"*PRECIŚVEINÉ* – Non-Invasive Vein Visualization Device"

By

Hafiz Usman Mahmood	**NUST201201317BSEECS60412F**
Arslan Ahmed	**NUST201204160BSEECS60412F**
Muhammad Talal Shahid	**NUST201201382BSEECS60412F**

A Project report submitted in partial fulfillment
of the requirement for the degree of
Bachelors in Electrical Engineering

Department of Electrical Engineering

School of Electrical Engineering & Computer Science
National University of Sciences & Technology
Islamabad, Pakistan

2016

CERTIFICATE

It is certified that the contents and form of thesis entitled **"*PRECI*ŚVEINÉ – Non-Invasive Vein Visualization Device"** submitted by *Hafiz Usman Mahmood* **(*NUST201201317BSEECS60412F*),** *Arslan Ahmed* **(*NUST201204160BSEECS60412F*),** and *Muhammad Talal Shahid* **(*NUST201201382BSEECS60412F*)** have been found satisfactory for the requirement of the degree.

Advisor: _____

(Dr. Khawar Khurshid)

Co-Advisor: _____

(Dr. Awais Mehmood Kamboh)

DEDICATION

To Allah the Almighty

&

To my Parents and Faculty

ACKNOWLEDGEMENTS

I am deeply thankful to my Advisor and Co-Advisor, Dr. Khawar Khurshid, and Dr. Awais Mehmood Kamboh for extending their full help/cooperation during the course in accomplishing my final project, successfully. Their guidance, support and motivation enabled me in achieving the objectives of the project. I am also thankful to every members of the committee i.e. Mr. Habeel Ahmed and Mrs. Maira Islam for their valuable feedback. Moreover, I am also thankful to Mr. Adrian Rosebrock for his help, at all the crucial stages of the project. Finally, I am extremely grateful to my parents for giving me every moral and financial support needed for the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Non-invasive detection of a particular vein for carrying out an intravenous (IV) procedure can be cumbersome, especially on people with thin veins or who are obese or patients with damaged skin. Pediatric patients {newly born babies} also belong to this class of people because their veins are extremely thin due to their infancy. In case of extreme emergencies involving the aforementioned patients, it becomes difficult to feel/detect specific veins for carrying out the IV procedure, and the delay in treatment may become life-threatening. Clinical staff often finds it difficult to locate the desired vein in pediatric patients. More than one procedures at one instance sometimes horrifies the patient.

In the modern days, the need for a more deep research had become essential on the topic. Vein detection is among the latest techniques being researched today to resolve this issue. The technology being developed involves the use of infrared (IR) rays to illuminate a vascular region, which is then observed using an infrared camera. The technology aims to improve the healthcare standards by eliminating the stick count, and increasing the staff productivity, patient satisfaction and overall clinical workflow.

By using the aforementioned device, the results achieved were quite encouraging besides indicating the huge market potential for the technology. The results obtained showed that the technology was quite helpful in detecting the veins of quite a large number of population, where the population belonged to different age groups.

Chapter 1

INTRODUCTION

Detection of veins in patients can be very difficult depending upon various factors, such as thin veins, fatty deposits, skin color or skin damage. In case of emergencies involving aforementioned patients, it sometimes becomes difficult to carry out an IV procedure in time, and the delay in treatment may become life-threatening. Procedures involving such cases call for a solution which can help the healthcare professionals in determining the exact location of a desired vein.

According to a locally conducted survey, it has been observed that at an average, two attempts are required for finding out a proper vein to carry out IV procedures. Multiple attempts or wrongly-administered injections might result in bruising, swelling of skin, nerve damage, and blood clotting.

1.1 IMPORTANCE

To address this important issue, many strategies have been proposed. Some of these include i) Chemical paste on skin (It is not suitable for children and does not work on dark skin), (ii) Ultrasound (It needs expensive equipment and well trained staff), (iii) Secondary light sources (It requires dark room and may cause burns), (iv) Near Infrared Spectroscopy. [1]

Infrared imaging is one of the most promising areas in biomedical engineering which is least harmful to human tissues as compared to other wavelengths of electromagnetic spectrum and free from the above mentioned flaws. But the problem lies with the high cost of other similar devices available, internationally. The currently available products are manufactured at quite a high cost of around $5500, and are

generally not affordable by general practitioners who perform IV procedures regularly, especially in a developing country like Pakistan [2]. Few examples include (i) AccuVein [3] (It still demands a ribbon to be tied on arm), (ii) VeinViewer [4] (It's much bulky in size, so not easy to use and not portable), (iii) Vasculuminator [5] (Uses separate LCD to display veins instead of on the arm. This gives rise to potential chance of human error). Such short-comings of the aforementioned devices call for a technology that is not only cost effective but is also able to successfully replicate the results of the other similar technologies.

1.2 PROJECT GOALS

The major goals of this project are:

- To develop a technology that would help the healthcare professionals in performing the intravenous (IV) procedures in a risk-free way.
- To analyze the effect of infrared radiations in the blood.
- To develop a technology that is not only cost effective but also provides results similar to its competitors.

1.3 SCOPE

The scope of this project is its utilization in the healthcare industry. The device is intended to be used in emergency section of the hospitals, blood donation banks, blood testing labs where blood is drawn for testing purposes, ambulances in which cases of extreme emergencies are needed to be handled on time, and etcetera.

The device can also be used in various other areas such as cosmetic surgeries, where it is needed to avoid certain veins while performing the surgical procedures.

1.4 METHODOLOGY

The method devised to approach the solution requires the use of infrared radiation to illuminate the vascular region of interest. This image received is then

processed in the controller and sent to the display. The display can be of multiple types, where the output can be displayed on a separate LCD, a TFT touch screen mounted on the controller, or a pico-projector which would display the image back on the arm. The methods used to process the image are basically contrast enhancement algorithms, which include Windows Levelling, Contrast Limited Adaptive Histogram Equalization (CLAHE). These algorithms are implemented using the OpenCV, whereby OpenCV provides extremely efficient algorithms for image processing.

1.5 REPORT ORGANIZATION

The report is organized into following chapters:

Chapter 2 *explains the application of engineering in medicine and the role of engineering in further mitigating the technological issues being faced by the medical community.*

Chapter 3 *presents the literature review done to achieve the basic understanding of infrared based imaging systems.*

Chapter 4 *presents the methodology devised to obtain the desired results.*

Chapter 5 *gives explanation about OpenCV, Python and Raspberry Pi.*

Chapter 6 *gives design details and the implementation methods.*

Chapter 7 *discusses the results obtained by the implementation of design.*

Chapter 8 *gives conclusion and future recommendations.*

Chapter 2

ENGINEERING IN MEDICINE [6]

Biomedical Engineering (BME) is the application of engineering principles to the field of medicine and healthcare. This field of engineering seeks to bridge the gap between engineering and medicine by providing healthcare solutions such as diagnosis and treatment. Previously, it was thought that BME is a branch of some major engineering domain, such as Electrical Engineering or Mechanical Engineering, and etcetera. But now, it has revolutionized so much as a field that it is considered as totally complete field in itself. Much of the work being done in this field is research and development. The major biomedical engineering applications includes the development of imaging systems such as MRI and CT scan machines, EEG, pharmaceutical drugs, and various prosthetic equipment.

2.1 ORIGINS OF BIOMEDICAL ENGINEERING

Biomedical engineering has existed for thousands of years, but it was actually during the Second World War that it was applied to practice. Biologists were needed to do work related to development of prosthetic equipment for wounded soldiers, which led them to electronic development in medicine. However, since the biologists were not experts in the field of technology, they couldn't actually understand it. So, a bridge was needed between technical knowledge and biology. Doctors and biologists who were interested in engineering and technology, and the engineers who were interested in biology became the first bioengineers. Those primarily concerned with medicine became the first biomedical engineers.

2.1.1 Major Milestones in BME

The achievements in BME range from early devices such as wooden teeth, platform shoes and others to more advanced equipment such as dialysis machine, imaging systems, artificial organs and advanced prosthetics.

- In 1895, Conrad Roentgen of Germany discovered X-Rays using gas discharged tubes.
- In 1896, Henry Becquerel from France discovered that X-Rays were emitted from uranium ore.
- In 1903, William Eindhoven invented the electrocardiogram (ECG).
- In 1929, Hans Berger invents EEG.
- 1940 saw the Cardiac Catheterization.
- In 1950, Electron Microscope was invented.

The aforementioned medical devices that were developed afterwards essentially provide solution for all the problems that were not solved by previously available devices. Previous devices used some sort of biological or chemical means that sometimes did more harm than good.

A medical device is intended for use in:

- The diagnosis of disease or other conditions, or
- In the cure, mitigation, treatment, or prevention of disease.

Some examples of the devices that are currently being used to perform the above mentioned tasks are infusion pumps, the heart-lung machine, dialysis machines, artificial organs, implants, artificial limbs, corrective lenses, dental implants, and etcetera.

Chapter 3

LITERATURE REVIEW

3.1 MEDICAL IMAGING SYSTEMS [7]

Biomedical imaging systems consist of a major portion of medical devices employed in the medical field. These devices enable the clinicians to view the things that are not visible by the naked eye. This involves the utilization of different waves from electromagnetic spectrum including ultraviolet rays and X-Rays.

Imaging technologies are essential for the diagnosis of certain diseases which are complex to diagnose otherwise. The most complex equipment found in the hospitals include the MRI machine, CT scan machine, X-Rays, electron microscopy and optical microscopy.

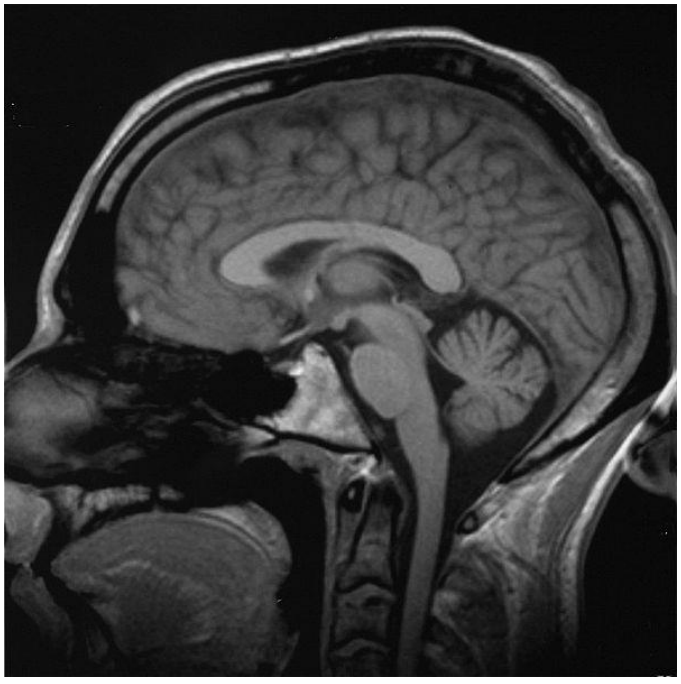


Fig 1. MRI scan of human head, an application of imaging systems [8]

3.1.1 Infrared Physics and Technology [9]

Abnormal body temperature is a natural indicator of illness. Infrared thermography (IRT) is a fast, passive, non-contact and non-invasive alternative to conventional clinical thermometers for monitoring body temperature. Besides, IRT can also map body surface temperature remotely. Last five decades witnessed a steady increase in the utility of thermal imaging cameras to obtain correlations between the thermal physiology and skin temperature. Moreover, IRT is not only limited to thermal imaging systems. Instead, IRT has been successfully used in diagnosis of breast cancer, diabetes neuropathy and peripheral vascular disorders. It has also been used to detect problems associated with gynecology, kidney transplantation, dermatology, heart, neonatal physiology, fever screening and brain imaging. With the advent of modern infrared cameras, data acquisition and processing techniques, it is now possible to have real time high resolution images, which is likely to surge further research in this field.

All objects with temperature above absolute zero emit electromagnetic radiations, which are known as infrared radiations or thermal radiations. Wavelength of this radiation lies within the range of 0.75-1000 μm. This wide range can be further subdivided into three smaller groups which are classified as near infrared or NIR, medium infrared or MIR, and far infrared or FIR. Each class of infrared rays have different applications where they are suited best.

The major components of an infrared imaging system includes an infrared radiator which radiates an infrared light of a particular wavelength, and an infrared camera, which captures the image only in the infrared region. This sort of camera only allows the infrared light to pass through, and blocks all the visible light.

3.1.2 Blood and Infrared Radiations

In the project, it is required to find out the connection between the vascular blood and infrared radiation. There are two types of blood flowing in the human body in terms of the hemoglobin presence. These are:

- Oxy-hemoglobin (Oxygenated blood)
- Deoxy-hemoglobin (De-oxygenated blood)

The oxygenated blood is one that is pumped out by the heart after it has passed through the lungs. The lungs add oxygen to the blood, which then flows towards heart and the heart then pumps it out to the whole body. This blood then flows through the capillaries in the nervous system to the whole body. The blood then starts flowing in the veins present in the body, like arm, legs, head, and etcetera.

In the veins, the blood starts loosing oxygen to the surrounding tissues as a result of metabolism by cells. This type of blood is known as de-oxygenated blood. This blood contains carbon dioxide, a waste product due to metabolism, which is sent to lungs and is exhaled.

The de-oxygenated blood has a property that it can absorb the infrared radiation. Hence, when a particular vascular region containing de-oxygenated blood is illuminated with the help of infrared radiations, the blood absorbs the radiation and the surrounding tissues reflect back the light. This reflected light is then captured by using an infrared (IR) camera which sends the image to controller for further processing.

Researches state that the level of absorption of infrared light in the blood is determined by the absorption coefficient of the IR light into the blood. The absorption coefficient determines how much of the incident IR wavelength has been absorbed by the blood. However, the absorption level of IR light into oxygenated and de-oxygenated blood is different, for same wavelength and for same amount of power of incident light. This can be observed from the figure below [10]:

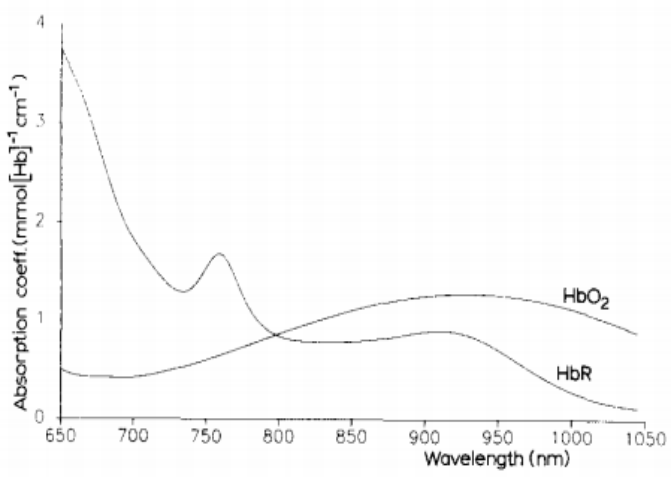


Fig 2. Absorption Coefficients of Oxy-Hemoglobin (HbO_2) and Deoxy-Hemoglobin (Hb)

From the above figure, it can be observed that the absorption of IR wavelength having a wavelength of 750 nm is maximum.

3.1.3 Use of IR radiations in Vein Detection

Intravenous (IV) procedures performed by the nurse are among the most painful and cumbersome procedures carried out daily in hospitals and blood testing labs. Although IV procedures are a common practice, the procedure sometimes gets nurses into trouble as not all the first attempts are successful. Moreover, since the IV procedures sometimes cause anxiety among the patients, these procedures frequently induce patient reactions to make IV procedures appear more difficult to perform [11].

As previous researches have mentioned, there are several factors that affect the IV procedures, such as nurse's younger age and inexperience, patient movement, patient dark skin, the vein is not palpable, or the patient's vein is not at all visible. However, the most significant factor causing the failure of IV procedures is the non-visibility of veins.

Human eye can only detect the waves belonging to visible region of electromagnetic spectrum, which ranges from 400 nm to approximately 700 nm.

However, there is more information contained in other regions of EM spectrum which can be used for a variety of purposes. For example, For example, X-Rays are being used in chest radiology but X-rays cannot be used in visualizing veins. However, near-infrared (NIR) spectroscopy has been found to solve the issue and can help in visualizing the veins because they can penetrate into the human skin to a depth of about 5 mm before scattering and deviating from initial direction [12].

Currently, there are some devices available internationally, such as AccuVein's AV400, Christiemed's VeinViewer, and Vasculuminator's Vision. These devices are not only costly but also contain some shortcomings as mentioned in chapter 1. The purpose of development of the project is to not only reduce the size of the prototype but also make it cost effective.

3.2 DELIVERABLES

The final deliverable would be a fully working prototype, which would not only be cost effective but also light weight. It will also be portable for easy transportation purposes. The major components of the final deliverable prototype would contain an array of IR LEDs, an IR camera, an image processing unit and a pico-projector.

Chapter 4

METHODOLOGY

As mentioned in previous chapters, the blood flowing in the veins can absorb the IR wavelength. This property of blood has been exploited for the purpose of development of device.

The block level functionality of the device has been shown as follows:

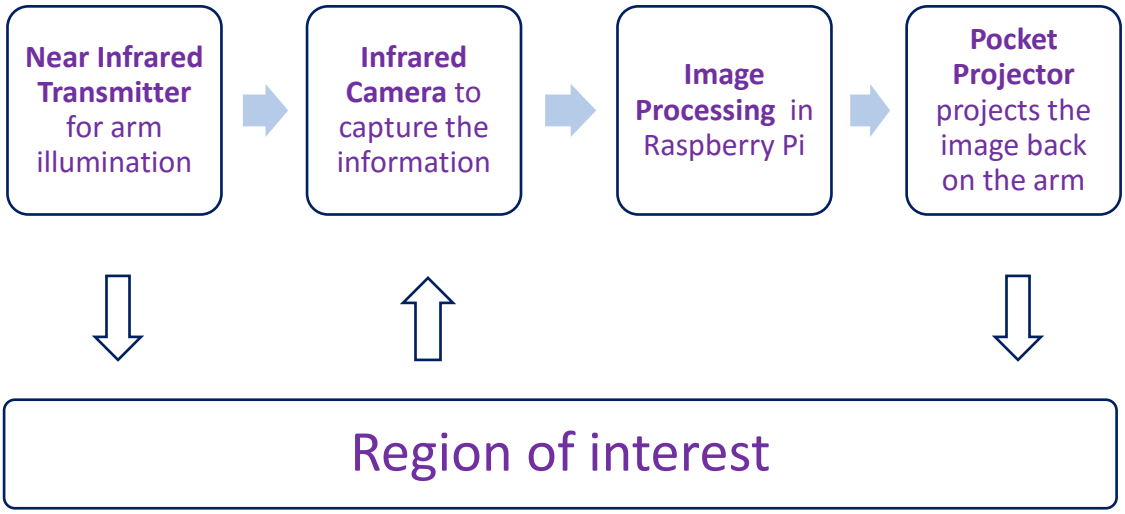


Fig 3. Block Diagram of Prototype

4.1 NEAR INFRARED (NIR) TRANSMITTER

An IR transmitter is used for illuminating the arm under consideration. The wavelength being used lies in near infrared region, which stretches from about 750 nm to 900 nm. In this region, the absorption of IR wavelength in the blood is maximum as given in figure 2.

4.2 INFRARED CAMERA

Infrared camera is a special purpose camera being used in this project. It is a simple camera with its infrared filter, which is used to block the infrared light, removed and the visible light filter, which is used to block the visible light, inserted in front of the charge coupled device (CCD) and behind the lens of the camera. Hence, this special purpose camera has the ability to block all the visible light and let all the infrared light to pass through it.

4.3 RASPBERRY PI AS IMAGE PROCESSOR

Raspberry Pi is being used as an image processing unit. Raspberry Pi is a linux-based computer board, which can be used as a standalone device for embedded system applications. More details on Raspberry Pi and its operating system would be discussed in next chapters.

4.4 PICO PROJECTOR

Pico-projector is a palm sized projector which is used to project the image back on the arm. Since the central point of the image being captured by the camera is different than that being projected back, hence some sort of calibration is needed to make the central point of captured image and the projected image same.

For example, if the camera is capturing the image from some area whose corner points are mentioned as (a,b,c,d), then it is necessary that the projector projects the image back to the very same region having the corner points as (a,b,c,d).

4.5 GENERAL CONCEPT OF VEIN DETECTION

The general concept behind the vein detection is that the deoxygenated blood has the property, as mentioned in previous chapters, of absorbing the infrared light. When a particular vascular region is illuminated with the IR source, the light gets absorbed into the blood and the surrounding tissues reflect back the light. This light is captured by the infrared camera and the captured image is sent to raspberry pi for further processing. The general processing being done is contrast enhancement. After doing contrast enhancement, the resulting image is sent to the display. Since all of this

is to be done in real time, hence about 24 frames are captured per second and the processing is done frame by frame. Each frame is taken as an image, the processing is done and then sent to the output for display.

4.6 IMAGE PROCESSING ALGORITHMS

The most important part of this project is image processing, which is being done in Raspberry Pi using OpenCV. Details on both of them will be given in subsequent chapters. As for now, the image processing algorithm used to enhance the image is histogram equalization. This algorithm enhances the contrast of the image by stretching the histogram of a 2-dimensional image over the whole range of pixel intensities. The details on histogram equalization are given as follows:

4.6.1 Histogram Equalization [13]

Histogram Equalization is a method used in image processing that is used to enhance the contrast of a grayscale image over the whole range of pixel intensities. This is done using the image's histogram.

The implementation of histogram equalization is done by following the given procedure:

Consider a discrete grayscale image {x} having a total 'n' number of pixels. Let 'i' represent a particular gray level. So, n_i would be total number of pixels corresponding to a particular gray level. If 'L' is the total number of gray levels in an image (typically for an 8-bit image, L=256), then:

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 < i < L$$

The above equation determines the probability of the occurrence of a pixel with a particular gray level i, in an image. Typically, the above equation normalizes the image's histogram from [0, 1].

The cumulative distribution function (cdf) corresponding to p_x can be defined as:

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

Which is also the image's accumulated normalized histogram. The above function basically accumulates the total number of pixels up to a particular gray level *i* and assigns the corresponding cumulated value to that particular intensity. For example, if an image contains a gray level with value 54, which has 50 corresponding pixels, and the total number of pixels before this particular gray level, i.e. 54, is also 50, then the cdf value for gray level i=54 would be 100, i.e. the sum of total pixels up to i=54.

A transformation of the form *g=T(x)* is needed to produce a new image 'g' with a flat histogram as compared to a more concentrated and peaking histogram of image 'x'. Such an image 'g' would have a more linear cumulative distribution function.

The general histogram equalization formula is:

$$h(v) = round(\frac{cdf(v) - cdf_{min}}{(MxN) - cdf_{min}} x(L-1))$$

The above mentioned formula normalizes the histogram from any range [x, y] to [0,255], where: *0<x<y<255*

For example, consider a 2x2, 8-bit image {x} with pixel values given as follows:

$$\begin{bmatrix} 54 & 83 \\ 83 & 157 \end{bmatrix}$$

The above given 2x2 image has 2-pixels corresponding to gray level 83, and one pixel each corresponding to 54 and 157. The histogram for the above image in tabular form can be shown as follows:

Value	Count
54	1
83	2
157	1

The cdf is shown below:

v, Pixel Intensity	*cdf(v)*	*g(v), Equalized v*
54	1	0
83	3	170
157	4	255

Table -1: Cumulative Distribution Function (CDF)

Applying the general histogram equalization formula to above pixels:

$$g(54) = round\left(\frac{1-1}{4-1}x255\right) = 0$$

$$g(83) = round\left(\frac{3-1}{4-1}x255\right) = 170$$

$$g(157) = round\left(\frac{4-1}{4-1}x255\right) = 255$$

Hence, after equalization, the new image obtained has pixel intensities as follows:

$$\begin{bmatrix} 0 & 170 \\ 170 & 255 \end{bmatrix}$$

It can be seen that the histogram of the image has been stretched over the whole range of pixel intensities, i.e. a gray level corresponding to 54 has now become

completely black at a value 0 and a gray level corresponding to 157 has now become completely white at 255.

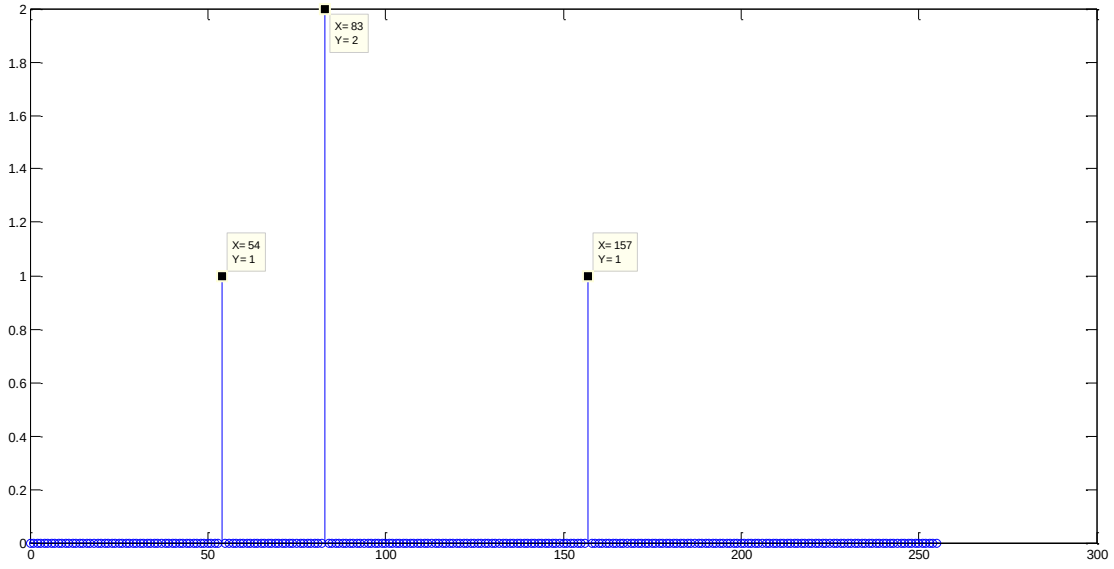


Fig 4. Histogram of Original Image, *x*

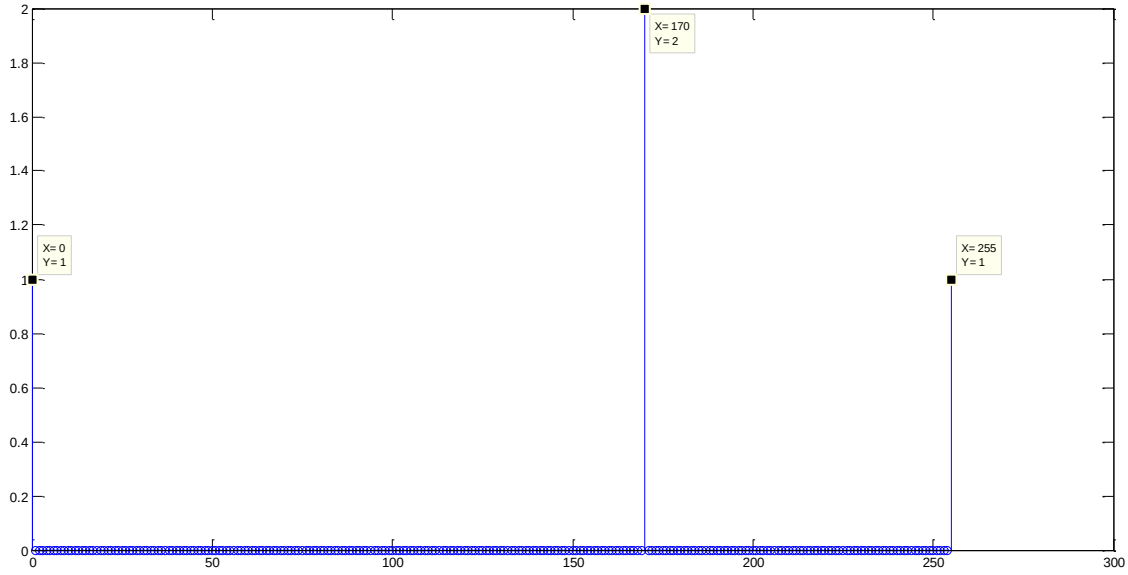


Fig 5. Histogram of Transformed Image, *g*

One of the major drawbacks of histogram equalization is that it enhances the global contrast of the image. This means that if there is some noise in the image, it will also be enhanced. Furthermore, if the gray levels in an image are far apart, then this method fails.

4.6.2 Adaptive Histogram Equalization (AHE)

Unlike the normal histogram equalization which enhances the global contrast of the image keeping in view all the pixel intensities, the adaptive histogram equalization enhances the local contrast of the image. AHE does this by computing the several histograms, each corresponding to a distinct section of an image.

However, AHE has the tendency to over-amplify the noise, if any, present in the image. This would produce unnecessary distortions in the resultant image. To get rid of this problem, another variant of adaptive histogram equalization is used which is know as Contrast Limited Adaptive Histogram Equalization (CLAHE).

4.6.3 Contrast Limited Adaptive Histogram Equalization (CLAHE)

CLAHE differs from ordinary AHE in its contrast limiting, as the name suggests. In CLAHE, the contrast limiting procedure is applied to each neighborhood from which a transformation function is derived. CLAHE has the tendency to reduce the over-amplification of noise.

This reduction in amplification of noise is achieved by limiting the contrast of AHE. CLAHE limits the amplification by clipping the histogram of the image before computing its cumulative distribution function (cdf). Hence, CLAHE performs two functions before applying histogram equalization:

- Takes a region in an image having a grid size of MxN.
- Clips the histogram before computing the cdf.

In this way, the clip limit helps to reduce the noise by reducing the slope of the distribution function. By defining the clip limit, all the values above the clip limit are

clipped and are evenly distributed over the whole range of pixel intensities. This reduces the noise and further enhances other pixel intensities.

4.6.4 Window/Level [14]

Window/Level is another contrast enhancement algorithm where a linear scaling function is applied to a particular window of histogram of the image. This means that, only particular pixel intensities are scaled to a new value, whereby these values are scaled only in a tight window. All the pixel intensities outside this window are either scaled down to zero or scale up to maximum value, i.e. 255 in case of an 8-bit image.

The basic concept behind the application of window/level is to apply a linear grayscale function, in form of a lookup table (LUT), specified by two parameters, window and level. Window describes the width of the window ranging from minimum pixel intensity to maximum pixel intensity whose values are to be transformed. Level basically describes the midpoint of this window.

In the following figure, it can be observed that for a given input image with a particular histogram, application of window and level produces the output histogram as follows:

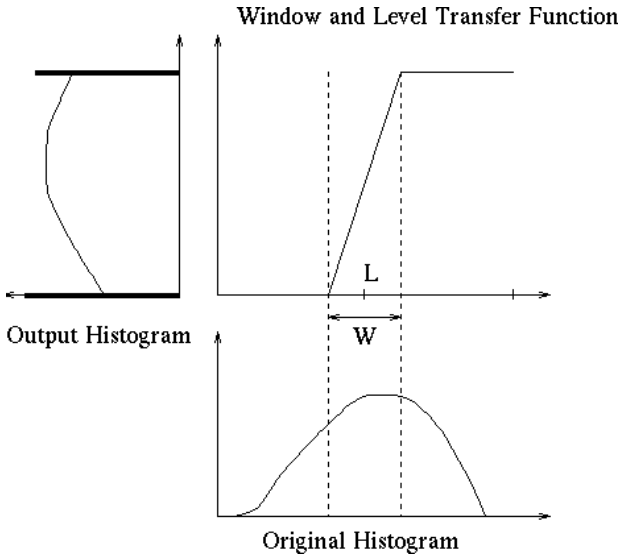


Fig 6. Window and Level Transfer Function [15]

One way of specifying a linear grayscale transform in figure 6 is to define the width of the LUT where the slope is non-zero (window) and the center of that same segment of the LUT (level). So holding the window constant while changing the level has the effect of moving the window to left of right which causes different sorts of contrast enhancements each time the level is changed. Similarly, if the level is kept constant and the window is changed, then the pixel range to be highlighted is either broadened or narrowed.

The general algorithm applied for window/level is:

For a q-bit image;

$A = 2^q - 1$

LUT[] = int(A+1) (Array to store the grayscale transform function)

a = level – window/2

b = level + window/2

for each i = 0 to a-1 (1^{st} portion all zeros)

    LUT[i] = 0

end

for each i = a to b (2^{nd} portion linear scaling function)

    LUT[i] = round[(A/window)(i-a)]

end

for each i = b+1 to A (3^{rd} portion – maximum pixel intensity)

    LUT[i] = A

end

In designed project, the algorithm used for contrast enhancement is **CLAHE.**

Chapter 5

OPENCV, PYTHON and RASPBERRY PI

5.1 OPENCV

Computer Vision is an area of computer science, mathematics and electrical engineering. It includes ways to acquire, process, analyze and understand the images and videos from the real world in order to copy the human vision. Furthermore, computer vision can also be used to analyze and process depth and infrared images.

Computer Vision is also concerned with the theory of information extraction from the images and videos. A computer vision system can accept different forms of data as input and extract information from that data. This data is not only limited to images and videos but also other signals.

The typical task of computer vision includes the object recognition and classification, motion detection and analysis, and image reconstruction.

OpenCV (Open Source Computer Vision) is a library of programming functions for computer vision. This is a cross-platform library, which means that it can be implemented and operated on different operating systems. It focuses mainly on image and video processing. Furthermore, it also contains GUI features for user convenience.

5.1.1 History of OpenCV

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five beta versions were released between 2001 and 2005. The first 1.0 version was released in 2006.

The second major release of OpenCV was in October 2009. OpenCV 2.0 includes easier, type-safe patterns, newer functions and better optimization on previously present functions.

5.2 PYTHON

Python is one of the most widely used high-level, general purpose programming language. Its design philosophy implies that the code must be easily readable, and it must give the programmers complete freedom to express the concepts in fewer lines of code as compared to C/C++ or Java.

Python supports multiple paradigms including the object oriented approach, and the conventional structured and sequential programming approach.

5.2.1 OpenCV-Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Compared to languages like C/C++, Python is slower. This implies that Python can be easily extended with C/C++, which allows to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This provides 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 is easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

5.2.2 NumPy

NumPy is a library for the Python programming language that (among other things) provides support for large, multi-dimensional arrays. Using NumPy, we can express images as multi-dimensional arrays. Representing images as NumPy arrays is not only computational and resource efficient, but many other image processing libraries use NumPy array representations as well. Furthermore, by using NumPy's

built-in high-level mathematical functions, we can quickly perform numerical analysis on an image.

5.3 RASPBERRY PI

Raspberry Pi is a series of low-cost, palm size, single board computers developed by Raspberry Pi Foundation, UK. The basic intention behind the development of such a platform was to promote the teaching of basic computer skills among the school children. Raspberry Pi has, since then, expanded its footprint well beyond its intended purpose by penetrating into the market of embedded systems and research.

Raspberry Pi models – A, A+, B and B+ - are based on BroadCom BCM2835, which includes an ARM11 700 MHz CPU. This speed can be overclocked. Pi 2 has a quad core ARM Cortex A7 processor. Raspberry Pi 2 has 1 GB of RAM

Raspberry Pi Foundation provides Debian and Arch variants and Linux ARM distributions for download. Python is the main programming platform and languages like C++, Java and Ruby can be used to program Raspberry Pi.

5.3.1 Operating System

Raspberry Pi primarily uses Unix-like, Linux-kernel-based operating systems, like the variants of Debian and Fedora.

The Raspberry Pi 2 is based on ARM Cortex A7, which is capable of running both Windows 10 and Ubuntu (Snappy Core). The following operating systems are officially supported by all models of Raspberry Pi:

- OpenELEC
- Pidora

- RASPBMC
- RISC OS
- Raspbian

5.3.2 Raspbian

Raspbian is an unofficial variant of Debian Wheezy that is compile for hard float code that will run on Raspberry Pi computers. It is a free operating system based on Debian that is optimized for Raspberry Pi hardware. Raspbian comes with over 35,000 packages and a precompiled software for Raspberry Pi.

Chapter 6

DESIGN IMPLEMENTATION

The implementation of design requires a certain number of prerequisites like installing Raspbian OS on Raspberry Pi 2 and installing OpenCV 3 with Python bindings on Raspberry Pi 2. The hardware design required a number of components, which are already mentioned in chapter 4.

6.1 INSTALLATION OF RASPBIAN OS [16]

For installing Raspbian OS along with OpenCV, it is required to have at least 16GB microSD card. After this is ensured, following steps must be followed to install Raspbian on Pi:

1. Go to https://www.raspberrypi.org/downloads/noobs/ and download the NOOBS file in .zip format.
2. After it has been downloaded, extract the files in a specific folder.
3. Download the recommended SD card formatter from the following website: https://www.sdcard.org/downloads/formatter_4/
4. Install the SD formatter and open it.
5. Using a memory card reader, insert the microSD card in PC and format it using the formatter.
6. After format is complete, copy all the extracted files to the microSD card.
7. Plug the memory card into Raspberry Pi 2 memory jack.
8. For the first time installation of Raspbian OS on Pi, it is required that Raspberry Pi be connected with a TV having HDMI support.
9. Connect Raspberry Pi with TV using an HDMI cable.
10. After the TV is connected, a GUI will appear which will help the user to install Raspbian OS on Raspberry Pi. The installation will complete in about 20 minutes.

After installation is complete, the OS is good to go. Now if someone wants to keep working on TV, it's fine. If someone wants to use a monitor as a display panel, follow the following commands:

1. Go to terminal and type the following command to edit the configuration file:

   sudo nano /boot/config.txt

2. Edit the following lines: (The new lines should be set as follows)

   hdmi_force_hotplug=1

   hdmi_group=2 (Sets display mode to DMT group)

   hdmi_mode=16 (Sets the resolution to 1024x768)

   hdmi_drive=2

   Remove the # sign before these, as # sign represents the commented lines.

3. Press Ctrl+X, Y, and then press Enter to save the configuration file.
4. Shutdown the OS.
5. Take an HDMI to VGA converter to connect Raspberry Pi with the VGA monitor.
6. Restart Pi.

The editing of configuration file is needed because if it is not done, then monitor will not support the display and nothing will appear on the screen.

6.2 INSTALLING OPENCV AND PYTHON BINDINGS [17]

After installing the OS, the next important step in setting up the Raspberry Pi for our project is installation of OpenCV and Python bindings. The tutorial by Adrian Rosebrock from *pyimagesearch* was followed for this purpose. The prerequisite for installing OpenCV and Python is an internet connection. The Raspberry Pi must be connected to internet before moving on to installing OpenCV. After internet connection has been ensured, follow the following steps to install OpenCV:

1. Open the Terminal.
2. Type the following commands:

   sudo apt-get update

   sudo apt-get upgrade

   sudo rpi-update

   After the process is complete, Raspberry Pi needs to be rebooted.
3. '**sudo reboot**' is the command to reboot the Raspberry Pi.
4. A few developer tools are then required which can be installed using:

   sudo apt-get install build-essential git cmake pkg-config
5. Just like image I/O packages are needed, video I/O packages are also needed. They can be installed using the following commands:

   sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

   sudo apt-get install libxvidcore-dev libx264-dev
6. GTK development library is also needed so we can compile the 'highgui' sub-module of OpenCV, which allows us to display images to our screen and build simple GUI interfaces:

   sudo apt-get install libgtk2.0-dev
7. Some added dependencies are also required:

   sudo apt-get install libatlas-base-dev gfortran
8. Lastly, install the Python 2.7 and Python 3 header files so we can compile OpenCV + Python bindings:

   sudo apt-get install python2.7-dev python3-dev
9. To get the 3.0.0 version of OpenCV from OpenCV repository, type the following commands:

   cd ~

   wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.0.0.zip

   unzip opencv.zip
10. Furthermore, if full install of OpenCV is required, one must also grab the opencv_contrib repository as well:

wget –0 opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.0.0.zip

unzip opencv_contrib.zip

11. To setup Python for OpenCV, install pip, a Python package manager:

    wget https://bootstrap.pypa.io/get-pip.py

    sudo python get-pip.py

12. The next step is to install the virtualenvwrapper package which helps to create the isolated python environments. This means that one can run multiple versions of Python, with different versions of packages installed into each virtual environment. To install these packages, type the following commands:

    sudo pip install virtualenv virtualenvwrapper

    sudo rm -rf ~/.cache/pip

13. After virtualenv and virtualenvwrapper have been installed, ~/.profile needs to be updated. .profile is a file that contains all the necessary information that needs to be loaded on bootup. So open the .profile using the following code:

    nano .profile

    After this, add the following lines in the end of the file:

    export WORKON_HOME=$HOME/.virtualenvs

    source /usr/local/bin/virtualenvwrapper.sh

14. Now that the file has been updated, one needs to reload it so that the changes can take effect. This can be done by using the following command:

    source ~/.profile

15. The next step is to create our python virtual environment where we'll be doing our computer vision work:

    mkvirtualenv cv

    This command will create a virtual environment named **cv** using Python 2.7.

16. If the system is rebooted anytime, one needs to enter the following commands to enter into the virtual environment to work on OpenCV:

    Open the terminal and enter the following commands to enter into the virtual environment:

source ~/.profile

workon cv

17. Once in the cv virtual environment, NumPy can be installed, which is an important dependency when compiling Python bindings for OpenCV.

    pip install numpy

18. After numpy has been installed, OpenCV is ready to be compiled. Make sure that we are operating in cv virtual environment by typing the following command:

    workon cv

19. Type the following commands:

    cd ~/opencv-3.0.0/

    mkdir build

    cd build

    **cmake -D CMAKE_BUILD_TYPE=RELEASE **

    **-D CMAKE_INSTALL_PREFIX=/usr/local **

    **-D INSTALL_C_EXAMPLES=ON **

    **-D INSTALL_PYTHON_EXAMPLES=ON **

    **-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.0.0/modules **

    -D BUILD_EXAMPLES=ON ..

    After this, OpenCV is ready for compilation. Be sure to be in cv virtual environment. After it has been ensured, type the following command:

    make –j4

    The –j4 switch implies that all four cores of raspberry pi are being used for a more quick compilation of OpenCV.

20. If OpenCV has compiled without error, it must be installed on our system:

    sudo make install

    sudo ldconfig

21. Type the following command:

    ls -l /usr/local/lib/python2.7/site-packages/

22. The last step is to sym-link the OpenCV bindings into the cv virtual environment:

    cd ~/.virtualenvs/cv/lib/python2.7/site-packages/

    ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so

23. To verify that OpenCV has been installed for our Python, enter the following commands:

    workon cv

    python

    >>> **import cv2**

    >>> **cv2.__vesion__**

    The output should be:

    '3.0.0'

    which implies that the OpenCV version 3.0 has been installed successfully.

6.3 HARDWARE DESIGN

The hardware design requires the following components:

1. Raspberry Pi 2 with Python and OpenCV installed.
2. Raspberry Pi NoIR camera with visible light filter (A small piece of floppy disk tape, a fully exposed negative, or a dark arylic glass sheet). In our design, we implemented the visible light filter using the floppy disk piece which was installed behind the lens and in front of the CCD.
3. Specially designed LED array for Raspberry Pi. Each of the LEDs have a wavelength of 850 nm.
4. An HDMI output pico-projector.

The camera was connected to Raspberry Pi using a CSI connector cable. The cable from camera was connected to the CSI connector on the Raspberry Pi board.

Hardware Design

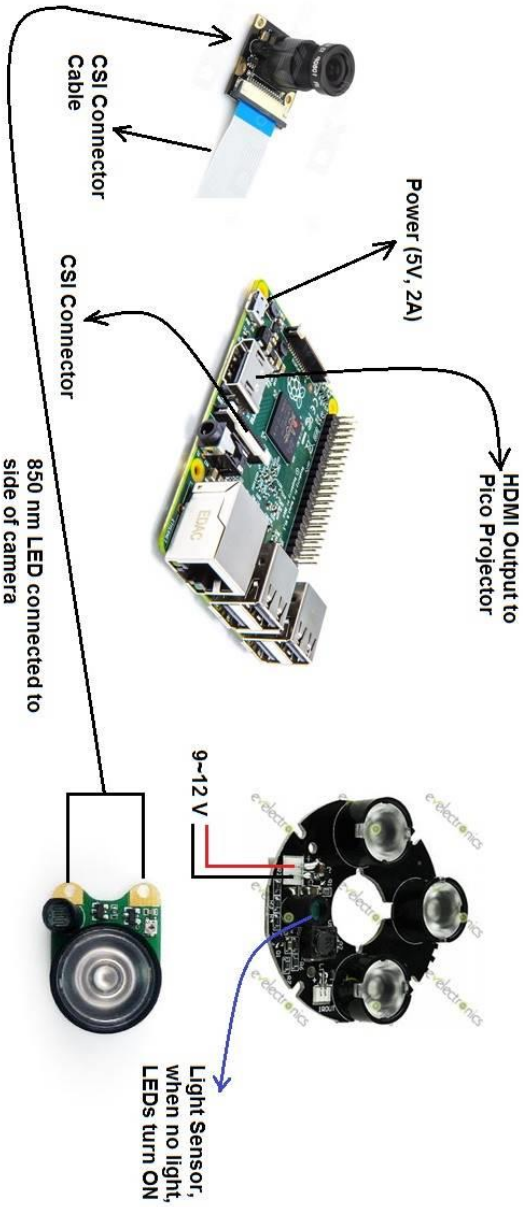


Fig 7. Hardware Connections

The hardware box in which the whole assembly is enclosed is designed as follows:

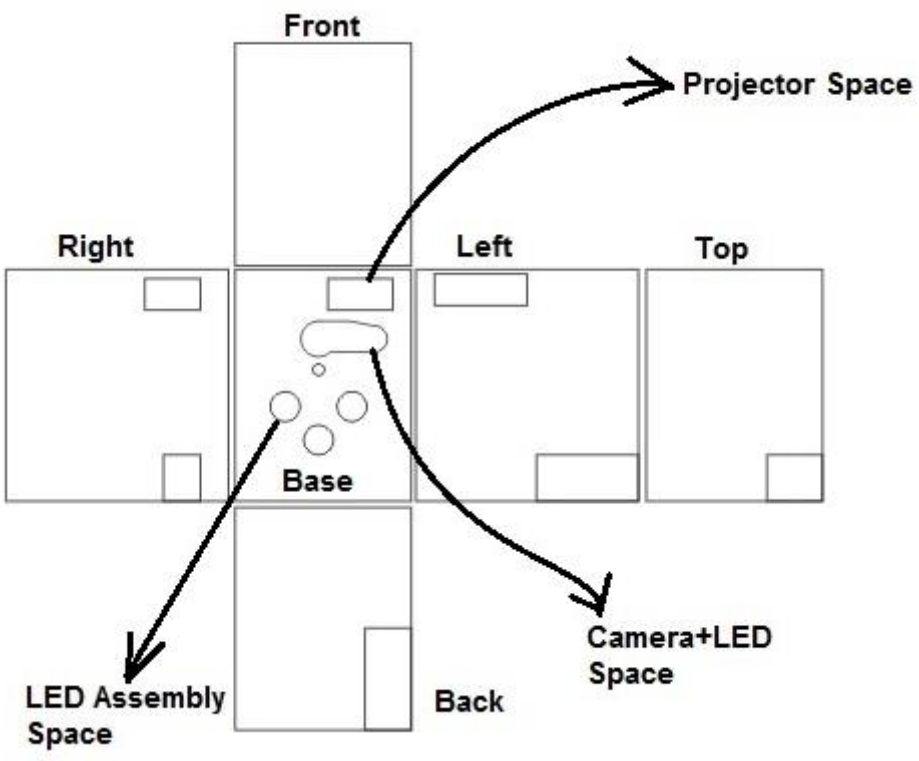


Fig 8. Hardware Assembly

The above given hardware was implemented using the following measurments:

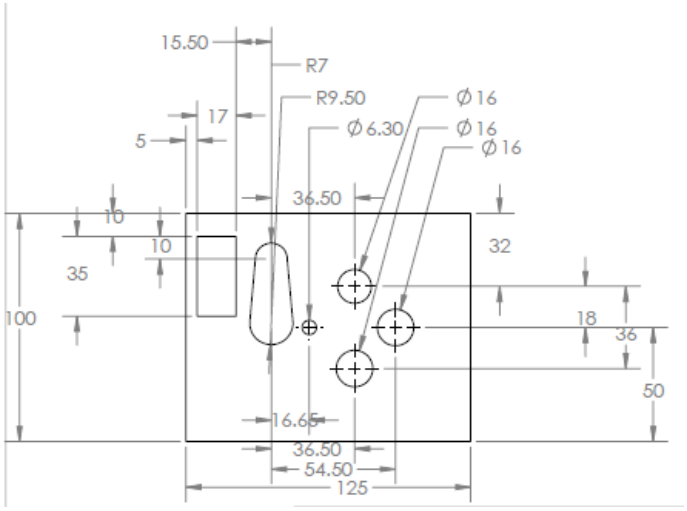


Fig 9. Hardware Assembly's Base measurements

All the readings are in millimeters.

The final hardware looked like:

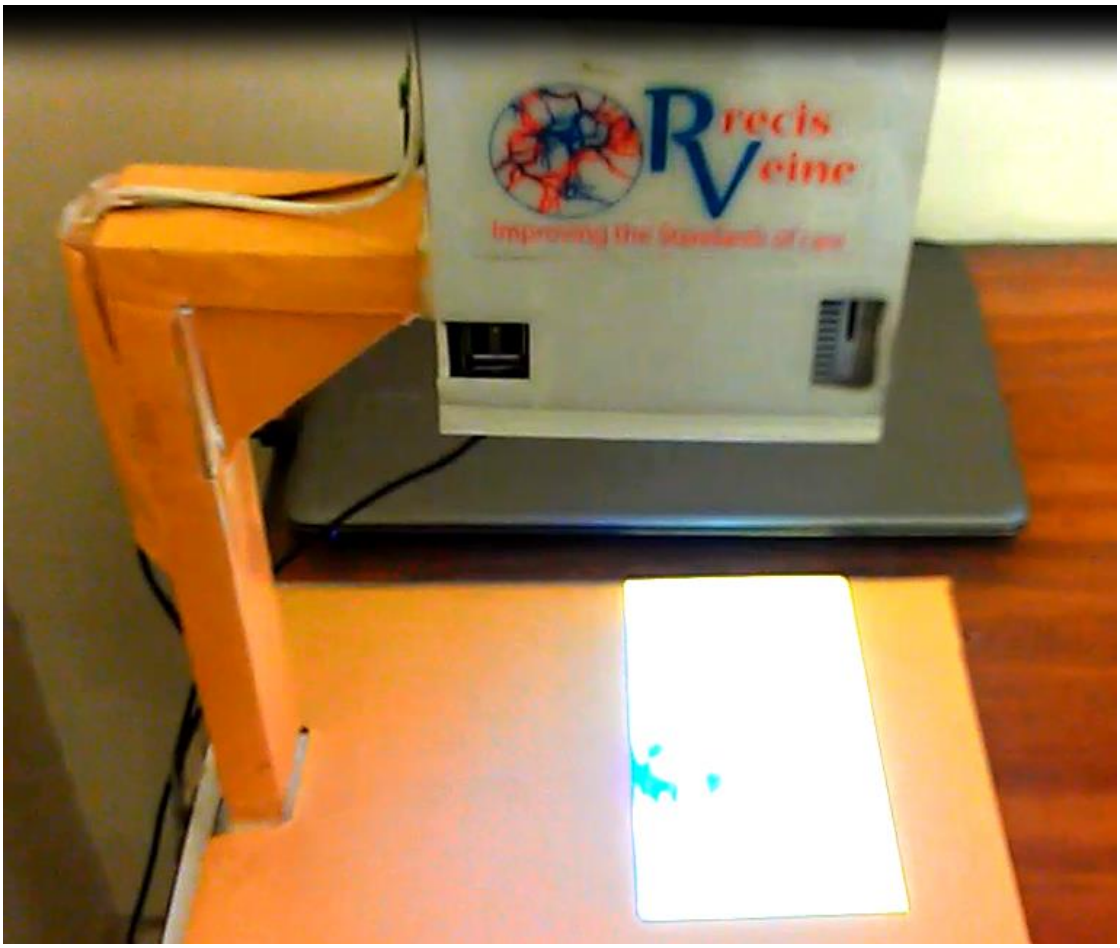


Fig 10. Final Hardware Prototype

6.4 Automating the Raspberry Pi

In order to make the Raspberry Pi operate in a standalone mode, it is necessary that Pi be operated in such a way that when it boots up, it starts to execute the above code automatically without any requirement to enter extra commands. To do so, follow the given procedure:

1. Make a shell script with extension .sh using the following command. Give the filename as launcher.

nano launcher.sh

2. Write down the following script:

   #! /bin/sh

   cd /

   cd /home/pi

   sudo /home/pi/.virtualenvs/cv/bin/python filename.py (test_video.py is your python script)

   cd ~

3. Press Ctl+X, Y and then Enter to save the file.
4. To make the shell script executable, type the following command:

   chmod 755 launcher.sh

   To test whether the file is executable, type:

   sh launcher.sh

   This should run the python script.

5. Open the following files using the terminal:

   sudo nano /etc/profile

   and

   nano .profile

6. Enter the following command to the end of each of these files:

   /home/pi/launcher.sh

   Now, when the Raspberry Pi will boot up, it will automatically run the specified python script.

Chapter 7

RESULTS

The project was done in different phases. These included the results taken from a normal camera, results taken from infrared camera and processed in MATLAB, results taken from IR camera and processed using OpenCV. Each of these phases helped us achieve different insight into the project, and how the results from previous stages could be improved further. These results in graphical form as mentioned below:

7.1 PHASE 1

Fig 11. Results of Phase 1

The results in phase 1 were not very much encouraging as were not able to detect veins. This was due to the fact that neither the IR camera was being used nor any sort of post processing was being applied.

7.2 PHASE 2

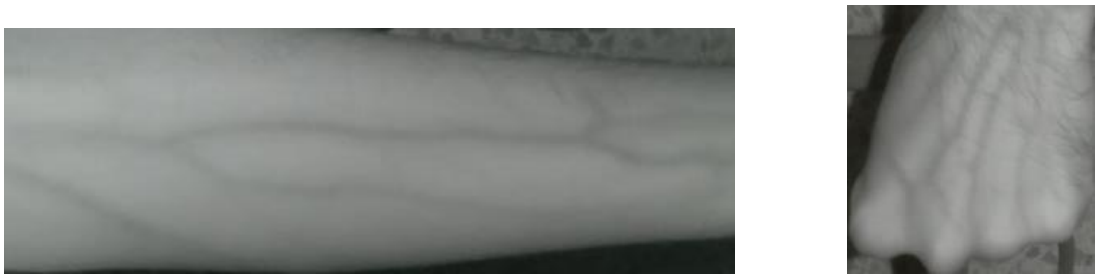


Fig 12. Results of Phase 2 (Arm – Left, Hand – Right)

The results in second phase were quite encouraging. We were able to detect veins of a sample because we were using the night vision camera. A night vision camera is somewhat different from the infrared camera in a way that night vision camera has neither visible light nor infrared light filter installed. So it allows both the visible and infrared light to pass through it. The results were displayed on a LCD monitor. The method to connect LCD monitor with Raspberry Pi is given in previous chapters.

It can be observed that the results of second phase appear to be better than that of the first phase, even though no image processing was applied.

7.3 PHASE 3

Phase 3 of the project was carried out by doing post processing of the image. Histogram Equalization or Window/Level was performed on the images, which increased the contrast of the images. The resulting images had darker areas become darker and the lighter areas become lighter.

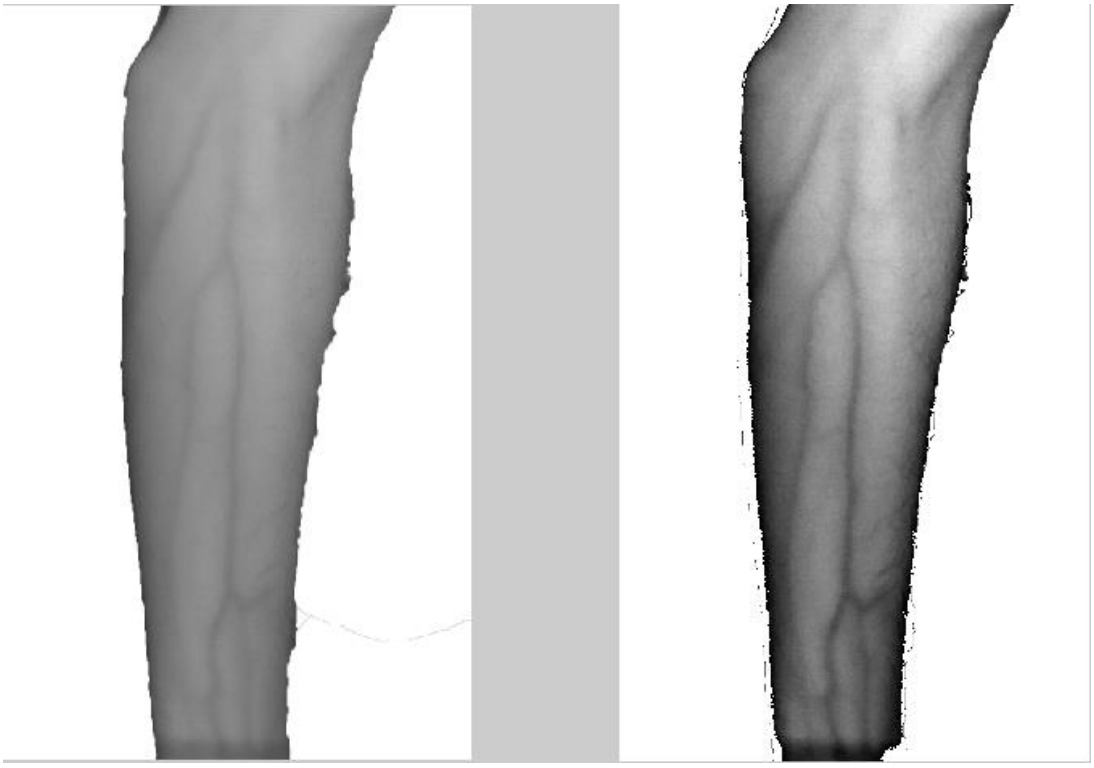


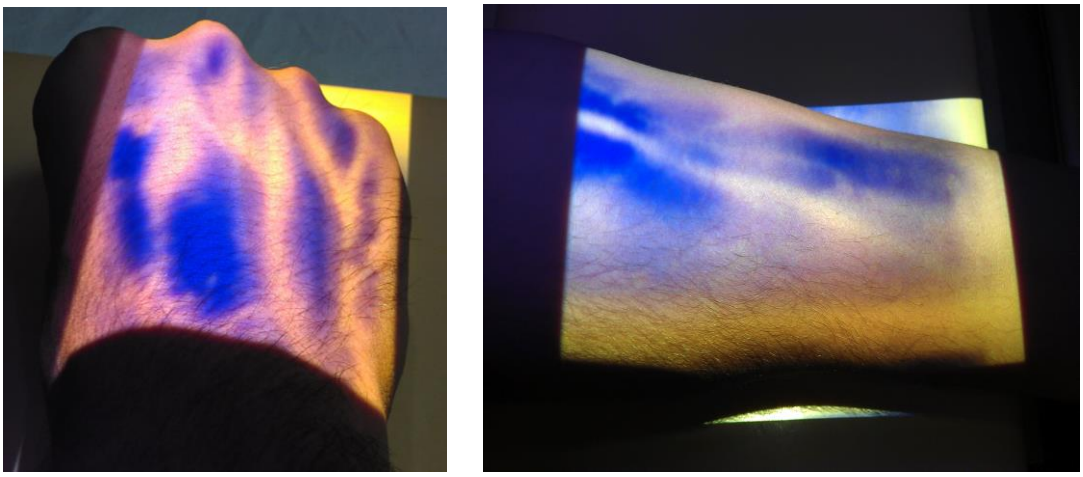
Fig 13. Results of Phase 3 (Contrast Enhancement Done - Arm)

Fig 14. Results of Phase 3 (Contrast Enhancement Done – Hand)

7.4 PHASE 4

The fourth phase was the final phase and major changes took place in this phase. The display panel was changed from LCD monitor to a pico-projector. Instead of Window/Level, Contrast Limited Adaptive Histogram Equalization was applied as a major contrast enhancement tool. Furthermore, a hardware filter was inserted between the lens and the charged coupled device (CCD). This caused the blockage of visible light and let pass through the infrared light.

The results obtained are given below:



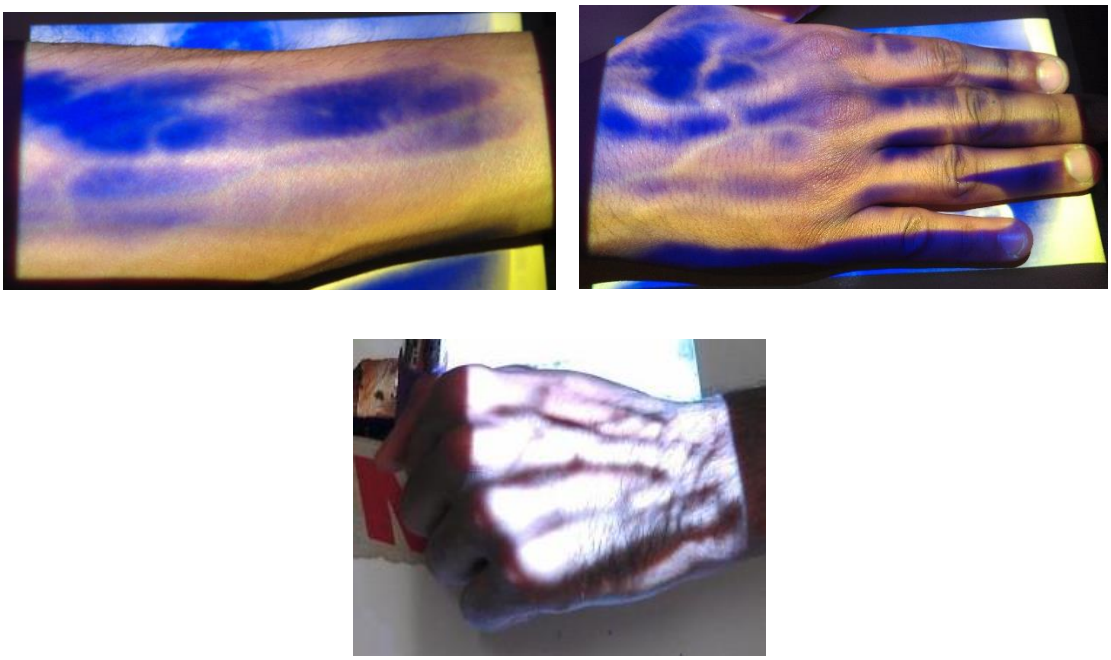


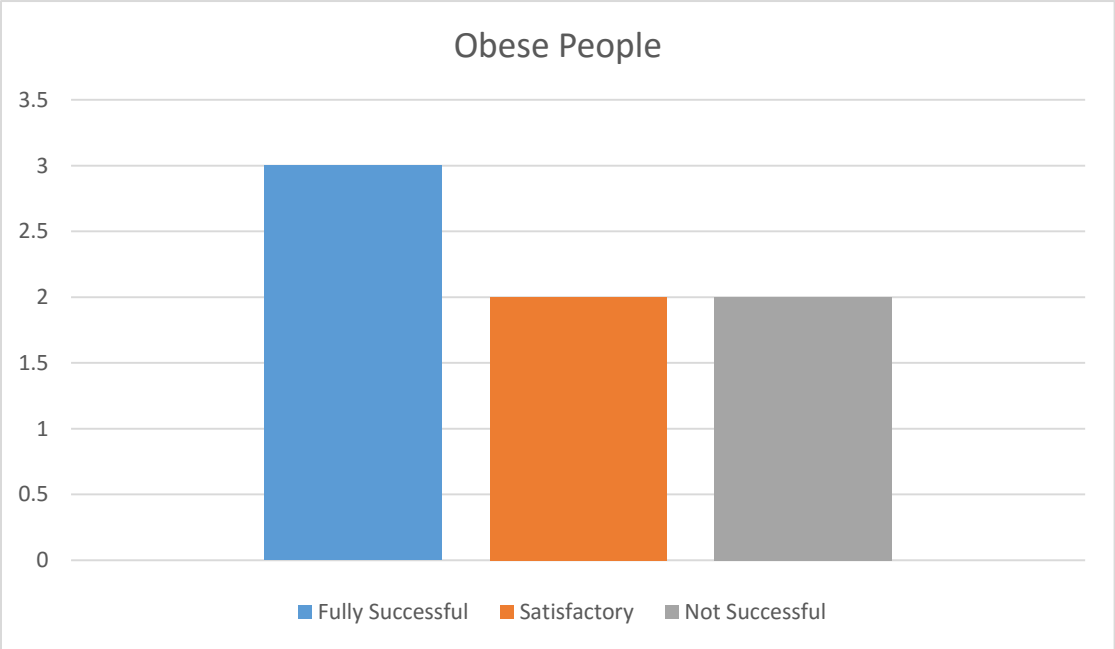
Fig 15. Results of Phase 4

As can be seen from the above figures, the results obtained in the phase 4 are improved as compared to previous stages.

In tabular form, the total success rate of the device is:

Type / Success Rate	Fully Successful in vein detection	Satisfactory Results	Not Successful
Obese	Males = 2 Females = 1	Males = 2 Females = 0	Males = 2 Females = 0
Non-Obese	Males = 13 Females = 3	Males = 2 Females = 2	Males = 0 Females = 0

Table – 2: Experimental Verification of vein detection



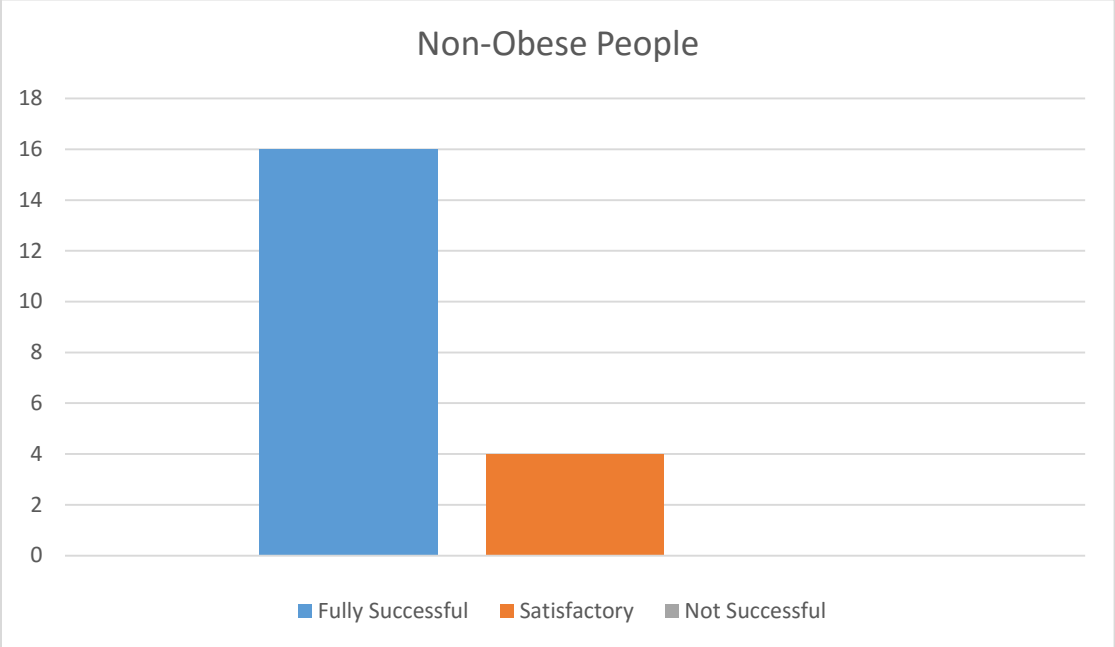


Fig 16. Results in Graphical form

It can be easily observed that the success rate of the device is quite satisfactory. From the above graphs, if both successful and satisfactory results are considered fine, then

for obese people, the success rate of the device is 71.43 % and the failure rate is 28.57%. For non-obese people, the success rate is 100% and the failure rate is 0%.

7.5 CHALLENGES FACED DURING IMPLEMENTATION

Three of the most major challenges faced during the implementation of the project are:

1. While installing OpenCV, a step is involved where compilation of OpenCV is done. During compilation of OpenCV, there appeared some compilation errors. This reason for this was not known immediately as we had a properly working internet connection.

   However, after consulting with fellow students and searching on internet, it was found that during compilation, the memory card ran out of space. We were, at that time, using an 8 GB microSD card, which is OK to use for just normal running of Raspbian OS, but if one needs to do image processing using OpenCV and Python, it is recommended that at least 16 GB memory card be used with Raspberry Pi.
2. Second major issue we faced while working with pico-projector was the continuous image-in-image phenomena. We are using pico-projector to display the obtained image of the veins back on the arm or hand, from where it is captured. Since, we were using night vision camera with no filter installed in it, the image being projected by the projector was being re-captured by the camera and the resultant image contained two images which appeared like there is an image inside another image.

   The solution for this issue was that a hardware filter in the form of a floppy disk tape, which is extremely dark in color, was installed. This helped to block the visible light being projected by the projector. Since the visible light filter was installed in the camera, the camera was unable to capture the visible light, thereby mitigating the issue.
3. The third major issue we faced while implementing the design was the calibration issues between camera and projector. As can be observed from the

hardware design, the center point of the projector's lens is different from the center point of the camera's lens. This means that whatever image is being captured by the camera will be projected back at a different point, i.e. if a camera is capturing a frame from an area having the corner points as (a,b,c,d), then the projector will project that image back to some region having corner points as (a',b',c',d'), where primed values are different from the non-primed values.

This issue was solved by shifting the center point of the image by a certain length which was calculated based on hit and trial method. This method caused the points (a,b,c,d) and (a',b',c',d') to match and hence the image to image mapping was done.

Chapter 8

CONCLUSION AND FUTURE RECOMMENDATIONS

The work being done is the first of its kind in Pakistan. No other similar product is available in Pakistan currently. This will be the first time that such a product will be commercialized. The target market of this product is hospital wards, emergencies, blood banks, blood testing labs, cosmetic surgical clinics, and around-the-corner dispensaries.

8.1 LIMITATIONS AND FUTURE RECOMMENDATIONS

As it can be observed from the results, the success rate of this device is quite less on obese people as compared to non-obese people. This is because of the fact that the infrared rays being utilized in our design is 850 nm. According to the research presented in chapter 3, the maximum absorption of infrared wavelength in blood is achieved at 750 nm. Hence, if an IR wavelength of 750 nm is used instead of 850 nm, the results are anticipated to improve further.

Another limitation is the low quality camera being utilized. Keeping in view the financial constraints, the best possible camera available was utilized in the project.

Finally, the projector being used is also of low quality as its native resolution was 848x480. This implies that the results can be further improved by utilizing a projector of a higher quality.

Hence, future recommendations for improving the design are:

- Utilization of 750 nm instead of 850 nm.
- Use a high quality camera.
- Use a better projector.

The most important finding of this project is the utilization of the infrared rays in medical field. This project has helped us determine the usefulness of IR

wavelengths and this can further other areas of research in biomedical industry. IR rays can be used in detection of blood pressure, gastro-intestinal cancer, brain tumor and others. Furthermore, IR rays might also replace other harmful rays such as X-rays and Ultraviolet rays in near future. It is possible that the imaging systems might be completely revolutionized in future because infrared rays have appeared as a potential replacement to other rays being used in imaging systems today.

This product is intended to help the healthcare professionals perform the intravenous procedures in a risk-free way, thereby not only increasing the staff productivity but also patient satisfaction by decreasing the stick count.

REFERENCES

[1] Juric, S., & Zalik, B. (2014). “An innovative approach to near-infrared spectroscopy using a standard mobile device and its clinical application in the real-time visualization of peripheral veins”. BMC medical informatics and decision making, 14(1), 1.

[2] Nundy, Koushik Kumar, and Shourjya Sanyal. "A low cost vein detection system using integrable mobile camera devices." India Conference (INDICON). 2010.

[3] AccuVein Website. URL: http://www.accuvein.com/. [accessed on 2016-03-07]

[4] VeinViewer Vision Website. https://www.christiemed.com/. [accessed on 2016-03-07]

[5] Cuper, N. J., de Graaff, J. C., Kalkman, C. J., & Verdaasdonk, R. M. (2010, February). “The VascuLuminator: effectiveness of a near-infrared vessel imaging system as a support in arterial puncture in children”. In BiOS (pp. 75550U-75550U). International Society for Optics and Photonics.

[6], [7], [8] Biomedical Engineering https://en.wikipedia.org/wiki/Biomedical_engineering [accessed on 2016-05-08]

[9] Lahiri, B. B., Bagavathiappan, S., Jayakumar, T., & Philip, J. (2012). Medical applications of infrared thermography: a review. *Infrared Physics & Technology*, *55*(4), 221-235.

[10] Wray, S., Cope, M., Delpy, D. T., Wyatt, J. S., & Reynolds, E. O. R. (1988). Characterization of the near infrared absorption spectra of cytochrome aa3 and haemoglobin for the non-invasive monitoring of cerebral oxygenation.*Biochimica et Biophysica Acta (BBA)-Bioenergetics*, *933*(1), 184-192.

[11] Mansoor, M., Sravani, S. N., Zahra Naqvi, S., Badshah, I., & Saleem, M. (2013, February). Real-time low cost infrared vein imaging system. In *Signal Processing*

Image Processing & Pattern Recognition (ICSIPR), 2013 International Conference on (pp. 117-121). IEEE.

[12] F. B. Chiao; F. Resta-Flarer; J. Lesser, J. Ng; A. Ganz; D. Pino-Luey; H. Bennett; C. Perkins Jr; B. Witek, "Vein visualization: patient characteristic factors and efficacy of a new infrared vein finder technology," *British Journal of Anaesthesia,* vol. 110, no. 6, pp. 966-971, 2013.

[13] Histogram Equalization. https://en.wikipedia.org/wiki/Histogram_equalization [accessed on 2016-05-08]

[14] Window/Level. URL: http://www.cs.ioc.ee/~khoros2/one-oper/window-level/front-page.html [accessed on 2016-05-08]

[15] Window/Level. URL: http://www.cs.ioc.ee/~khoros2/one-oper/window-level/winlevfig.gif [accessed on 2016-05-08]

[16] Install NOOBS on Raspberry Pi. URL: https://www.raspberrypi.org/help/noobs-setup/ [accessed on 2016-05-08]

[17] Install OpenCV 3 on Raspbian Jessie. URL: http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/ [accessed on 2016-05-08]

APPENDIX A

Matlab code – Window/Level

```
Orig=imread('your_image.jpg');
J=rgb2gray(Orig);
DoubJ=im2double(J);
subplot(1,3,1);
imshow(DoubJ);
Lval=0.35;                    % Specify the lower limit of the window
Hval=0.75;                    % Specify the upper limit of the window
[length, width] = size(J);
Newim = ones(length,width)*255;
for lop = 1:length
   for lop2= 1:width
      if DoubJ(lop,lop2)>=Lval && DoubJ(lop,lop2)<=Hval
         Newim(lop,lop2) = (DoubJ(lop,lop2)-Lval)/(Hval-Lval);
      end
   end
end
subplot(1,3,2);
imshow(Newim);
```

Python+OpenCV Code

To write the code, we need some sort of text editor where this code can be written. After this code has been written save it as a '.py' file which implies a python extension. To do so:

1. Open Terminal.
2. Type: leafpad filename.py
3. Type the code given below.

4. Save it using Ctrl+S and exit.

```
from picamera import PiCamera

# picamera is a library installed in connection with Raspberry Pi NoIR camera.

# PiCamera is a class of the library.

import time

# Time library to time the events.

import cv2

# OpenCV library

import numpy as np

# Numpy is a highly efficient library to convert images into multi-dimensional
# Numpy arrays. These are used to perform quick numerical analysis on the
# image and allows to use the idexing to access different region(s) of images.
camera = PiCamera()

# camera is an object of PiCamera() class.

camera.resolution = (848,480)

# Resolution of camera is set to 848x480

camera.framerate = 24

# Framerate is set to 24 fps.

rawCapture = PiRGBArray(camera, size=(848,480))

# rawCapture is an object of PiRGBArray class.

cv2.namedWindow('PrecisVeine', cv2.WINDOW_AUTOSIZE)

# Creates a window to display the frames.
```

time.sleep(0.1)

Allows the camera to warmup.

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

Capture frames from the camera.

image=frame.array

Captured frame is converted to Numpy array and stored in image

variable

b, g, r = cv2.split(image)

Splits the captured image into its respective RGB channels.

image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

Used to convert the color space of the image. The flag #'COLOR_BGR2GRAY' is used to convert RGB image into grayscale image.

clahe = cv2.createCLAHE(clipLimit = 25, tileGridSize=(4,4))

Creates an object of CLAHE and defines the clip limit and the tile size

Clip Limit is used to reduce the noise in an image. Tile grid size defines

the size of region in an image over which Histogram Equalization is to be

applied.

cl1 = clahe.apply(image)

Apply the Histogram Equalization to the capture frame to enhance the

contrast.

```
    negative = abs(255-cl1)
    # Take the negative of the captured frame
final = cv2.cvtColor(negative,cv2.COLOR_GRAY2BGR)
# Converts the grayscale image to RGB image
final = cv2.merge((b,final))
# Used to merge different multi-channel images and produce a resultant
# image that having channels equal to the sum of input images' channel.
cv2.imshow("PrecisVeine", final)
# Used to show the frame in the created window.
rawCapture.truncate(0)
# Used to destroy the frame before taking next one. If the previous frame
# is not destroyed, then the buffer containing the images is overflowed and
# no further frames can be stored, thereby causing a crash.
key = cv2.waitKey(1) & 0xFF
# Waits for a key to be pressed to escape from the loop.
if key == ord("q"):
        # Compares the pressed key to 'q'
        break
        # Breaks from the loop if the pressed key is matched with the
        # condition in the if-statement
```