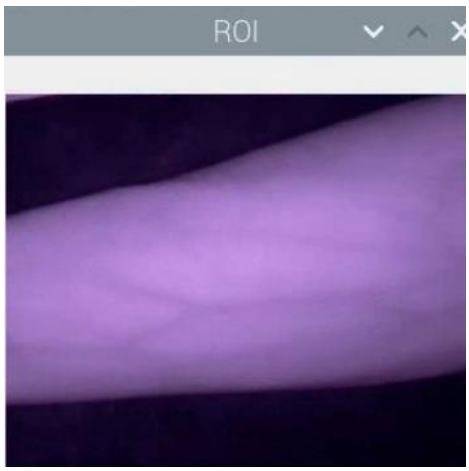


Fig [5.9] ROI

Vein image converted into a binary image using adaptive thresholding. The result of adaptive thresholding is not perfectly getting the vein pattern because there is some point of noise.

To remove some small parts that are not categorized as a vein pattern, we perform the morphology opening process. However, the morphology opening process causes some pattern loss.

Fig [5.10] grey

We need to restore it and then morphology closing process is performed to solve the problem. Since the two morphological process cannot be maximized to remove noise, then we use the contour area to determine the noise area and perform the elimination process to obtain a clean vein image and ready to project. Before the back-projection process is performed, the clear vein image colorized to yellow to avoid feedback rays from white lights. The results obtained by NIR or V2 cameras still do not provide clear information about the difference between the veins and the skin. Because of this, some studies use histogram equalization to clarify vein images. However, in this case it would be more appropriate to use contrast limited adaptive histogram equalization (CLAHE). Since this method so it will is local so it will not be affected by background conditions.

Fig [5.11] HBF

Fig [5.12]Clahe

There is a problem where the result of CLAHE causes the appearance of dark parts that make the segmentation process even more difficult. To deal with the problem, we propose to use a method that is able to clarify the veins without causing the appearance of dark parts. This method is called high-boost filter where the working principle is by subtracting the original image with the original image that has been given a low pass filter.

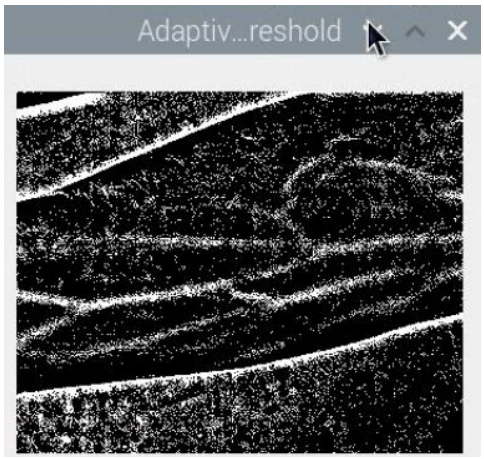


Fig [5.13] Adaptive

Because the result of the histogram equalization process and the high boost filter also amplify the noise, the vein image needs to be filtered so that the segmentation process becomes easier. Based on Habib's research [11], they used the median filter after grayscale process while in this study using Gaussian filter with kernel size 7 x 7 to reduce noise. To know better filter results can be analyzed with peak signal to noise ratio (PSNR). The PSNR method computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is often used as a quality measurement between the original and the compressed or reconstructed image.

The result is the image filtered with median filter has a PSNR of 41.5 dB, while the image filtered with the Gaussian filter has a PSNR of 42.6 dB Both of these results have a PSNR value that is not much different but the Gaussian filter is chosen because having a faster computation time. The next process is vein image segmentation by using adaptive thresholding. Parameters that we used is the block size is set to 11 and in determining the threshold value, we prefer to use mean

neighbor rather than Gaussian. On this system the use of Gaussian type threshold value is less suitable because it causes many parts of the veins pattern disconnected. While the use of mean threshold value is more suitable because the vein part is only slightly disconnected.

We used morphology closing after thresholding process. In this case, to remove the noise we attempt to use morphology opening. This process in addition to removing the noise also causes the loss of some vein patterns and to restore the vein pattern can be solved by morphology closing.

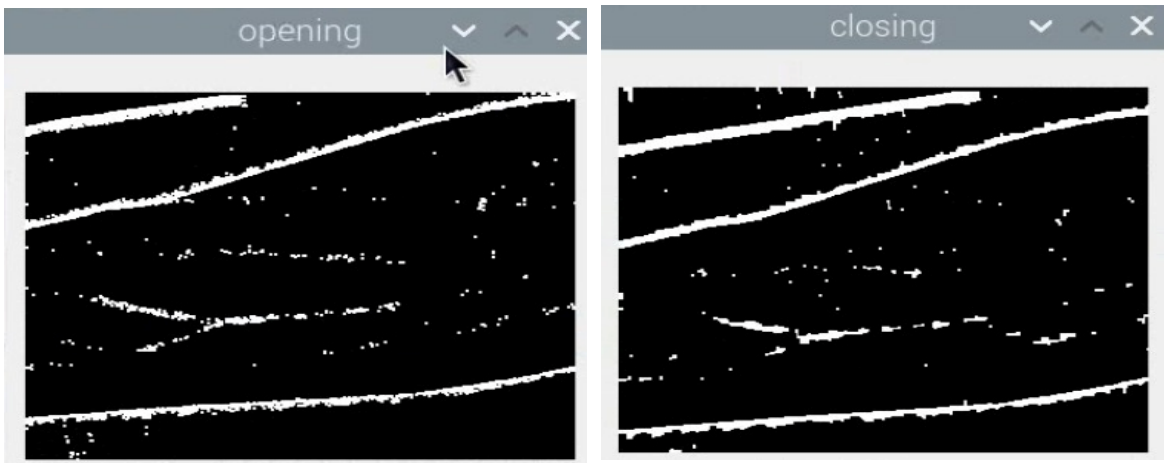


Fig [5.14] Opening, Closing Image

Improvement method:

We used the Thresholding method, in which the threshold value is used on the image, but there is a problem, which is that we may lose a large amount of information in the image, and to solve this problem we use of the mean Adaptive Threshold method, in which the image is divided into several parts.

the Threshold value is calculated for each part From the parts of the image, then it is applied to each part of the image But the output image sometimes with different color gradations from one image to another, we need to change the Threshold value every time we change the image in order to get the best output and so as not to lose an amount of information in the output image.

To solve this problem, we made a trackbar that controls the image parts, their size (M * N) and the Threshold value for each part of the image
All the operations that take place on the image, such as filters, and converting the colors in the image from white to black and vice versa, are also controlled

TrackBar

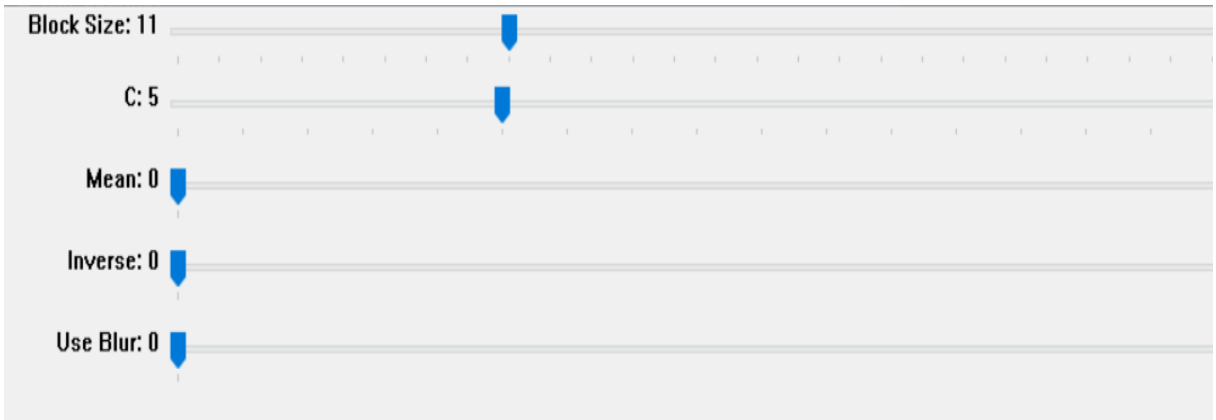


Fig [5.15] Trackbar

Block Size Track

The first (Block Size) path, which controls the (M * N) of the images that have been split from the original image, Take the values from 3 to 50.

For example, if the path is moved to the number 11, this means that each part has been fragmented in the image, the size of which is 11 * 11

C Track

The value of T (the Threshold value) is subtracted from this value (C) so that the Threshold value that is applied to the image is closest to the truth, take value from 0 to 30.

Mean Track

Mean Track contains two values 0 and 1

-If the value is zero, then the value of the Threshold is calculated using Gauss's law

-But if the value is 1 it means that the value of the Threshold is calculated using the mean's law.

Inverse Track

In this path, the colors in the image can be changed from white to black and black to white or vice versa

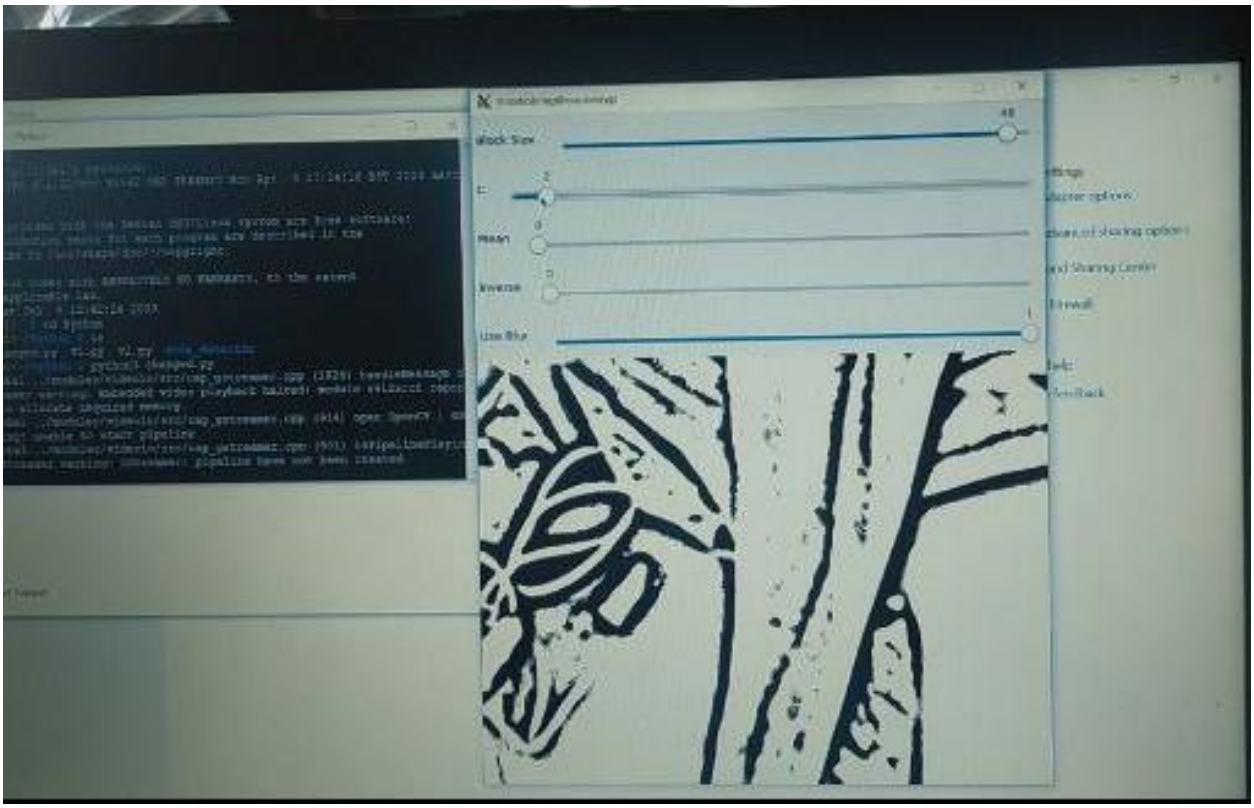
Blur Track

The low pass Filter (Gaussian Filter) is used to remove noise from the image to improve the image.

Chapter Six

Results

6.1 Software results



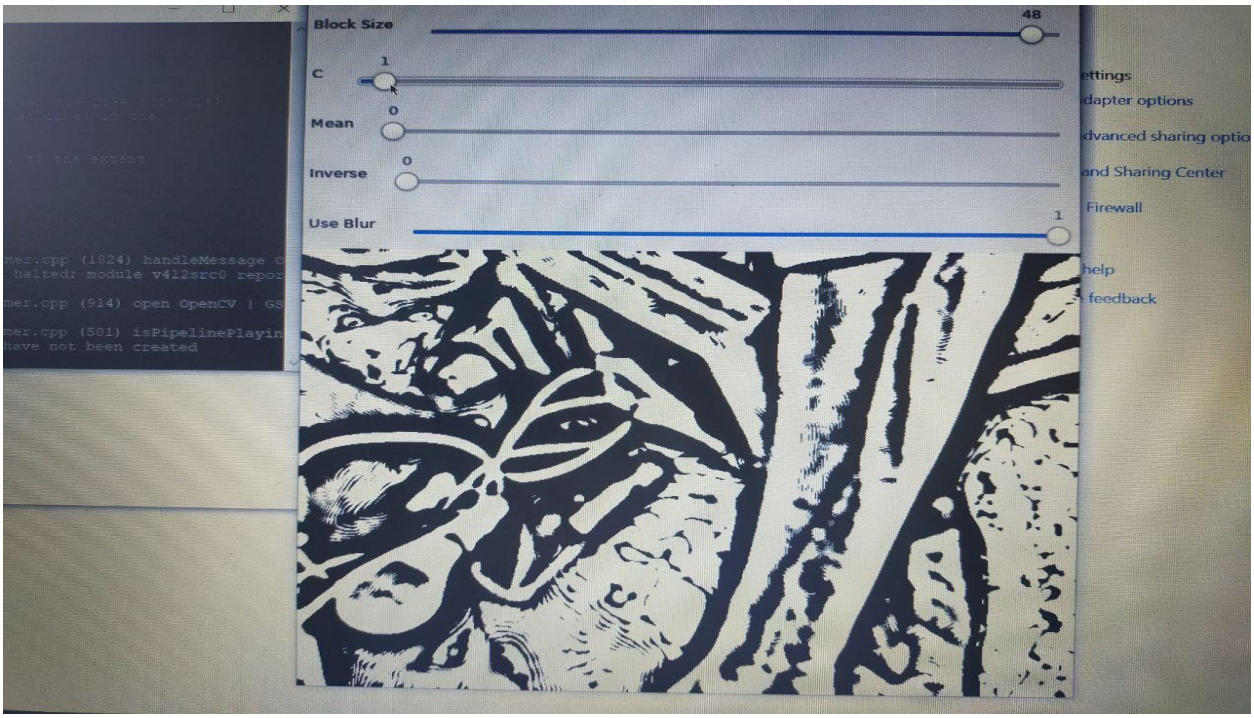
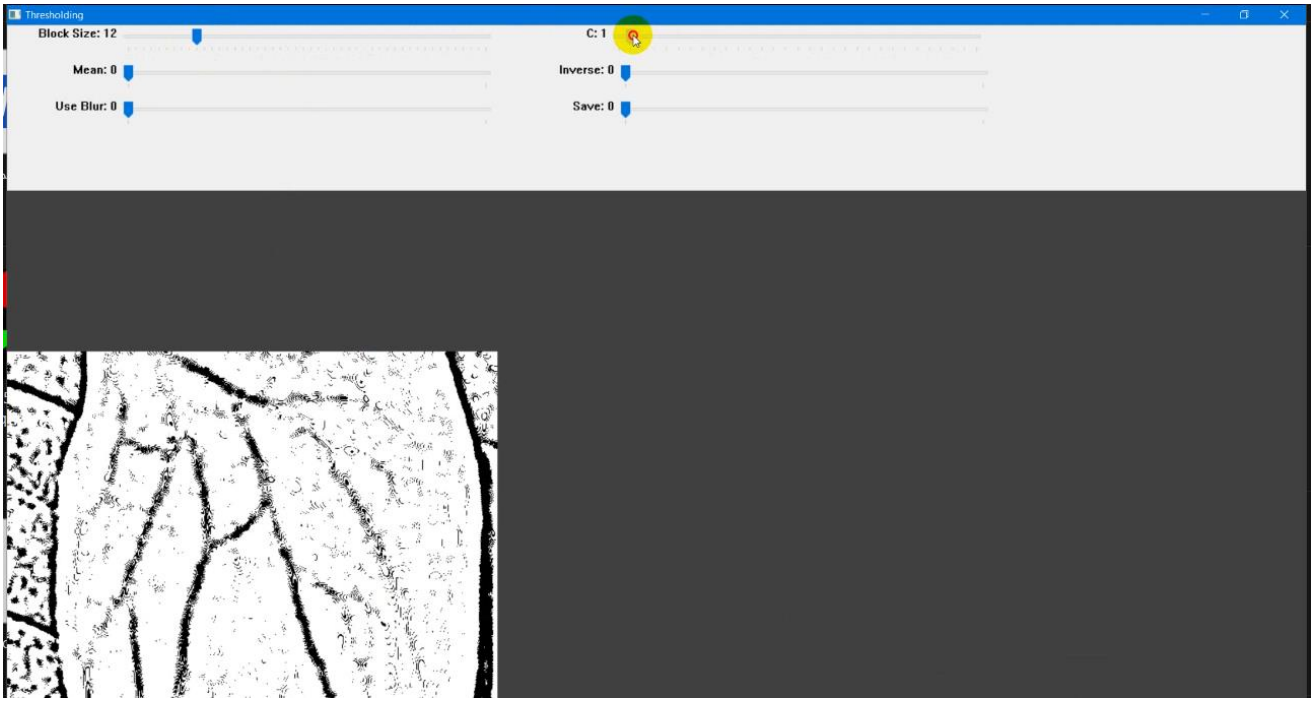
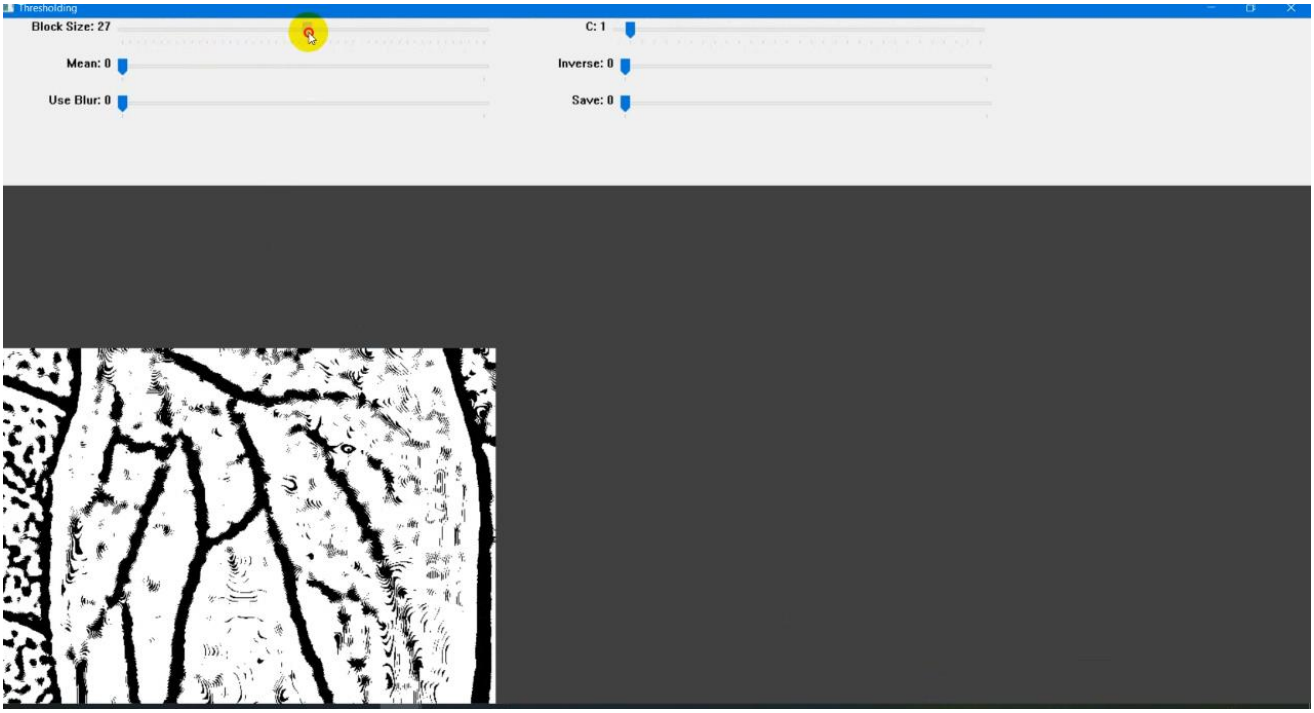


Fig [6.1] Software results using Raspberrypi camera V2


Thresholding
Block Size: 12
C: 1
Mean: 0
Inverse: 0
Use Blur: 0
Save: 0


Thresholding
Block Size: 27
C: 1
Mean: 0
Inverse: 0
Use Blur: 0
Save: 0

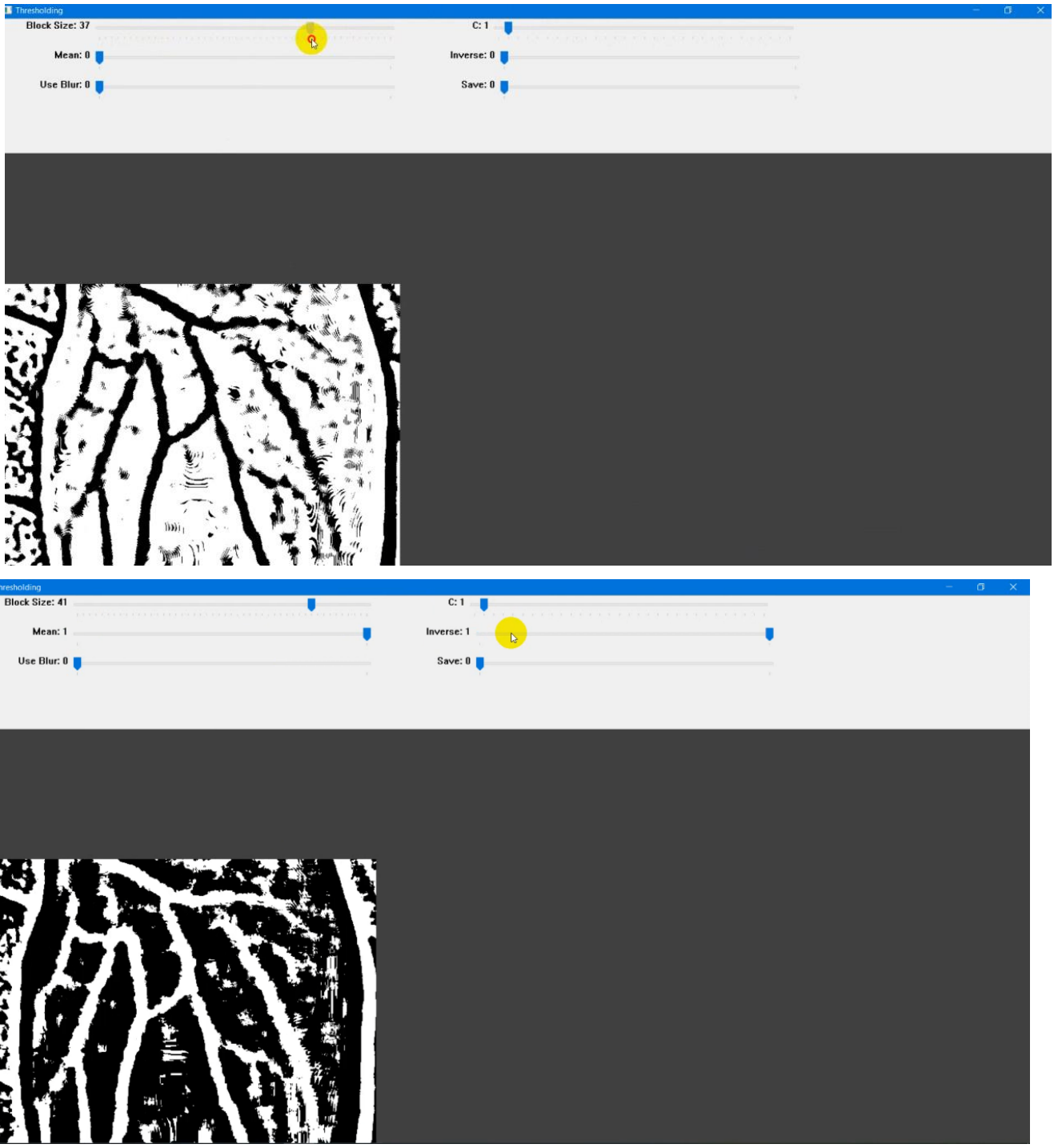


Fig [6.2] Software results using IR camera

Chapter Seven

Appendices

7.1 Code1:

```python
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time
import cv2
# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))
# allow the camera to warmup
time.sleep(0.1)
# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
  # grab the raw NumPy array representing the image, then initialize the timestamp
  # and occupied/unoccupied text
  image = frame.array

  #rotating the image
  (h,w)= image.shape [:2]
  center= (w / 2, h / 2)

  M= cv2.getRotationMatrix2D(center, 180, 1.0)
  rotated= cv2.warpAffine(image, M, (w,h))

  #Extract ROI
  ROI_image = image[144:336,192:448]

  # Grayscale frame
  grey= cv2.cvtColor(ROI_image,cv2.COLOR_BGR2GRAY)

  # Apply Highbootsting filter (histogram equalization)
  kernel = np.array([[0, -1, 0], [-1, 5,-1],[0, -1, 0]])
  image_sharp = cv2.filter2D(src=grey, ddepth=-1, kernel=kernel)
```

```python
# Adaptive histogram equalization
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
equalized = clahe.apply(image_sharp)

# Adaptive Mean Threshold
thresh = cv2.adaptiveThreshold(equalized, 255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 21, 10)

# Opening
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2,2))
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)

#closing
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (4,4))
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)

# Show the frame
cv2.imshow("Frame", image)
cv2.imshow("ROI", ROI_image)
cv2.imshow("grey" , grey)
cv2.imshow("HBF", image_sharp)
cv2.imshow("CLAHE",equalized )
cv2.imshow("Adaptivethreshold", thresh )
cv2.imshow("opening", opening)
cv2.imshow("closing", closing)
key = cv2.waitKey(1) & 0xFF
# clear the stream in preparation for the next frame
rawCapture.truncate(0)
# if the `q` key was pressed, break from the loop
if key == ord("q"):
    break
```

Fig [7.1] Code 1

7.2 Code2:

```python
import cv2

import numpy as np


import mahotas

Win_NAME = "Thresholding"
cv2.namedWindow(Win_NAME)
TB_T = "T"
TB_INVERSE = "Inverse"
TB_Blur = "Use Blur"
TB_Mean = "Mean"
TB_C = "C"
TB_SAVE = "Save"
TB_Block_Size = "Block Size"


def do_nothing(x):
    pass


def global_thresholding(image, blurred, T, inverse=False, use_blur=False):
    if inverse:
        if use_blur:
            _, b_inv_img = cv2.threshold(
                blurred, T, 255, cv2.THRESH_BINARY_INV)
            return b_inv_img
            # cv2.imshow("Blur INV Global", b_inv_img)
        else:
            _, t_inv_img = cv2.threshold(image, T, 255, cv2.THRESH_BINARY_INV)
            return t_inv_img
            # cv2.imshow("Blur INV Global", t_inv_img)

    else:
        if use_blur:
            _, b_img = cv2.threshold(blurred, T, 255, cv2.THRESH_BINARY)
```

```
            return b_img
            # cv2.imshow("Blur Global", t_img)
        else:
            _, t_img = cv2.threshold(image, T, 255, cv2.THRESH_BINARY)
            return t_img


def adaptive_thresholding(image, blurred, use_blur=False, mean=True, inverse=False, blockSize=11, C=5):
    if mean:
        if inverse:
            if use_blur:
                b_atm_inv = cv2.adaptiveThreshold(blurred,
                                                  255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                                  cv2.THRESH_BINARY_INV, blockSize, C)

                return b_atm_inv

            else:
                atm_inv = cv2.adaptiveThreshold(image,
                                                255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                                cv2.THRESH_BINARY_INV, blockSize, C)
                return atm_inv

        else:
            if use_blur:
                b_atm = cv2.adaptiveThreshold(blurred,
                                              255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                              cv2.THRESH_BINARY, blockSize, C)

                return b_atm

            else:
                atm = cv2.adaptiveThreshold(image,
                                            255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                            cv2.THRESH_BINARY, blockSize, C)

                return atm
```

```
    else:
        if inverse:
            if use_blur:
                b_atg_inv = cv2.adaptiveThreshold(blurred,
                                                  255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                                  cv2.THRESH_BINARY_INV, blockSize, C)

                return b_atg_inv

            else:
                atg_inv = cv2.adaptiveThreshold(image,
                                                255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                                cv2.THRESH_BINARY_INV, blockSize, C)

                return atg_inv

        else:
            if use_blur:
                b_atg = cv2.adaptiveThreshold(blurred,
                                              255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                              cv2.THRESH_BINARY, blockSize, C)
                return b_atg

            else:
                atg = cv2.adaptiveThreshold(blurred,
                                            255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                            cv2.THRESH_BINARY, blockSize, C)
                return atg


cv2.createTrackbar(TB_Block_Size, Win_NAME, 11, 50, do_nothing)
cv2.setTrackbarMin(TB_Block_Size, Win_NAME, 3)

cv2.createTrackbar(TB_C, Win_NAME, 5, 30, do_nothing)
cv2.createTrackbar(TB_Mean, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_INVERSE, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_Blur, Win_NAME, 0, 1, do_nothing)
```

```python
cv2.createTrackbar(TB_Blur, Win_NAME, 0, 1, do_nothing)

cap = cv2.VideoCapture('D:\vien\armvien.mp4')
while(True):
    _, frame = cap.read()

    #image = cv2.cvtColor(cv2.imread("H:/folder9/download.jfif"), cv2.COLOR_BGR2GRAY)
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # image = cv2.cvtColor(cv2.imread(r'H:/folder9/download.jfif'), cv2.COLOR_BGR2GRAY)
    # image = np.hstack((image, image, image, image, image))

    blurred = cv2.GaussianBlur(image, (17, 17), 0)
    # blurred = cv2.bilateralFilter(image, 10, 31, 31)

    inv = cv2.getTrackbarPos(TB_INVERSE, Win_NAME)
    blur = cv2.getTrackbarPos(TB_Blur, Win_NAME)
    mean = cv2.getTrackbarPos(TB_Mean, Win_NAME)
    c = cv2.getTrackbarPos(TB_C, Win_NAME)

    size = cv2.getTrackbarPos(TB_Block_Size, Win_NAME)
    if size % 2 == 0:
        size += 1

    threshold_image = adaptive_thresholding(image, blurred, mean=bool(mean),
                                            inverse=bool(inv),
                                            use_blur=bool(blur), blockSize=size, C=c)

    cv2.imshow(Win_NAME, threshold_image)
    if cv2.waitKey(1) == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Fig [7.2] Code 2

Chapter Eight

Conclusions

8.1 Conclusions

The acquisition of vein images is determined by the IR LED. In the preprocessing stage the high boost filter method is very important in strengthening the difference between the veins and the skin. To deal with noise, Gaussian filter giving a good result with 42.6 dB PSNR value. In the process of segmentation, adaptive thresholding with mean neighbor threshold value is better than gaussian threshold value. By using combination of morphology and contour area, the final result of segmentation in this research got 84.62% accuracy and 89.66% sensitivity. system and the future work is focused on adjusting the projection in term of distance change between device and object.

Future improvements:

- Detect and identify the depth and the width of veins and display the results and compare it with the standard vein's depth and width to realize if there is lesion.
- Make sure of the place where the sample will be drawn if the sample was drawn in it before or not
- Add a robot arm to draw blood samples without the need for any member of the medical crew.

References

1. Richardson, M., & Wallace, S. (2012). Getting started with raspberry PI. “O’Reilly Media, Inc.".
2. Howse, J., Joshi, P., & Beyeler, M. (2016). Opencv: computer vision projects with python. Packt Publishing Ltd.
3. Raspberry Pi Documentation - Raspberry Pi Hardware
4. Accessing the Raspberry Pi Camera with OpenCV and Python - PyImageSearch
5. Design and enhance the vein recognition using near infrared light and projector | Request PDF (researchgate.net)
6. Getting Started — Picamera 1.13 Documentation
7. Accessing the Raspberry Pi Camera with OpenCV and Python - PyImageSearch
8. https://how2electronics.com/getting-started-setting-up-raspberry-pi-4-model-b
9. https://www.researchgate.net/publication/330659359_A_Real_Time_Vein_Detection_System
10. https://www.researchgate.net/publication/327659996_Design_and_enhance_the_vein_recognition_using_near_infrared_light_and_projector
11. https://www.researchgate.net/publication/309754923_Implementation_of_Image_Processing_on_Raspberry_Pi
12. https://www.researchgate.net/publication/330472942_Vein_Visualization_System_Using_Camera_and_Projector_Based_on_Distance_Sensor
13. S. Prahl, “Optical Absorption of Hemoglobin,” 1999. [Online]. Available: https://omlc.org/spectra/hemoglobin/. [Accessed: 21-Jan2018].

14. (PDF) A Novel Method of Vein Detection with the Use of Digital Image Correlation (researchgate.net)
15. Vein Detector using Image Processing | Mergu | Recent Trends in Electronics and Communication Systems (stmjournals.in).