

# TRAINING REPORT

## 6-Week Summer Training



DEPARTMENT OF BIOMEDICAL INSTRUMENTATION

---

CSIR-Central Scientific Instrumentation Organisation Sector-30  
Chandigarh

---

Report submitted in partial fulfillment of the requirements for the degree of  
Bachelor of Technology in the department of

Electronics & Communication Engineering  
By

Ritu Raj [1731903]



---

I.K.GUJRAL PUNJAB TECHNICAL UNIVERSITY JALANDHER-  
KAPURTHALA HIGHWAY KAPURTHALA -144603 (PB) INDIA

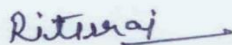
---

June-July 2019

---

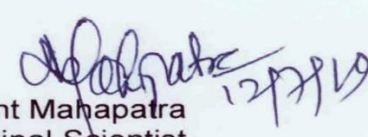
## CERTIFICATE

I hereby declare that the work being presented in this 6-Week Summer Training Report in partial fulfilment of requirements for the award of degree of Bachelor of Technology submitted in the Department of Electronics & Communication Engineering at Main Campus, I. K. Gujral Punjab Technical University, Kapurthala is an authentic record of my work under the supervision of Dr. Prasant Mahapatra. The matter presented in this report has not been submitted by us in any other University / Institute for the award of Degree / Diploma. I declare that the above statement made by the me is correct to the best of my knowledge.



Ritu Raj [1731903]

---

  
Dr. Prasant Mahapatra  
Principal Scientist  
Biomedical Instrumentation  
CSIO-CSIR, Chandigarh

---

The Viva-Voce Examination is held at Department of Electronics & Communication Engineering, Main Campus, IKGPTU, Kapurthala on .....

\_\_\_\_\_  
Signature of the EXAMINER

\_\_\_\_\_  
Signature of the HEAD,  
Department of Electronics & Communication Engineering

---

## ACKNOWLEDGEMENTS

Being Engineering student learning new concepts is most important in my life. In this project-oriented training students where are asked to realize concepts themselves under the supervision of Scientists. I would like to express my special thanks of gratitude to our Principal scientist **Dr. Prasant Mahapatra** for their able guidance and support throughout my training. I express my sincere regards to CSIO-CSIR Biomedical instrumentation department for providing the world class infrastructure and the resources for the training.

I would take this opportunity to thank to my guide, PhD scholar **Viney lohchab**, Biomedical instrumentation Department CSIO-CSIR for guiding me throughout this training. I also want to acknowledge the support received from **Mr. Lalit Maurya, Mr. Vikas Kumar, Mr. Nitin Mohan Sharma**.

I would also like to extend my gratitude to **Ms Kavita Singh** coordinator, trainee and apprenticeship cell CSIO-CSIR. Without her support this training would not have been possible.

I also appreciate encouragement received from my fellow lab mate **Mr. Raviranjan Kumar**, who constantly forced me to learn and practice concepts. We learnt many concepts together while doing the training.

This experience is beginning of learning that could occur only with support of our teachers: **Dr Avtar Singh Buttar, Dr Satvir Singh, Dr Dalveer Kaur, Dr Rakesh Goyal and Dr Amit Gupta** Electronics and Communication Department, I. K. Gujral Punjab Technical University. I acknowledge their support and teachings from the depths of my heart.

I express my great regards and gratitude for my parents who gave me this opportunity to study at IK Gujral Punjab Technical University, Kapurthala. It is possible due to their sacrifices and constant blessings that kept me motivated and committed.

Finally, I must thank GOD for giving me the environment to study, people to help, opportunities to encash and potential to succeed.

Ritu Raj [ 1731903 ]

---

## ABSTRACT

Practical exposure given to trainees in such summer training during play important role in getting jobs and having successful professional career. Trainees are taught simple but important fundamental concepts of Digital Image Processing.

First of all trainees were asked to go through the theories of Digital Image Processing and learn the required concepts to do image processing. Trainees learnt various operations like import, export, filtration, enhancement of image. Trainees learnt to do the segmentation and analysis of image.

Trainees used MATLAB for doing the image processing. It was important to understand all the required commands of Image processing toolbox to do the image processing and to complete the project work.

Segmentation of hand image was done and Morphological operations and many other algorithms were applied to get the length of each fingers of the hand.

Real life examples of Digital Image Processing were shared throughout the training period to trainees to understand the importance of Digital Image Processing in the present scenario.

In a nutshell, this was the time when students enjoyed while learning and applying the gained knowledge, and get their work done.

Ritu Raj [ 1731903 ]

---

# CONTENTS

.....	1
ACKNOWLEDGEMENTS .....	3
ABSTRACT .....	4
CHAPTER – 1 .....	7
INTRODUCTION .....	7
1.1    About CSIR .....	7
1.2    About CSIR-CSIO .....	8
1.3    History.....	8
1.4    Mission .....	9
1.5    Mandate .....	9
1.6    Achievements .....	9
CHAPTER-2 .....	11
IMAGE PROCESSING .....	11
2.1    Introduction .....	11
2.2    In CSIO-CSIR .....	11
2.3    Digital Image Processing .....	11
2.4    Different Image Standard.....	12
2.5    Types of Digital Images .....	13
2.6    Fundamental digital processing steps: .....	15
2.7    Image File Size.....	15
2.8    Image Conversion.....	15
2.9    Spatial Resolution .....	15
2.10    Key Stages in Digital Image Processing .....	17
2.11    Applications .....	20
CHAPTER-3 .....	21
MATLAB.....	21
3.1    Introduction .....	21
3.2    Basic image processing commands in MATLAB .....	21
3.3    Image Processing Toolbox.....	30
CHAPTER-4 .....	32
PROJECT WORK.....	32
4.1    Introduction .....	32

4.2	Image Acquisition .....	32
4.3	Commands Used .....	33
4.4	Steps.....	33
4.5	Code .....	36
CHAPTER- 5.....		39
FINAL DISCUSSIONS .....		39

---

# CHAPTER – 1

## INTRODUCTION

### 1.1 About CSIR

Council of Scientific & Industrial Research (CSIR), India was constituted in 1942 as an autonomous body under the provision of the Registration of Societies Act XXI of 1860. After independence, the need for bettering the living standards of the common man by promoting industry and for helping the industry to solve its problems through stimulus of scientific research was greatly stressed. The Council, through its constituent laboratories, has helped the country in increasing the economic growth and industrialization.

The Council has also helped the creation of new schools of research and in enlarging facilities for research by means of grants, training of research personnel, etc. The main functions of the Council are :

- Promotion, guidance and coordination of scientific and industrial research in India including other institutions and financing the specific research activities.
- Scientific study of problems affecting industries and trade.
- Award of Research Fellowships.
- Utilization of the results of researches conducted under the Council towards the development of industries in India.
- The establishment, maintenance and management of laboratories, workshops and organizations to further scientific and industrial research.
- The collection and dissemination of information in regard not only to research but also to industrial matters generally.
- Publication of scientific papers.
- Other activities to promote generally the objects of resolution.

Council of Scientific & Industrial Research (CSIR), India is perhaps among the world's largest publicly funded R&D organisation. Its chain of 38 world class R&D establishments with their 80 field stations spread across India are manned by highly qualified scientists and engineers, besides auxiliary and other staff. Its range of activities cover practically the entire spectrum of industrial R&D ranging from aerospace to mining to microelectronics to metallurgy and so on. CSIR is truly a global R&D resource as its patrons and partners hail from over 50 countries.

## 1.2 About CSIR-CSIO

Central Scientific Instruments Organisation (CSIO), a constituent unit of Council of Scientific & Industrial Research (CSIR), is a premier national laboratory dedicated to research, design and development of scientific and industrial instruments. It is a multi-disciplinary and multi-dimensional apex industrial research & development organisation in the country to stimulate growth of Instrument Industry in India covering wide range and applications.

CSIO is a multi-disciplinary organization having well equipped laboratories manned by highly qualified and well-trained staff with infrastructural facilities in the areas of Agrionics; Medical Instrumentation and Prosthetic Devices; Optics and Cockpit based Instrumentation; Fiber/Laser Optics based Sensors & Instrumentation; Analytical Instrumentation; Advanced Materials based Transducers etc. Large number of instruments ranging from simple to highly sophisticated ones, have been designed and developed by the Institute and their know-hows have been passed on to the industry for commercial exploitation. Having contributed substantially towards the growth of the scientific instruments industry in the country, CSIO enjoys high degree of credibility among the users of the instruments as well as the instrument industry.

## 1.3 History

Established in October 1959, CSIO was chartered to stimulate the growth of indigenous instrument industry in the country through development of contemporary technologies and other scientific & technological assistance.

Initially located at New Delhi, CSIO moved to Chandigarh (the City Beautiful) in 1962. CSIO Campus (spread over an area of approximately 120 acres) comprises of Office Buildings, R&D Laboratories, Indo-Swiss Training Centre and a Housing Complex. An austere four-story building and the accompanying workshops were inaugurated in December 1967. Another four-story block was added in 1976 for housing R&D Divisions, Library, etc. During mid-eighties, the laboratory buildings and infrastructural facilities were modernized in order to gear the Institute towards taking up development projects in challenging and emerging areas of technology.

A separate Administrative Block was inaugurated in September 1994.

With a view to meeting the growing demand of well-trained instrument technologists, Indo-Swiss Training Centre (ISTC) was started in December 1963 with the co-operation of Swiss Foundation for Technical Assistance, Zurich, Switzerland.



## 1.4 Mission

To carry out research in niche areas of measurement science and innovative instrumentation technology for strategic and societal application.

To improve quality service and human resource development in advance instrumentation.

To emerge as a global player in the field of instrumentation sciences

## 1.5 Mandate

- Research, design & development of scientific & industrial instruments, components and systems

- Service, maintenance, testing & calibration of instruments/components

- HRD in the area of instrumentation

- Technical assistance to industry

## 1.6 Achievements

### TRANSFERS OF TECHNOLOGIES:

1. Absorption System of Atomic Absorption Spectrophotometer
2. Anesthesia Ventilator
3. Automatic Counterfeit Currency Detector
4. Clinical Chemistry Analyser
5. Digital Aflatoxin Meter
6. Digital Cereal/Grain Analyser
7. Digital Titrator
8. Dispergraph for measuring And Analysing Carbon Black Dispersion in Rubber
9. Driver Reflexes Testing System
10. Drug Infusion Pump and Controller
11. Electronic Stethoscope
12. Electronic three-phase Real Energy Meter
13. Energy Management System
14. Energy Management System (EMS) based on MODBUS Protocol
15. Explosive Detector
16. Formaldehyde Measuring System
17. Glow Discharge Lamp- Atomic Emission Spectrophotometer
18. Gold Analyser
19. Head Up Display (HUD) for LCA

20. Hydraulic Knee Joint
21. IR Based Snow Surface Temperature Probe alongwith Installation Tower
22. Improve Lathe Tool Post
23. Instant Soil pH Meter
24. Iodine Value Meter
25. Low Cost Oxygen Monitor
26. Low Voltage Electronic Precipitator
27. Micro Hardness Tester
28. Moisture Computer
29. Multifiber Intrusion Detection System
30. NEO-NATAL OXYGEN MONITOR
31. Nephelometer
32. Non-Linear Junction Detector
33. Oil Spectrometer
34. On-line Analyser for Energy Monitoring and Conservation
35. On-line Energy Monitoring & Control System
36. Personal Dust Monitor
37. Portable Pulse Oximeter
38. Portable Stack Opacity Monitor
39. Pulse Oximeter
40. Resuscitation BAG
41. Semi -Automatic Pick & Place Machine for SMDs
42. Servo Control Baby Care Incubator
43. Single Puncture LAPAROSCOPE
44. Sodium-Potassium Analyser
45. Surgical Microscope

---

# CHAPTER-2

## IMAGE PROCESSING

### 2.1 Introduction

Biomedical instrumentation involves devices designed and connected together in scientifically appropriate manner to sense signals and process them for human display and further processing for control, therapy and other purposes. Some instruments are digital thermometer, electro-cardiograph and so on.

### 2.2 In CSIO-CSIR

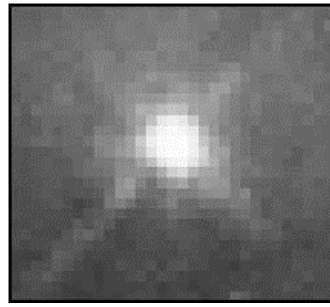
Development of technologies for Biomedical instruments is one of the focus area of research in CSIO. The Biomedical group is working towards research, design & development of futuristic medical devices for diagnostic, therapeutic, prosthetic & assistive devices for rehabilitation and orthopedic implants. Besides R&D, on-site and in-house test/calibration services for medical instruments are also being provided to the valued customers through ISO 9001-2000 certified national calibration facility established at Delhi Centre.

At present, R&D activities of the group are focused towards providing technological Solutions for Solid Tumour Targeting using Peptides & Plasmonic Photo-Thermal Technique, Exoskeleton devices (ExoD), Intelligent Patient Movement Aids, Automation of Surgical Processes, Design & Development of State-of-the-art patient specific Orthopedic Implants, Postural Stability Assessment System, Biomechatronic Rehabilitative Solutions and Virtual Intelligence based Rehabilitation System.

### 2.3 Digital Image Processing

#### DIGITAL IMAGE

An image may be defined as a two-dimensional function,  $f(x, y)$ , where  $x$  and  $y$  are spatial (plane) coordinates, and the amplitude of  $f$  at any pair of coordinates  $(x, y)$  is called the intensity or gray level of the image at that point. When  $x$ ,  $y$ , and the amplitude values of  $f$  are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital images by means of a digital computer.



$g(x, y)$



$g(i, j)$



$f(i, j)$  Digital Image

$f(i_0, j_0)$  : Picture Element, Image Element, Pel, Pixel

A digital image is a representation of a two dimensional image as a finite set of digital values, called picture elements or 'pixels or image elements or pel'

A digitised image is represented as:

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \dots & f(1, N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1, 0) & f(M-1, 1) & \dots & f(M-1, N-1) \end{bmatrix}$$

Each element in the array is referred to as a pixel or an image element.

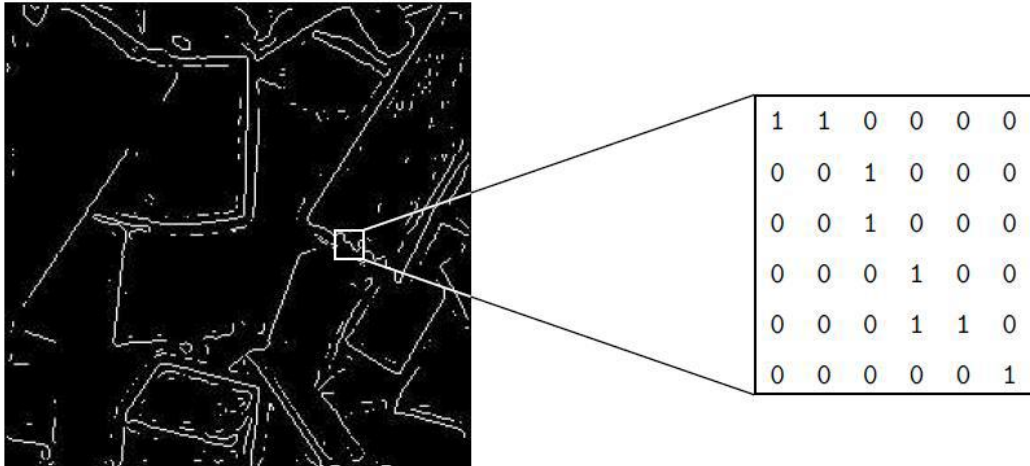
## 2.4 Different Image Standard

- JPEG: Joint Photographic Experts Group
- GIF: Graphics Interchange Format
- BMP: Bit Mapped
- TGA :Truevision Graphics Adaptor
- PNG: Portable Network Graphics
- TIFF :Tagged Interchange/Image File Format
- EXIF: Exchangeable Image File Format
- PCX: Personal Computer eXchange

## 2.5 Types of Digital Images

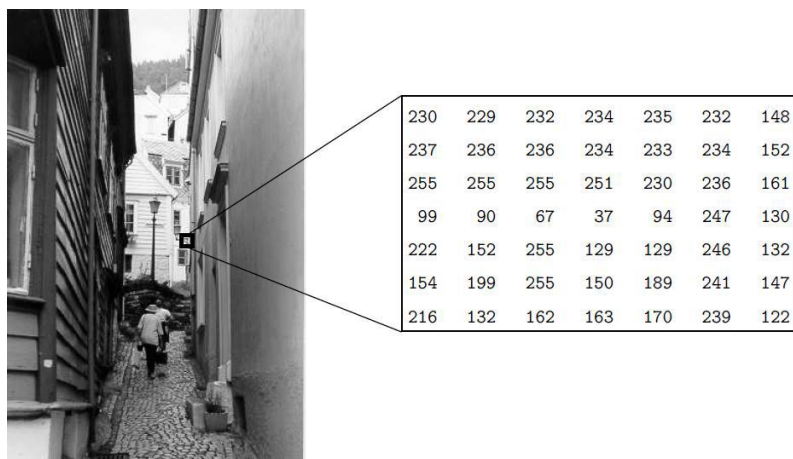
### 2.5.1 A BINARY IMAGE

- A binary image is shown in figure below. In this image, we have only the two colours: White for the edges, and black for the background.



### 2.5.2 A GRAY SCALE IMAGE

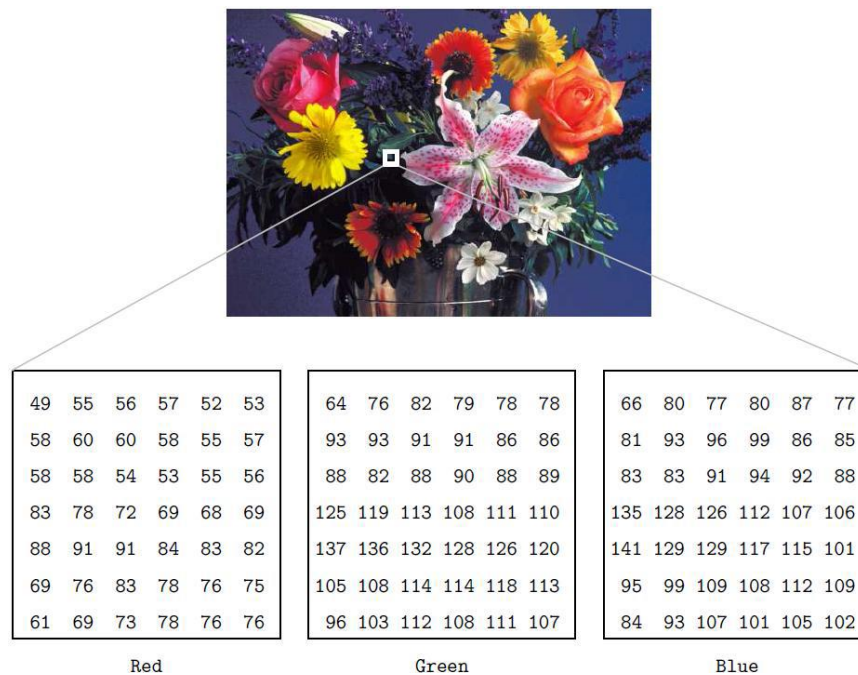
- Each pixel is a shade of gray, normally from 0 (BLACK) to 255 (WHITE). This range means that each pixel can be represented by eight bits, or exactly one byte. This is a very natural range for image file handling. Other grayscale ranges are used, but generally they are a power of 2. Such images arise in medicine (Xrays), images of printed works, and indeed 256 different gray levels is sufficient for the recognition of most natural objects.



### 2.5.3 TRUE COLOUR OR RGB IMAGE

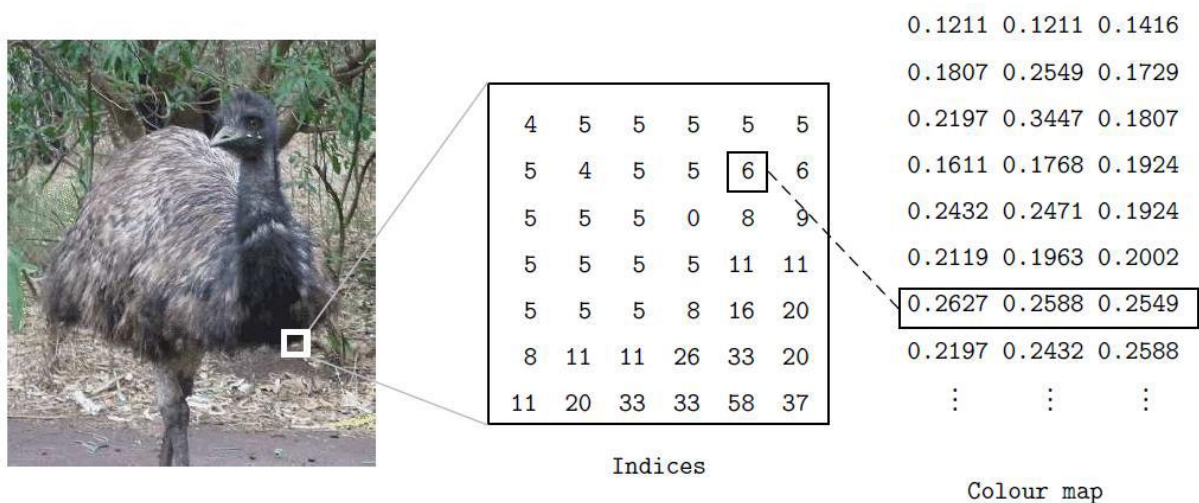
- Each pixel has particular colour, that colour is described by the amount of RED, GREEN and BLUE in it.

- If each these have a components 0-255, this gives a total of  $256^3 = 16777216$  different possible colours in the image. This is enough for any image.
- Total number of bits required for each pixel is 24 bits colour image.



## 2.5.4 INDEXED IMAGE

- Most colour image has only a small subset of colour image of more than sixteen million possible colours.
- For convenience to store and file handling the image has associated a colour map or colour palette which is simply a all colours used in the image.
- Each pixel has a value which does not gives its colour but index to a colour map.
- It is convenient if a colour image has 256 colours or less then only 1 byte is required to store each pixel.



## 2.6 Fundamental digital processing steps:

- Image Acquisition
- Image Enhancement
- Image Restoration
- Image Analysis
- Image Reconstruction
- Image Compression

## 2.7 Image File Size

- A BINARY IMAGE: Let us consider a 512 x 512 binary image. The number of bits used in this image is:  $512 \times 512 \times 1 = 262,144$  bits = 32768 bytes
- A GRAY SCALE IMAGE: A gray scale image of same size requires:  
 $512 \times 512 \times 8 = 2097152$  bits = 262144 bytes
- A COLOUR OR RGB IMAGE: Let us consider a 512 x 512 colour image. The number of bits used in this image is:  $512 \times 512 \times 3 \times 8 = 6291456$  bits = 786432 bytes

## 2.8 Image Conversion

- Image can be converted from one type to another. We can understand with the below tables.

Function	Use	Format
ind2gray	Indexed to Greyscale	<code>y=ind2gray(x,map);</code>
gray2ind	Greyscale to indexed	<code>[y,map]=gray2ind(x);</code>
rgb2gray	RGB to greyscale	<code>y=rgb2gray(x);</code>
gray2rgb	Greyscale to RGB	<code>y=gray2rgb(x);</code>
rgb2ind	RGB to indexed	<code>[y,map]=rgb2ind;</code>
ind2rgb	Indexed to RGB	<code>y=ind2rgb(x,map);</code>

## 2.9 Spatial Resolution

- Spatial resolution is the density of pixels over the image.
- The greater the spatial resolution, the more pixels are used to display the image.
- We can experiment with spatial resolution with MATLAB
- imresize function. Suppose we have an 256 X 256 8-bit Grayscale image saved to matrix X Then command `>>imresize(x,1/2);` will halve the size of image  
`>>x2=imresize(imresize(x,1/2),2);` can recover original.



Command	Effective resolution
<code>imresize(imresize(x,1/4),4);</code>	$64 \times 64$
<code>imresize(imresize(x,1/8),8);</code>	$32 \times 32$
<code>imresize(imresize(x,1/16),16);</code>	$16 \times 16$
<code>imresize(imresize(x,1/32),32);</code>	$8 \times 8$



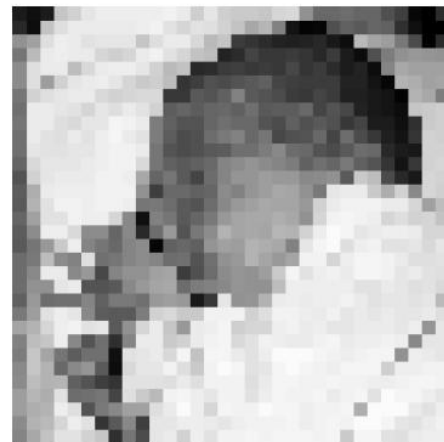
(a) The original image



(b) at  $128 \times 128$  resolution



(a) At  $64 \times 64$  resolution



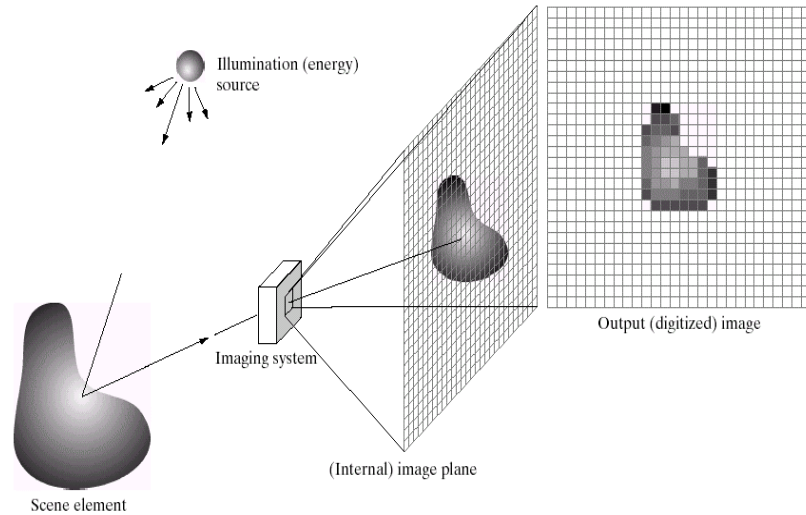
(b) At  $32 \times 32$  resolution



## 2.10 Key Stages in Digital Image Processing

### I. Image Acquisition

Image acquired from sensors/cameras/CCD Cameras etc.. In digital form.

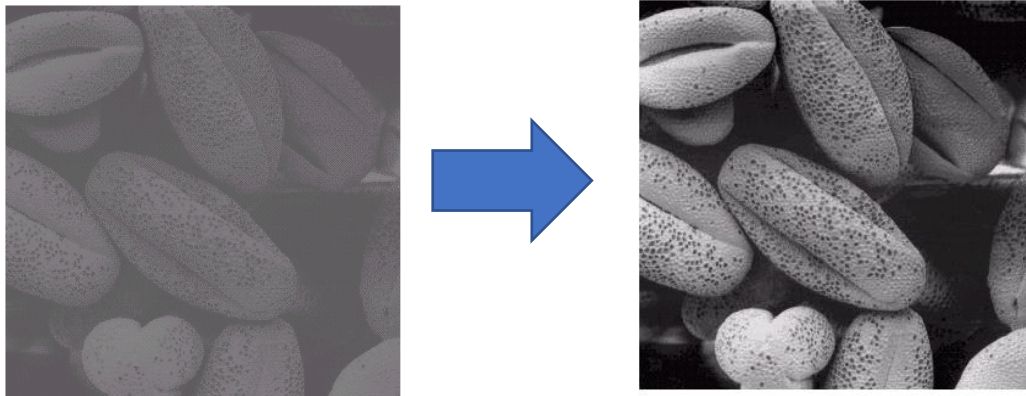


### II. Image Enhancement:

This refers to processing an image so that the result is more suitable for a particular application.

Example include:

- a) Sharpening or de-blurring an out of focus image
- b) Highlighting the edges
- c) Improving Image contrast or Brightening and Image
- d) Removing Noise.

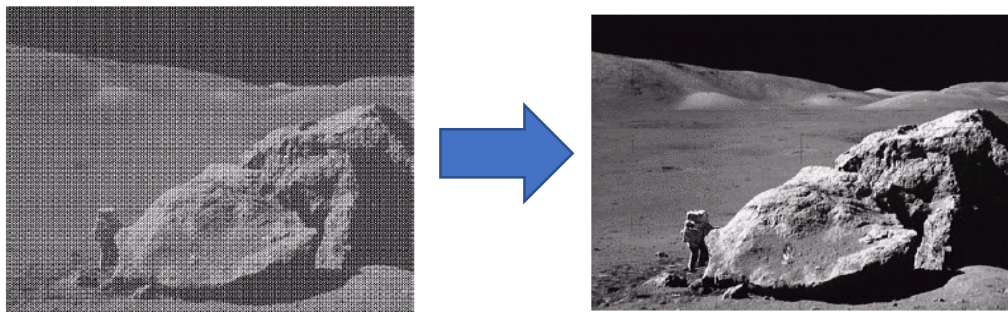


### III. Image Restoration

This may be considered as reversing the damage done to an image by a known cause,

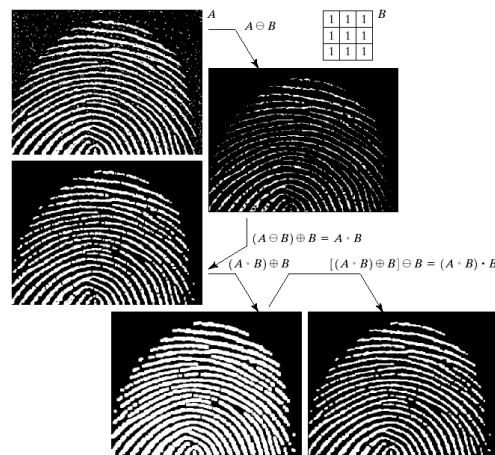
for example:

- a) Removing of blur caused by linear motion
- b) Removal of optical distortions
- c) Removing periodic interference.



### IV. Morphological Processing

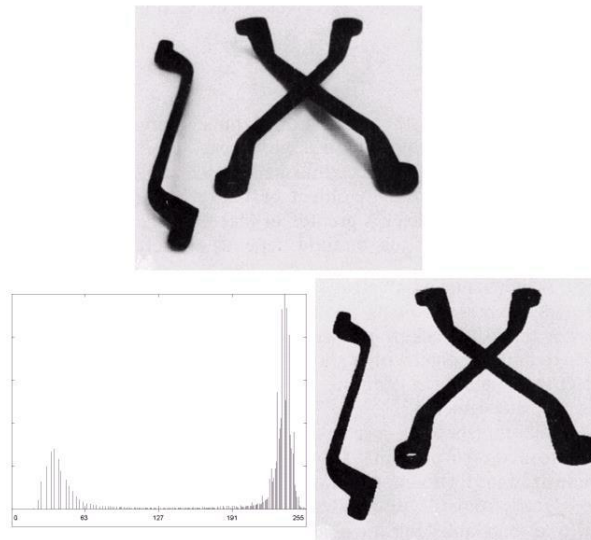
Extraction of image feature that may be useful to identify shapes or structures or boundaries or skeletons.



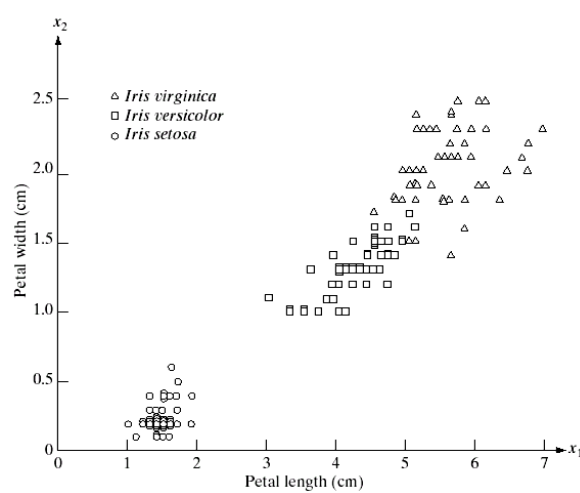
### V. Segmentation

This involves subdividing an image into constituent parts, or isolating certain aspects of an image:

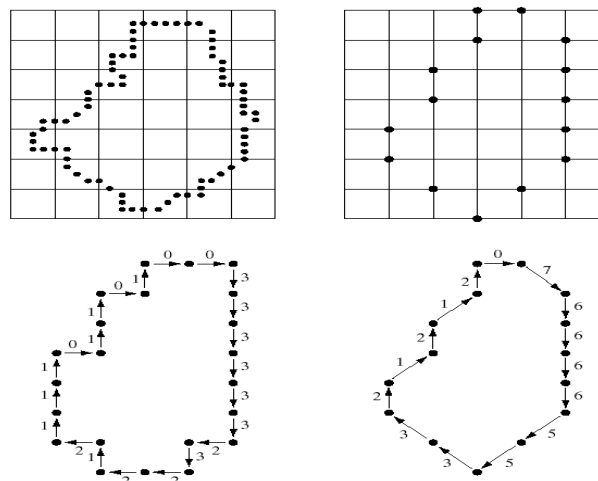
- a) Finding lines, circles, or particular shapes in an image,
- b) In an aerial photograph, identifying cars, trees, buildings, or roads.



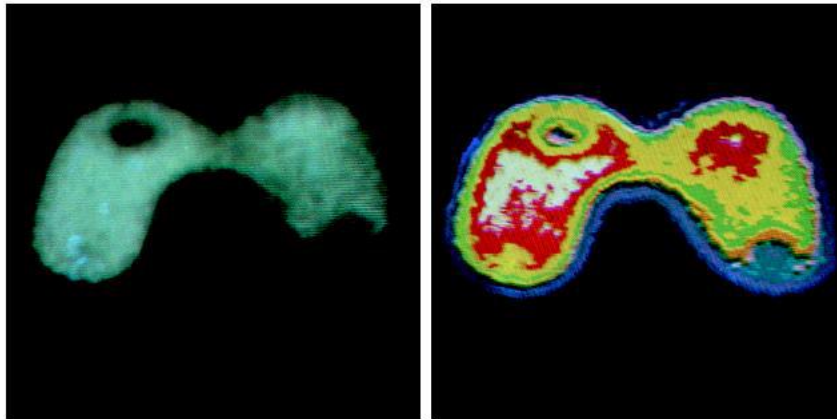
## VI. Object Recognition



## VII. Representation & Description



## VIII. Colour Image Processing



### 2.11 Applications

Present era is digital era, so digital processing of image is widely used in many applications, some of them are listed.

- Computer Vision
- Face Detection
- Human Computer Interaction
- Medical image processing (Gamma Ray Imaging, X-Ray Imaging)
- Analysis Space photograph
- Object Tracking
- Intelligent Transport system

---

## CHAPTER-3

### MATLAB

#### 3.1 Introduction

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

In MATLAB, the IPT( Image processing toolbox ) is a collection of functions that extends the capability of the MATLAB numeric computing environment. It provides a comprehensive set of reference-standard algorithms and workflow applications for image processing, analysis, visualisation and algorithm development. It can be used to perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration and 3D image processing operations. Many of the IPT functions support C/C++ code generation for desktop prototyping and embedded vision system deployment

#### 3.2 Basic image processing commands in MATLAB

##### 3.2.1 Reading images

Images are read in MATLAB environment using the function 'imread.' Syntax of imread is:

```
imread('filename');
```

where 'filename' is a string having the complete name of the image, including its extension.

For example,

```
>>F=imread(Penguins_grey.jpg);  
>>G = imread(Penguins_RGB.jpg);
```

Please note that when no path information is included in 'filename,' 'imread' reads the file from the current directory. When an image from another directory has to be read, the path of the image has to be specified.

Semicolon (;) at the end of a statement is used to suppress the output. If it is not included, MATLAB displays on the screen the result of the operation specified in that line.

'>>' indicates the beginning of a command line as it appears in the MATLAB command window.



*Fig. 1: Grayscale image of penguins*



*Fig. 2: RGB image of penguins*

Figs 1 and 2 show grayscale and RGB images of penguins, respectively. These images can be downloaded from the EFY website and stored in the current working directory.

imread, imshow and imwrite functions in MATLAB are used to read images in MATLAB environment, display them on MATLAB desktop and write them to the current directory, respectively

In case of grayscale images, the resultant matrix of the imread statement comprises 256×256 or 65,536 elements. The first statement takes grey values of all the pixels in the grayscale image and puts them into a matrix F(256×256 elements), which is now a MATLAB variable on which various matrix operations can be performed.

In case of RGB images, pixel values now consist of a list of three values, giving red, green and blue components of the colour of the given pixel. Matrix 'G' is a three-dimensional matrix 256×256×3. If there is no semicolon, the result of the command is displayed on the screen.

In nutshell, the imread function reads pixel values from an image file and returns a matrix of all pixel values.

To get the size of a 2D image, you can write the command:

[M,N] = size(f)

This syntax returns the number of rows (M) and columns (N) in the image.

You can find additional information about the array using 'whos' command. 'whos f' gives name, size, bytes, class and attributes of the array 'f'. As an example, 'whos F' in our case gives:

Name	Size	Bytes	Class Attributes
F	768×1024×3	2359296	uint8

'whos G' gives:

Name	Size	Bytes	Class Attributes
G	768×1024	786432	uint8

### 3.2.2 Image display

Images are displayed on the MATLAB desktop using the function imshow, which has the basic syntax:

imshow(f)

where 'f' is an image array of data type 'uint8' or double. Data type 'uint8' restricts the values of integers between 0 and 255.

It should be remembered that for a matrix of type 'double,' the imshow function expects the values to be between 0 and 1, where 0 is displayed as black and 1 as white. Any value between 0 and 1 is displayed as grayscale. Any value greater than 1 is displayed as white, and a value less than zero is displayed as black. To get the values within this range, a division factor can be used. The larger the division factor, the darker will be the image.

As an example, if you give the command imshow(G), the image shown on the desktop is given in Fig. 3. 'imshow(F)' displays the image shown in Fig. 4. 'imshow(f, [low high])' displays as black all values less than or equal to 'low' and as white all



values equal to or greater than 'high.' The values in between are displayed as intermediate intensity values.



*Fig. 3: Image obtained with command `imshow(G)`*



*Fig. 4: Image obtained with `imshow(F)` command*

'`imshow(f, [ ])`' sets variable low to the minimum value of array 'f' and high to its maximum value. This helps in improving the contrast of images having a low dynamic range.



VARIOUS IMAGE TOOLS AND THEIR CAPABILITIES	
Image tools	Capabilities
Pixel information	Displays information about the pixel under the mouse pointer
Pixel region	Superimposes pixel values on a zoomed-in pixel view
Measure distance	Measures the distance between two pixels
Image information	Displays information about images and image files
Adjust contrast	Adjusts the contrast of the displayed image
Crop image	Defines a crop region and crops the image
Display range	Shows the display range of the image data
Overview	Shows the currently visible image
Choose colormap	Shows the image under different colour settings

The Image tool in the image processing toolbox provides a more interactive environment for viewing and navigating within images, displaying detailed information about pixel values, measuring distances and other useful operations. To start the image tool, use the `imtool` function. The following statements read the image `Penguins_grey.jpg` saved on the desktop and then display it using 'imtool':

```
>>B = imread (Penguins_grey.jpg);
>>imtool(B)
```

Fig. 5 shows the window that appears when using the image tool. The status text at the bottom of the main window shows the column/row location and the value of the pixel lying under the mouse cursor.



*Fig. 5: The window that appears when using the image tool*

The Measure Distance tool under Tools tab is used to show the distance between the two selected points. Fig. 6 shows the use of this tool to measure distances between the beaks of different penguins.

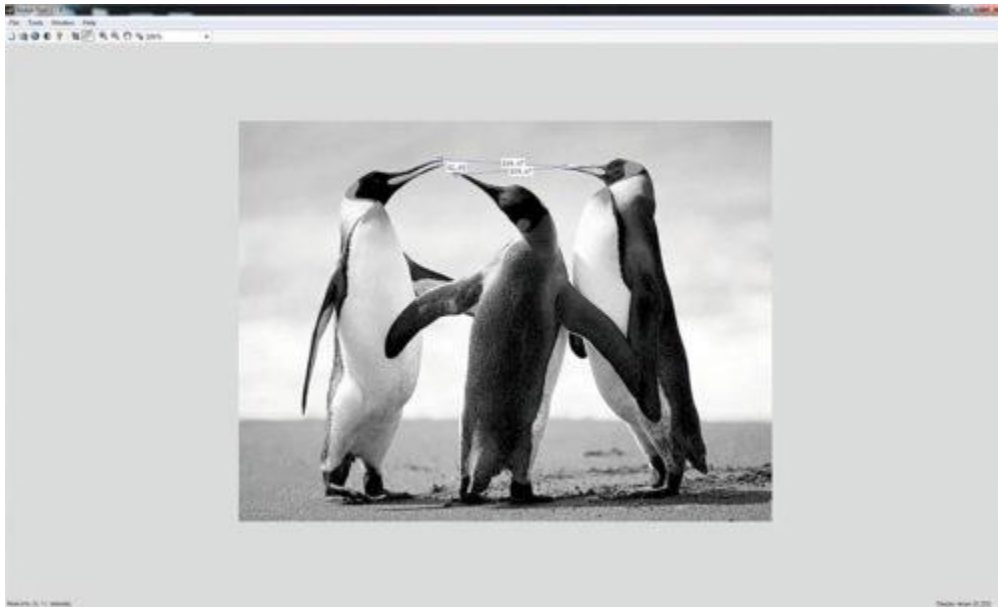


Fig. 6: The Measure Distance tool under Tools tab is used to measure distances between the beaks of different penguins

The Overview tool under Tools tab shows the entire image in a thumbnail view. The Pixel Region tool shows individual pixels from the small square region on the upper right tip, zoomed large enough to see the actual pixel values. Fig. 7 shows the snapshot.



Fig. 7: The Overview tool shows the entire image in a thumbnail view. The Pixel Region tool shows individual pixels from the small square region on the upper right tip, zoomed large enough to see the actual pixel values

Multiple images can be displayed within one figure using the subplot function. This function has three parameters within the brackets, where the first two parameters specify the number of rows and columns to divide the figure. The third parameter specifies which subdivision to use. For example, subplot (3,2,3) tells MATLAB to

divide the figure into three rows and two columns and set the third cell as active. To display the images Penguins\_RGB.jpg and Penguins\_grey.jpg together in a single figure, you need to give the following commands:

```
>> A=imread('Penguins_grey.jpg');  
>> B=imread('Penguins_RGB.jpg');  
    >>figure  
    >>subplot (1,2,1), imshow (A)  
    >>subplot (1,2,2) , imshow(B)
```

The figure that appears on the screen after the third command is shown in Fig. 8.



*Fig. 8: Grayscale and RGB images shown together*

### 3.2.3 Writing images

Images are written to the current directory using the `imwrite` function, which has the following syntax:

```
Imwrite (f, 'filename');
```

This command writes image data 'f' to the file specified by 'filename' in your current folder. The `imwrite` function supports most of the popular graphic file formats including GIF, HDF, JPEG or JPG, PBM, BMP, PGM, PNG, PNM, PPM and TIFF and so on.

The following example writes a 100×100 array of grayscale values to a PNG file named `random.png` in the current folder:

```
>> F = rand (100);  
>>imwrite (F, 'random.png')
```

When you open the folder in which you are working, an image file named 'random.png' is created.

For JPEG images, the `imwrite` syntax applicable is:

```
Imwrite (f, 'filename.jpg', 'quality', q)
```

where 'q' is an integer between 0 and 100.

It can be used to reduce the image size. The quality parameter is used as a trade-off between the resulting image's subjective quality and file size.

You can now read the image Penguins\_grey.jpg saved in the current working folder and save it in JPG format with three different quality parameters: 75 (default), 10 (poor quality and small size) and 90 (high quality and large size):

```
F = imread('Penguins_grey.jpg');  
imwrite(F,'Penguins_grey_75.  
jpg','quality',75);  
imwrite(F,'Penguins_grey_10.  
jpg','quality',10);  
imwrite(F,'Penguins_grey_90.  
jpg','quality',90);
```

Figs 9 through 11 show images for quality factors of 75, 10 and 90, respectively. You will note that that a low-quality image has a smaller size than a higher-quality image.



*Fig. 9: Image for quality factor of 75*

The imwrite syntax applicable to TIFF images has the format:

```
imwrite (g, 'filename.tif', 'compression',  
        'parameter', ..... 'resolution',  
        [colresrowres])
```

### 3.2.4 Image information

File details including file name, data, size, format, height and width can be obtained using the command:

`Imfinfo 'filename'`

As an example, the command `imfinfo ('Penguins_grey.jpg')` gives the following information:

```
Ans=  
Filename: [1x50 char]  
FileModDate: '02-Aug-2016 09:24:12'  
FileSize: 100978  
Format: 'jpg'  
FormatVersion: "  
Width: 1024  
Height: 768  
BitDepth: 8  
ColorType: 'grayscale'  
FormatSignature: "  
NumberOfSamples: 1  
CodingMethod: 'Huffman'  
CodingProcess: 'Sequential'  
Comment: {}
```

Information fields displayed by 'imfinfo' can be captured into a structural variable that can be used for subsequent computation.



Fig. 10: Image for quality factor of 10



Fig. 11: Image for quality factor of 90

The following example stores all the information into variable 'K':

```
K = imfinfo('Penguins_grey.jpg')
```

The information generated by 'imfinfo' is appended to the structure variable by means of fields, separated from 'K' by a dot. As an example, the image height and width are stored in structure fields K. height and K. weight.

### 3.3 Image Processing Toolbox

Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing.

Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large datasets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs).

You can accelerate your algorithms by running them on multicore processors and GPUs. Many toolbox functions support C/C++ code generation for desktop prototyping and embedded vision system deployment.

#### [Getting Started](#)

Learn the basics of Image Processing Toolbox.

#### [Import, Export, and Conversion](#)

Image data import and export, conversion of image types and classes.

### Display and Exploration

Interactive tools for image display and exploration.

### Geometric Transformation and Image Registration

Scale, rotate, perform other N-D transformations, and align images using intensity correlation, feature matching, or control point mapping.

### Image Filtering and Enhancement

Contrast adjustment, morphological filtering, deblurring, ROI-based processing.

### Image Segmentation and Analysis

Region analysis, texture analysis, pixel and image statistics.

### Deep Learning for Image Processing

Perform image processing tasks, such as removing image noise and creating high-resolution images from low-resolutions images, using convolutional neural networks (requires Neural Network Toolbox™).

### 3-D Volumetric Image Processing

Filter, segment, and perform other image processing operations on 3-D volumetric data.

### Code Generation

Generate C code and MEX functions for toolbox functions.

### GPU Computing

Run image processing code on a graphics processing unit (GPU).



---

## CHAPTER-4

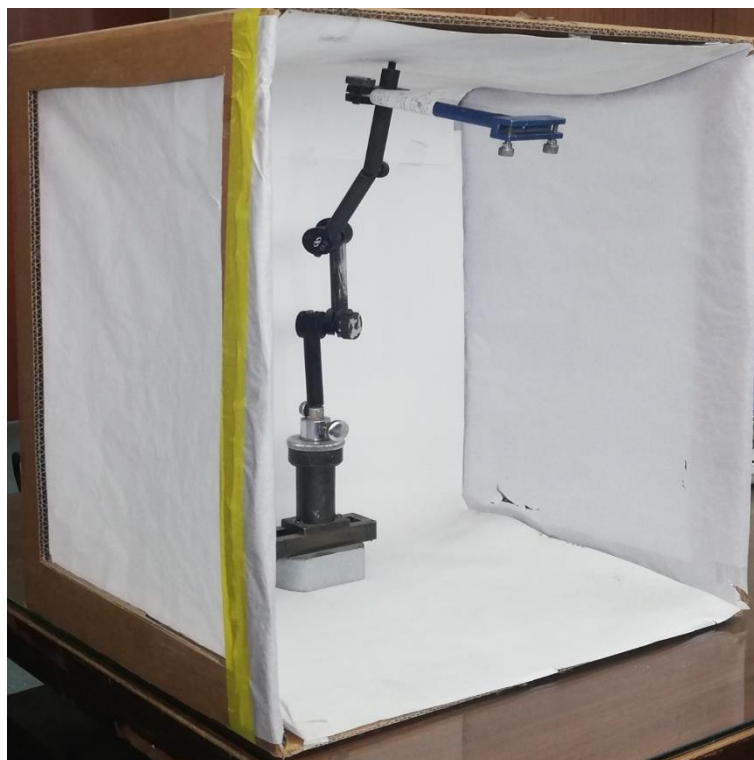
### PROJECT WORK

#### 4.1 Introduction

This Project is based on Image processing and the software used for that is MATLAB. By using IPT( Image processing toolbox ) processing of hand image is done to get the length of all the fingers. A proper discussion of all the processes, commands used, code and the results are done in this chapter.

#### 4.2 Image Acquisition

A perfect setup is required for image acquisition as image should be shadow free for image segmentation and morphological operations.





## 4.3 Commands Used

imread : Read image from graphics file  
imshow : Display image  
imfinfo : Information about graphics file  
rgb2gray : Convert RGB image or colormap to grayscale  
graythresh : Global image threshold using Otsu's method  
im2bw : Convert image to binary image, based on threshold  
bwareafilt : Extract objects from binary image by size  
imcomplement : Complement image  
imfill : Fill image regions and holes  
strel : Morphological structuring element  
imopen : Morphologically open image  
bwlabel : Label connected components in 2-D binary image  
ismember : Array elements that are members of set array  
imclearborder : Suppress light structures connected to image border  
sprintf : Format data into string  
title : Add title  
regionprops : Measure properties of image regions  
sort : Sort array elements  
struct2table : Convert structure array to table  
find : Find indices and values of nonzero elements  
any : Determine if any array elements are nonzero

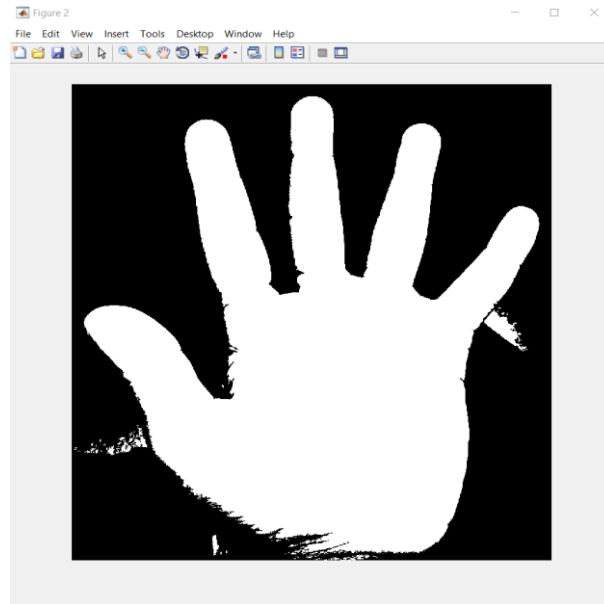
## 4.4 Steps

1. Read the acquired image by imread command and show the same by using imshow command .



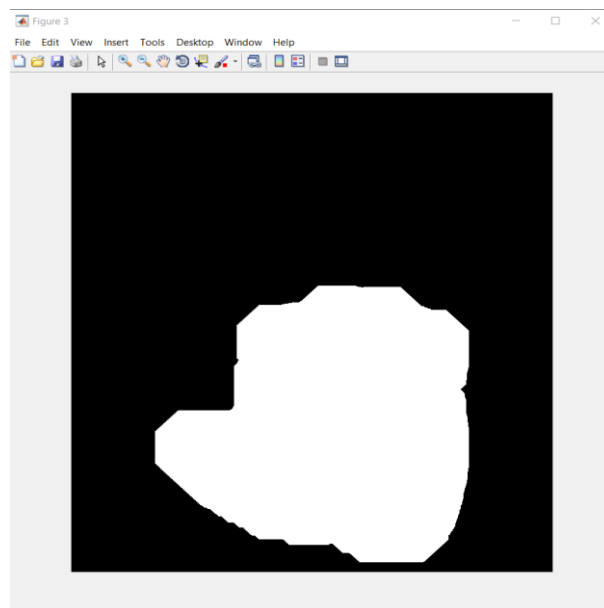
Original image

2. Convert the original image that is RGB image to GRAY image by using `rgb2gray` command and then do the segmentation by using `graythresh` and `im2bw` command to convert the GRAY image to BINARY image.



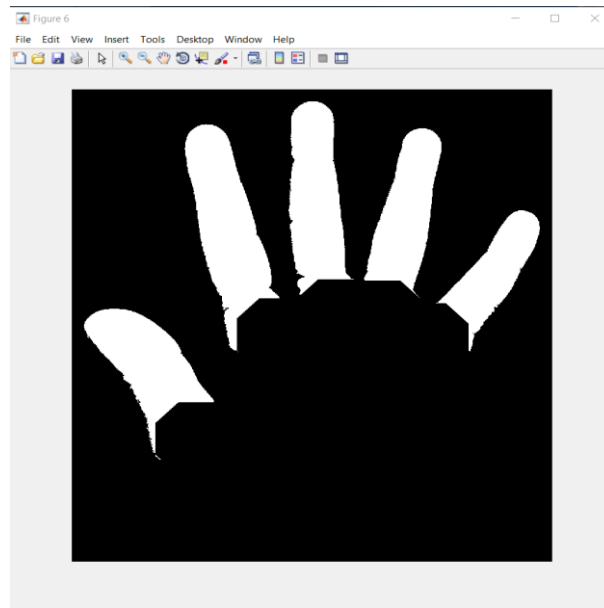
Segmented Image

3. Do the filtering and enhancement of the image to get the proper image for further operations then use the structuring element 'disk' of proper diameter to get the required structure by using `strel` command to use it for morphological operation to get the morphologically opened image.



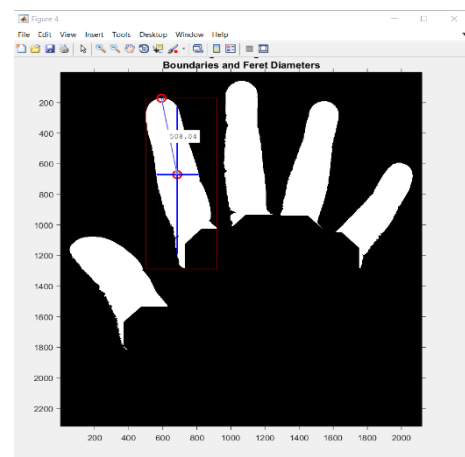
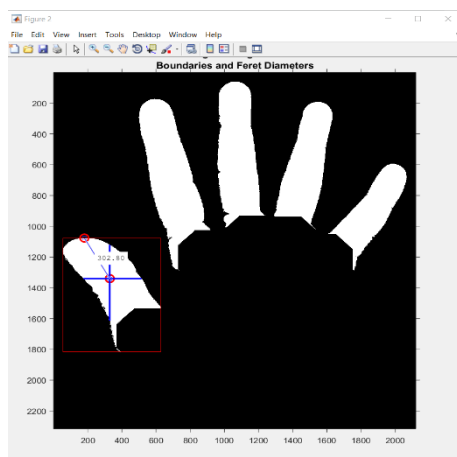
Morphologically opened image

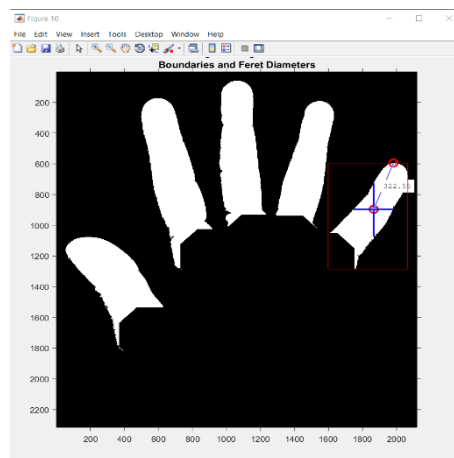
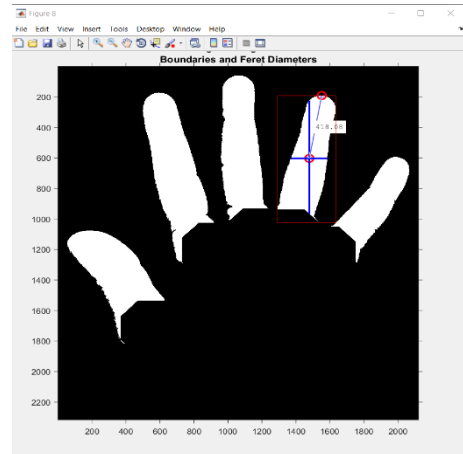
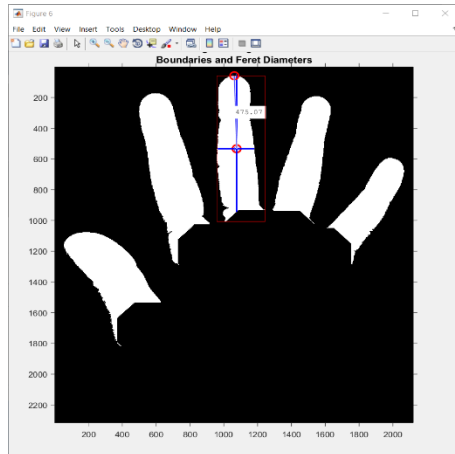
4. Subtract the morphologically opened image from the segmented one to get the required objects (hand's fingers)



Extracted image

5. Label connected components in 2-D binary image. Use 'for' loop to Extract the boradest contour out of many. Use regionprops command to get the centroid and struct2table to get the centroid coordinates. Similarly find the Extrema of each bolb. Use command imdistline by using coordinates of centroid and extremas to get the distance. Results are as follows:





## 4.5 Code

```
%Reading image
n=imread('Image.jpg');
%showing the image
imshow(n)
%converting RGB image to GRAY image
I = rgb2gray(n);
%converting GRAY image to BINARY image using segmentation technique
level=graythresh(I);
m = im2bw(I,level);
imshow(m)
% Extracting objects from binary image
f= bwareafilt(m,100);
% Complementing image
f=imcomplement(f);
% Filling image regions and holes
f= imfill(f,'holes');
imshow(f)
% Using Morphological structuring element for morphological operation
se=strel('disk',500);
```

```

%Morphologically open image
c=imopen(f,se);
imshow(c)
se2=strel('disk',350);
d=imopen(c,se2);
imshow(d)
%subtracting image
x=f-c;
imshow(logical(x))
% Extracting objects from binary image
v=bwareafilt(logical(x),8);
v=bwareafilt(v,5);
imshow(v)
% Labeling connected components in 2-D binary image
[L, num]=bwlabel(v);
for k = 1 : num
    thisBlob = ismember(L, k);
    %% To Extract the broadest contour out of many
    % Fill holes
    binaryImage = imfill(thisBlob, 'holes');
    % Get rid of anything touching the edge of the image
    binaryImage = imclearborder(binaryImage);
    subplot(2, 2, 4);
    imshow(binaryImage, []);
    axis on;
    hold on;
    caption = sprintf('Filled, Cleaned Binary Image with\nBoundaries and Feret
    Diameters');
    title(caption, 'FontSize', 12);
    % Copy the gray scale image to the lower left.
    subplot(2, 2, 3);
    imshow(n, []);
    caption = sprintf('Original Image with\nBoundaries and Feret Diameters');
    title(caption, 'FontSize', 12);
    axis on;
    hold on;
    figure;
    imshow(v, []);% Label the image so we can get the average perpendicular width.
    labeledImage = bwlabel(binaryImage);
    % Let's find the areas
    props = regionprops(labeledImage, 'Area');
    allAreas = sort([props.Area], 'descend');
    % Measure the area
    measurements = regionprops(labeledImage, 'Area');
    caption = sprintf('Original Image with\nBoundaries and Feret Diameters');
    title(caption, 'FontSize', 12);
    axis on;
    hold on;
    [labeledImage, numberOfObjects] = bwlabel(binaryImage);
    measurements = regionprops(labeledImage, 'Centroid');

```

```

t = struct2table(measurements); % New with 2013 releases
xCentroids = t.Centroid(:,1);
yCentroids = t.Centroid(:,2);
% Find the column and row number nearest to the centroid
xCentroidColumns = int32(xCentroids);
yCentroidColumns = int32(yCentroids);
hold on;
% Plot centroids
for k = 1 : numberOfObjects
    plot(xCentroids(k), yCentroids(k), 'ro', 'Markersize', 10, 'linewidth', 2);
end
% Find vertical and horizontal lines
for k = 1 : numberOfObjects
    thisBlob = ismember(labeledImage, k);
    % Look at column yCentroidColumns(k) and find out
    % the top and bottom line of the blob.
    topRow = find(thisBlob(:,xCentroidColumns(k)), 1, 'first');
    bottomRow = find(thisBlob(:,xCentroidColumns(k)), 1, 'last');
    plot([xCentroidColumns(k), xCentroidColumns(k)], [topRow, bottomRow], 'b-',
'LineWidth', 2);
    % Horizontal lines
    leftColumn = find(thisBlob(yCentroidColumns(k), :), 1, 'first');
    rightColumn = find(thisBlob(yCentroidColumns(k), :), 1, 'last');
    plot([leftColumn, rightColumn], [yCentroidColumns(k), yCentroidColumns(k)], 'b-',
'LineWidth', 2);
[labeledImage, numberOfObjects] = bwlabel(binaryImage);
measurements = regionprops(labeledImage, 'Extrema');
t = struct2table(measurements); % New with 2013 releases
xExtremas = t.Extrema(:,1);
yExtremas = t.Extrema(:,2);
for k = 1 : numberOfObjects
    plot(xExtremas(k),yExtremas(k), 'ro', 'Markersize', 10, 'linewidth', 2);
end
h = imdistline(gca,[xExtremas(k) xCentroids(k)],[yExtremas(k) yCentroids(k)]);
end
vertical = any(binaryImage, 2);
horizontal = any(binaryImage, 1);
row1 = find(vertical, 1, 'first'); % Y1
row2 = find(vertical, 1, 'last'); % Y2
column1 = find(horizontal, 1, 'first'); % X1
column2 = find(horizontal, 1, 'last'); % X2
boxY = [row1 row2 row1];
boxX = [column1 column2 column1 column1];
hold on;
plot(boxX,boxY, 'r-');
figure
    imshow(thisBlob, []);

```

---

## CHAPTER- 5

### FINAL DISCUSSIONS

Training of budding engineers in present scenario employability is very important. Only theoretical knowledge is not sufficient for them to achieve maximum in their professional careers. Aspiring Minds (an employability evaluation and certification company) released a National Employability Report by earlier this year. According to this report, nearly 80% engineering graduates in India remains unemployed due to lack of problem solving skills. Indian curriculum is behind times as far programming languages are concerned. Whatever the reasons might be for the poor show, I believe it is sad that India's best universities are nowhere in the Top 100 in the world.

Most Indian engineering graduates, be it IT or Electronics engineers, fail when they are expected to apply basic principles to solve real-world problems. With neither the requisite analytical skills nor a commendable command of the domain, they flounder. They need specific trainings and that's an expense that not everyone in the industry wants to incur. Universities need to bridge this gap and inculcate professional skills and give hands on experience of real-world problems. Students can be encouraged for participation in extracurricular activities, coding challenges, preparation of competitive examinations, and development of projects practical, etc.

The Summer training provided by this organisation is a baby step toward upliftment of inculcating skills among students. Students were asked to study the all concepts of Image Processing, about digital image, structure of image, type of image and key stages of Digital Image Processing. A detailed knowledge of Digital Image Processing is given to the students.

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Trainees used MATLAB for Image Processing in which Image Processing Toolbox provides all the operations needed for image processing. Trainees learnt to image

data import and export and conversion of image type and classes. Trainees learnt many operations such as Image filtering and enhancement, Image segmentation and analysis.

Finally, in short we can conclude that this training as an important tool for students' learning where student got experience of Digital Image Processing and using MATLAB for that. Students need such trainings with more project intensive activities.