ANNEXURE-1

PROJECT REPORT

ON

HAND GESTURE RECOGNITION

BY

SHIVAM KUMAR

1613210154

AMEESHA SINGH

1613210026

NAMRATA

1613210104

SONAM

1613210169

Department of

COMPUTER SCIENCE AND ENGINEERING

G.N.I.O.T.Greater noida

A.K.T.U, LUCKNOW

ANNEXURE-II

PROJECT REPORT

OF

HAND GESTURE RECOGNITION

SUBMITTED IN PARTIAL FULFILLMENT FOR AWARD

OF

DEGREE

IN

COMPUTER SCIENCE AND ENGINEERING

(BATCH 2016-2020)

BY

SHIVAM KUMAR
1613210154
AMEESHA SINGH
1613210026
NAMRATA
1613210104
SONAM
1613210169

Department of

COMPUTER SCIENCE AND ENGINEERING

G.N.I.O.T. Greater Noida

A.K.T.U , LUCKNOW

ANNEXURE-III

Certificate of Originality

I hereby declare that the Project entitled "File Tracking System" submitted to the Department of Computer Science and Engineering, G.N.I.O.T., Greater Noida, U.P. in partial fulfillment for the award of the degree of BACHELOR OF TECHNOLOGY in session 2011-2015 is an authentic record of my own work carried out under the guidance of Mr. "Arvind Tomar" and that the project has not previously formed the basis for the award of any other degree/ diploma.

Place:

Date:

Signature of student
Shivam Kumar.-1613210154
Ameesha singh-1613210026
Namrata- 1613210104
Sonam- 1613210169

This is to certify that the above statement made by the above Candidate/Student is correct to the best of my Knowledge.

Signature of Guide
Arvind Tomar
(Assistant Prof.)

CHAPTER 1 INTRODUCTION

The essential aim of building hand gesture recognition system is to create a natural interaction between human and computer where the recognized gestures can be used for controlling a robot or conveying meaningful information.

Recent developments in computer software and related hardware technology have provided a value-added service to the users. In everyday life, physical gestures are a powerful means of communication. They can economically convey a rich set of facts and feelings. For example, waving one's hand from side to side can mean anything from a "happy goodbye" to "caution". Use of the full potential of physical gesture is also something that most human computer dialogues lack.

The task of hand gesture recognition is one the important and elemental problem in computer vision. With recent advances in information technology and media, automated human interactions systems are built which involve hand processing task like hand detection, hand recognition and hand tracking.

This prompted my interest so I planned to make a software system that could recognize human gestures through computer vision, which is a sub field of artificial intelligence. The purpose of my software through computer vision was to program a computer to "understand" a scene or features in an image.

A first step in any hand processing system is to detect and localize hand in an image. The hand detection task was however challenging because of variability in the pose, orientation, location and scale. Also, different lighting conditions add further variability.

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Current focuses in the field include emotion recognition from face and hand gesture recognition. Users can use simple gestures to control or interact with devices without physically touching them. Many approaches have been made using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of posture, gait, proxemics, and human behaviours is also the subject of gesture recognition techniques. Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse and interact naturally without any mechanical devices. Using the concept of gesture recognition, it is possible to point a finger at this point will move accordingly. This could make conventional input on devices such and even redundant.

1. Gesture types

In computer interfaces, two types of gestures are distinguished We consider online gestures, which can also be regarded as direct manipulations like scaling and rotating. In contrast, offline gestures are usually processed after the interaction is finished; e. g. a circle is drawn to activate a context menu. Offline gestures: Those gestures that are processed after the user interaction with the object. An example is the gesture to activate a menu. Online gestures: Direct manipulation gestures. They are used to scale or rotate a tangible object.

Touchless interface

Touchless user interface is an emerging type of technology in relation to gesture control. Touchless user interface (TUI) is the process of commanding the computer via body motion and gestures without touching a keyboard, mouse, or screen. For example, Microsoft's Kinect is a touchless game interface; however, products such as the Wii are not considered entirely touchless because they are tethered to controllers. Touchless interface in addition to gesture controls are becoming widely popular as they provide the abilities to interact with devices without physically touching them.

There are a number of devices utilizing this type of interface such as, smartphones, laptops, games, and television. Although touchless technology is mostly seen in gaming software, interest is now spreading to other fields including, automotive and healthcare industries. Soon to come, touchless technology and gesture control will be implemented in cars in levels beyond voice recognition.

The aim of the project then is to explore the use of touchless interaction within surgical settings, allowing images to be viewed, controlled and manipulated without contact through the use of camera-based gesture recognition technology. In particular, the project seeks to understand the challenges of these environments for the design and deployment of such systems, as well as articulate the ways in which these technologies may alter surgical practice. While our primary concerns here are with maintaining conditions of asepsis, the use of these touchless gesture-based technologies offers other potential uses.

Elliptic Labs software suite delivers gesture and proximity functions by re-using the existing earpiece and microphone, previously used only for audio. Ultrasound signals sent through the air from speakers integrated in smartphones and tablets bounce against a hand/object/head and are recorded by microphones, also integrated in these devices. In this way, Elliptic Labs' technology recognizes your hand gestures and uses them to move objects on a screen, similarly to the way bats use echolocation to navigate. While these companies stand at the forefront of touchless technology for the future in this time, there are many other companies and products that are currently trending as well and may also add value to this new field. Here are some of many examples.

Input devices

The ability to track a person's movements and determine what gestures they may be performing can be achieved through various tools. The kinetic user interfaces (KUIs) are an emerging type of user interfaces that allow users to interact with computing devices through the motion of objects and bodies. Examples of KUIs include tangible user interfaces and motion-aware games such as Wii and Microsoft's Kinect, and other interactive projects. Although there is a large amount of research done in image/videobased gesture recognition, there is some variation within the tools and environments used between implementations.

Wired gloves. These can provide input to the computer about the position and rotation of the hands using magnetic or inertial tracking devices. Furthermore, some gloves can detect finger bending with a high degree of accuracy (5-10 degrees), or even provide haptic feedback to the user, which is a simulation of the sense of touch. The first commercially available hand-tracking glove-type device was the Data Glove, a glove-type device which could detect hand position, movement and finger bending. This uses fibre optic cables running down the back of the hand. Light pulses are created and when the fingers are bent, light leaks through small cracks and the loss is registered, giving an approximation of the hand pose.

Depth-aware cameras. Using specialized cameras such as structured light or time-of-flight cameras, one can generate a depth map of what is being seen through the camera at a short range, and use this data to approximate a 3d representation of what is being seen. These can be effective for detection of hand gestures due to their short range capabilities.

Stereo cameras. Using two cameras whose relations to one another are known, a 3d representation can be approximated by the output of the cameras. To get the cameras' relations, one can use a positioning reference such as a lexian-stripe or infrared emitters. In combination with direct motion measurement (6D-Vision) gestures can directly be detected.

Gesture-based controllers. These controllers act as an extension of the body so that when gestures are performed, some of their motion can be conveniently captured by software. An example of emerging gesture-based motion capture is through skeletal hand tracking, which is being developed for virtual reality and augmented reality applications. An example of this technology is shown by tracking companies uses and Gestion, which allow users to interact with their surrounding without controllers.

Another example of this is **mouse gesture tracking,** where the motion of the mouse is correlated to a symbol being drawn by a person's hand, as is the Wii Remote or the Myo armband or the mForce Wizard wristband, which can study changes in acceleration over time to represent gestures. Devices such as the LG Electronics Magic Wand, the Loop and the Scoop use Hillcrest Labs' Freespace technology, which uses MEMS accelerometers, gyroscopes and other sensors to translate gestures into cursor movement. The software also compensates for human tremor and inadvertent movement. AudioCubes are another example. The sensors of these smart light emitting cubes can be used to sense hands and fingers as well as other objects nearby, and can be used to process data. Most applications are in music and sound synthesis,[33] but can be applied to other fields.

Single camera. A standard 2D camera can be used for gesture recognition where the resources/environment would not be convenient for other forms of image-based recognition. Earlier it was thought that single camera may not be as effective as stereo or depth aware cameras, but some companies are challenging this theory. Softwarebased gesture recognition technology using a standard 2D camera that can detect robust hand gestures.

Gesture recognition features

More
accurate High
stability
Time saving to unlock a device
The major application areas of gesture recognition in the current scenario
are Automotive sector
Consumer electronics
sector Transit sector
Gaming sector
To unlock smartphones
Defence Home automation
Automated sign language translation
Gesture recognition technology has been considered to be the highly successful technology as it saves time to unlock any device.

Gesture recognition can be conducted with techniques from computer vision and image processing. The literature includes ongoing work in the computer vision field on capturing gestures or more general human pose and movements by cameras connected to a computer.Gesture recognition and pen computing: Pen computing reduces the hardware impact of a system and also increases the range of physical world objects usable for control beyond traditional digital objects like keyboards and mice. Such implementations could enable a new range of hardware that does not require monitors. This idea may lead to the creation of holographic display. The term gesture recognition has been used to refer more narrowly to non-text-input handwriting symbols, such as inking on a graphics tablet, multi-touch gestures, and mouse gesture recognition. This is computer interaction through the drawing of symbols with a pointing device cursor. (see Pen computing)

DIGITAL IMAGE PROCESSING

The digital image processing deals with developing a digital system that performs operations on an digital image. Image processing is reckoned as one of the most rapidly involving fields of the software industry with growing applications in all areas of work. It holds the possibility of developing the ultimate machines in future, which would be able to perform the visual function of living beings. As such, it forms the basis of all kinds of visual automation.

Signal processing is a discipline in electrical engineering and in mathematics that deals with analysis and processing of analog and digital signals , and deals with storing , filtering , and other operations on signals. These signals include transmission signals , sound or voice signals , image signals , and other signals. Out of all these signals , the field that deals with the type of signals for which the input is an image and the output is also an image is done in image processing. As it name suggests, it deals with the processing on images.

Machine vision or computer vision deals with developing a system in which the input is an image and the output is some information. For example: Developing a system that scans human face and opens any kind of lock. This system would look something like this.

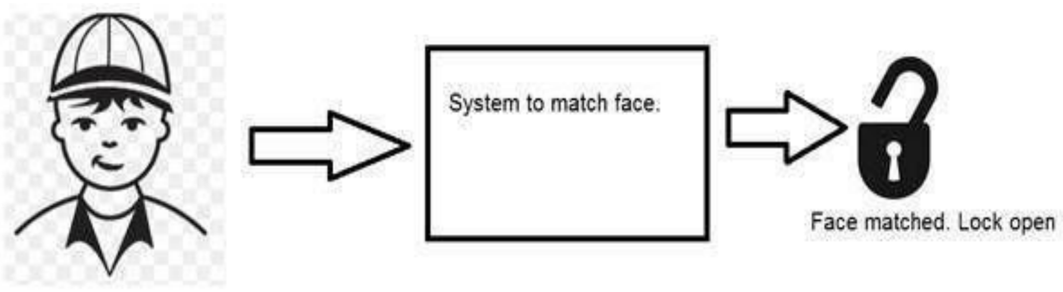
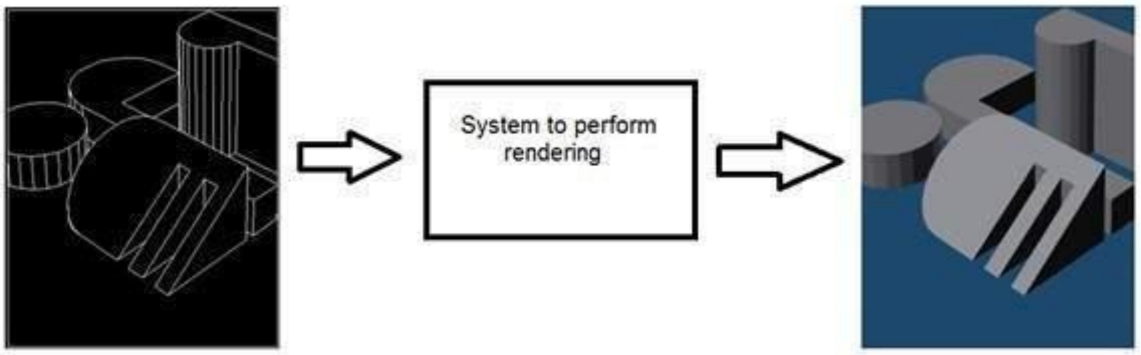


Figure:1.1

Computer graphics deals with the formation of images from object models, rather then the image is captured by some device. For example: Object rendering.



Generating an image from an object model. Such a system would look something like this.

Figure:1.2

Components of Digital Image Processing

- Image Sensor
- Image Processing Software
- Image Processing Hardware
- Computer
- Mass Storage

Image Sensor

An image sensor or imager is a sensor that detects and conveys information used to make an image. It does so by converting the variable attenuation of light waves (as they pass through or reflect off objects) into signals, small bursts of current that convey the information. The waves can be light or other electromagnetic radiation. Image sensors are used in electronic imaging devices of both analog and digital types, which include digital cameras, camera modules, medical imaging equipment, night vision equipment such as thermal imaging devices, radar, sonar, and others. As technology changes, digital imaging tends toreplace analog imaging.

Early analog sensors for visible light were video camera tubes. Currently, used types are semiconductor charge-coupled devices (CCD) or active pixel sensors in complementary metal–oxide–semiconductor (CMOS) or N-type metaloxidesemiconductor (NMOS, Live MOS) technologies. Analog sensors for invisible radiation tend to involve vacuum tubes of various kinds. Digital sensors include flat panel detectors.

In February 2018, researchers at Dartmouth College announced a new image sensing technology that the researchers call QIS, for Quanta Image Sensor. Instead of pixels, QIS chips have what the researchers call "jots." Each jot can detect a single particle of light, called a photon.

Image Processing Software

Image processing software is software that is designed to manipulate digital images. In particular, it captures the image if that hasn't already been done, it converts it to a digital form, and it performs a manipulation or manipulations on it These are examples of image processing at work.

- Adobe Camera Raw
- Amira (software)
- Analysis of Functional Neuroimages
- Analyse (imaging software)
- ANIMAL (image processing)
- Aphelion (software)
- AutoCollage 2008
- Aviso (software)

Digital Computer

It is simply a digital computer which is used to manipulate processing algorithm in order to display correct image. Basically, it used to process digital image. In computer science, a digital electronic computer is a computer machine which is both an electronic computer and a digital computer. Examples of a digital electronic computers include the IBM PC, the Apple Macintosh as well as modern smartphones. When computers that were both digital and electronic appeared, they displaced almost all other kinds of computers, but computation has historically been performed in various non-digital and non-electronic ways: the Lehmer sieve is an example of a digital non-electronic computer, while analog computers are examples of non-digital computers which can be electronic (with analog electronics), and mechanical computers are examples of nonelectronic computers (which may be digital or not). An example of a computer which is both non-digital and non-electronic is the ancient Antikythera mechanism found in Greece. All kinds of computers, whether they are digital or analog, and electronic or non-electronic, can be

Turing complete if they have sufficient memory. A digital electronic computer is not necessarily a programmable computer, a stored program computer, or a general-purpose computer, since in essence a digital electronic computer can be built for one specific application and be non-reprogrammable. As of 2014, most personal computers and smartphones in people's homes that use multicore central processing units (such as AMD FX, Intel Core i7, or the multicore varieties of ARM-based chips) are also parallel computers using the MIMD (multiple instructions - multiple data) paradigm, a technology previously only used in digital electronic supercomputers. As of 2014, most digital electronic supercomputers are also cluster computers, a technology that can be used at home in the form of small Beowulf clusters. Parallel computation is also possible with non-digital or non-electronic computers. An example of a parallel computation system using the abacus would be a group of human computers using a number of abacus machines for computation and communicating using natural language.

A digital computer can perform its operations in the decimal system, in binary, in ternary or in other numeral systems. As of 2014, all digital electronic computers commonly used, whether personal computers or supercomputers, are working in the binary number system and also use binary logic. A few ternary computers using ternary logic were built mainly in the Soviet Union as research projects.

A digital electronic computer is not necessarily a transistorized computer: before the advent of the transistor, computers used vacuum tubes. The transistor enabled electronic computers to become much more powerful, and recent and future developments in digital electronics may enable humanity to build even more powerful electronic computers. One such possible development is the memristor.

People living in the beginning of the 21st century use digital electronic computers for storing data, such as photos, music, documents, and for performing complex mathematical computations or for communication, commonly over a worldwide computer network called the internet which connects many of the world's computers. All these activities made possible by digital electronic computers could, in essence, be performed with non-digital or non-electronic computers if they were sufficiently powerful, but it was only the combination of electronics technology with digital computation in binary that enabled humanity to reach the computation power necessary for today's computing. Advances in quantum computing, DNA computing, optical computing or other technologies could lead to the development of more powerful computers in the future.

Digital computers are inherently best described by discrete mathematics, while analog computers are most commonly associated with continuous mathematics.

The philosophy of digital physics views the universe as being digital. Konrad Zuse wrote a book known as Rechnender Raum in which he described the whole universe as one all-encompassing computer.

Mass Storage

It is storage device which is used to store the input and output image for processing. It also keeps processing algorithm. In computing, mass storage [citation needed] refers to the storage of large amounts of data in a persisting and machine-readable fashion. Devices and/or systems that have been described as mass storage include tape libraries, RAID systems, and a variety of computer drives such as hard disk drives, magnetic tape drives, magneto-optical disc drives, optical disc drives, memory cards, and solid-state drives. It also includes experimental forms like holographic memory. Mass storage includes devices with removable and non-removable media. It does not include random access memory (RAM).

There are two broad classes of mass storage: local data in devices such as smartphones or computers, and enterprise servers and data centers for the cloud. For local storage, SSDs are on the way to replacing HDDs. Considering the mobile segment from phones to notebooks, the majority of systems today is based on NAND Flash. As for Enterprise and data centres, storage tiers have established using a mix of SSD and HDD.

Stages in Digital Image Processing

- Image Acquisition
- Image enhancement
- Image Restoration
- Color Image Processing
- Wavelets
- Morphological Image Processing
- Image Segmentation
- Image Compression
- Image Recognition

Image Acquisition

Image Acquisition in Digital Image Processing means capturing or retrieving image from object or storage. In image processing, it is defined as the action of retrieving an image from some source, usually a hardware-based source for processing. It is the first step in the workflow sequence because, without an image, no processing is possible.

Image Enhancement

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. For example, you can remove noise, sharpen, or brighten an image, making it easier to identify key features.

Here are some useful examples and methods of image enhancement:

- Filtering with morphological operators
- Histogram equalization
- Noise removal using a Wiener filter
- Linear contrast adjustment
- Median filtering
- Unsharp mask filtering

Image Restoration

Image Restoration is the operation of taking a corrupt/noisy image and estimating the clean, original image. Corruption may come in many forms such as motion blur, noise and camera mis-focus.[1] Image restoration is performed by reversing the process that blurred the image and such is performed by imaging a point source and use the point source image, which is called the Point Spread Function (PSF) to restore the image information lost to the blurring process.

Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbour procedure) provided by imaging packages use no a priori model of the process that created the image.

With image enhancement noise can effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a fluorescence microscope, resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object.

The objective of image restoration techniques is to reduce noise and recover resolution loss. Image processing techniques are performed either in the image domain or the frequency domain. The most straightforward and a conventional technique for image restoration is deconvolution, which is performed in the frequency domain and after computing the Fourier transform of both the image and the PSF and undo the resolution loss caused by the blurring factors. This deconvolution technique, because of its direct inversion of the PSF which typically has poor matrix condition number, amplifies noise and creates an imperfect deblurred image. Also, conventionally the blurring process is assumed to be shift-invariant. Hence more sophisticated techniques, such as regularized deblurring, have been developed to offer robust recovery under different types of noises and blurring functions. It is of 3 types: 1. Geometric correction 2. radiometric correction 3. noise removal

- Contrast-limited adaptive histogram equalization (CLAHE)
- Decorrelation stretch

Colour Image Processing

Color Image Processing on Digital Image. The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. The human visual system can distinguish hundreds of thousands of different colour shades and intensities, but only around 100 shades of grey. Therefore, in an image, a great deal of extra information may be contained in the colour, and this extra information can then be used to simplify image analysis, e.g. object identification and extraction based on colour.
The saturation is determined by the excitation purity, and depends on the amount of white light mixed with the hue. A pure hue is fully saturated, i.e. no white light mixed in. Hue and saturation together determine the chromaticity for a given colour. Finally, the intensity is determined by the actual amount of light, with more light corresponding to more intense colours.

Achromatic light has no colour - its only attribute is quantity or intensity. Greylevel is a measure of intensity. The intensity is determined by the energy, and is therefore a physical quantity. On the other hand, brightness or luminance is determined by the perception of the colour, and is therefore psychological. Given equally intense blue and green, the blue is perceived as much darker than the green. Note also that our perception of intensity is nonlinear, with changes of normalised intensity from 0.1 to 0.11 and from 0.5 to 0.55 being perceived as equal changes in brightness.

Colour depends primarily on the reflectance properties of an object. We see those rays that are reflected, while others are absorbed. However, we also must consider the colour of the light source, and the nature of human visual system.

For example, an object that reflects both red and green will appear green when there is green but no red light illuminating it, and conversely it will appear red in the absense of green light. In pure white light, it will appear yellow (= red + green

Wavelets

A wavelet is a wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero. It can typically be visualized as a "brief oscillation" like one recorded by a seismograph or heart monitor. Generally, wavelets are intentionally crafted to have specific properties that make them useful for signal processing. Using a "reverse, shift, multiply and integrate" technique called convolution, wavelets can be combined with known portions of a damaged signal to extract information from the unknown portions.

Seismic wavelet
For example, a wavelet could be created to have a frequency of Middle C and a short duration of roughly a 32nd note. If this wavelet were to be convolved with a signal created from the recording of a song, then the resulting signal would be useful for determining when the Middle C note was being played in the song. Mathematically, the wavelet will correlate with the signal if the unknown signal contains information of similar frequency. This concept of correlation is at the core of many practical applications of wavelet theory.

Image Segmentation

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.[1][2] Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).[1] When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like Marching cubes.

Image Compression

Image compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission. Algorithms may take advantage of visual perception and the statistical properties of image data to provide superior results compared with generic data compression methods which are used for other digital data.

Image Recognition

Image recognition, in the context of machine vision, is the ability of software to identify objects, places, people, writing and actions in images. Computers can use machine vision technologies in combination with a camera and artificial intelligence software to achieve image recognition.

Image recognition is used to perform a large number of machine-based visual tasks, such as labeling the content of images with meta-tags, performing image content search and guiding autonomous robots, self-driving cars and accident avoidance systems.

While human and animal brains recognize objects with ease, computers have difficulty with the task. Software for image recognition requires deep machine learning. Performance is best on convolutional neural net processors as the specific task otherwise requires massive amounts of power for its computeintensive nature. Image recognition algorithms can function by use of comparative 3D models, appearances from different angles using edge detection or by components. Image recognition algorithms are often trained on millions of pre-labeled pictures with guided computer learning.

Current and future applications of image recognition include smart photo libraries, targeted advertising, the interactivity of media, accessibility for the visually impaired and enhanced research capabilities. Google, Facebook, Microsoft, Apple and Pinterest are among the many companies that are investing significant resources and research into image recognition and related applications. Privacy concerns over image recognition and similar technologies are controversial as these companies can pull a large volume of data from user photos uploaded to their social media platforms.

Image formation on digital cameras

In the digital cameras , the image formation is not due to the chemical reaction that take place , rather it is a bit more complex then this. In the digital camera , a CCD array of sensors is used for the image formation.

CCD stands for charge-coupled device. It is an image sensor, and like other sensors it senses the values and converts them into an electric signal. In case of CCD it senses the image and convert it into electric signal . This CCD is actually in the shape of array or a rectangular grid. It is like a matrix with each cell in the matrix contains a censor that senses the intensity of photon.

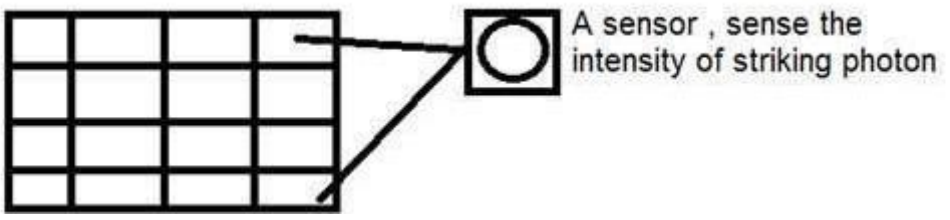


Figure:1.3

Like analog cameras , in the case of digital too , when light falls on the object , the light reflects back after striking the object and allowed to enter inside the camera. Each sensor of the CCD array itself is an analog sensor. When photons of light strike on the chip , it is held as a small electrical charge in each photo sensor. The response of each sensor is directly equal to the amount of light or (photon) energy striked on the surface of the sensor.

Since we have already define an image as a two dimensional signal and due to the two dimensional formation of the CCD array , a complete image can be achieved from this CCD array. It has limited number of sensors , and it means a limited detail can be captured by it. Also each sensor can have only one value against the each photon particle that strike on it. So the number of photons striking(current) are counted and stored. In order to measure accurately these , external CMOS sensors are also attached with CCD array.

Introduction to pixel

The value of each sensor of the CCD array refers to each the value of the individual pixel. The number of sensors = number of pixels. It also means that each sensor could have only one and only one value.

Storing image

The charges stored by the CCD array are converted to voltage one pixel at a time. With the help of additional circuits , this voltage is converted into a digital information and then it is stored.

Each company that manufactures digital camera, make their own CCD sensors. That include , Sony , Mistubishi , Nikon ,Samsung , Toshiba , FujiFilm , Canon e.t.c.

Apart from the other factors , the quality of the image captured also depends on the type and quality of the CCD array that has been used.

Pixel

Pixel is the smallest element of an image. Each pixel correspond to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255. The value of a pixel at any point correspond to the intensity of the light photons striking at that point. Each pixel store a value proportional to the light intensity at that particular location.

Gray level

The value of the pixel at any point denotes the intensity of image at that location, and that is also known as gray level.

We will see in more detail about the value of the pixels in the image storage and bits per pixel tutorial, but for now we will just look at the concept of only one pixel value.

Pixel value. (0)

As it has already been define in the beginning of this tutorial, that each pixel can have only one value and each value denotes the intensity of light at that point of the image.

We will now look at a very unique value 0. The value 0 means absence of light. It means that 0 denotes dark, and it further means that whenever a pixel has a value of 0, it means at that point, black color would be formed.

Have a look at this image matrix

0	0	0

0	0	0
0	0	0

Now this image matrix has all filled up with 0. All the pixels have a value of 0. If we were to calculate the total number of pixels form this matrix, this is how we are going to do it.

Total no of pixels = total no. of rows X total no. of columns

= 3 X 3 =

9.

It means that an image would be formed with 9 pixels, and that image would have a dimension of 3 rows and 3 column and most importantly that image would be black.
The resulting image that would be made would be something like this

RGB to Grey Scale conversion

We have already defined the RGB color model and gray scale format in our tutorial of Image types. Now we will convert an color image into a grayscale image. There are two methods to convert it. Both has their own merits and demerits. The methods are:

- Average method
- Weighted method or luminosity method

Average method

Average method is the most simple one. You just have to take the average of three colors. Since its an RGB image, so it means that you have add r with g with b and then divide it by 3 to get your desired grayscale image.

Its done in this way.

Grayscale = (R + G + B / 3)

For example:

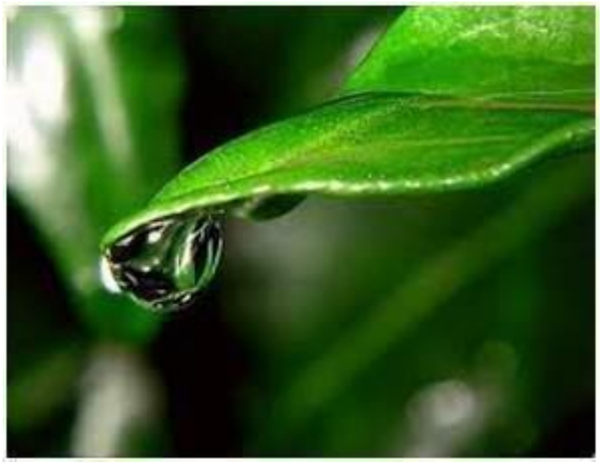


Figure:1.4

If you have an color image like the image shown above and you want to convert it into grayscale using average method. The following result would appear.

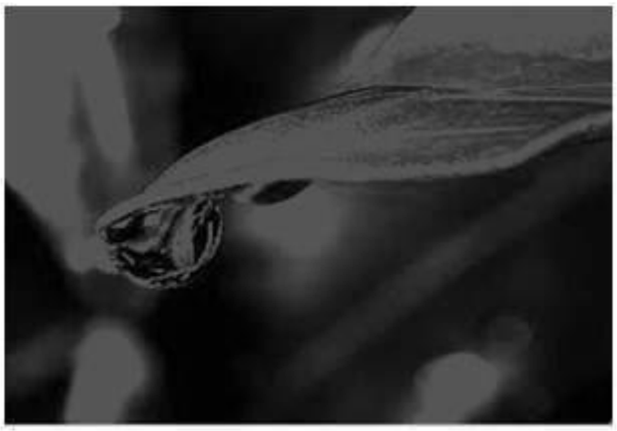


Figure:1.5

Explanation

There is one thing to be sure, that something happens to the original works. It means that our average method works. But the results were not as expected. We wanted to convert the image into a grayscale, but this turned out to be a rather black image.

Problem

This problem arises due to the fact, that we take average of the three colors. Since the three different colors have three different wavelength and have their own contribution in the formation of image, so we have to take average according to their contribution, not done it averagely using average method. Right now what we are doing is this,

33% of Red, 33% of Green, 33% of Blue

We are taking 33% of each, that means, each of the portion has same contribution in the image. But in reality thats not the case. The solution to this has been given by luminosity method.

Weighted method or luminosity method

You have seen the problem that occur in the average method. Weighted method has a solution to that problem. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.
It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.

So the new equation that form is:

New grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B)).

According to this equation, Red has contributed 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

Applying this equation to the image, we get this

Original Image:

Figure:1.6

Grayscale Image:

Figure:1.7

Explanation

As you can see here, that the image has now been properly converted to grayscale using weighted method. As compare to the result of average method, this image is more brighter.

Sampling

Sampling has already been introduced in our tutorial of introduction to signals and system. But we are going to discuss here more.

Here what we have discussed of the

sampling. The term sampling refers to take

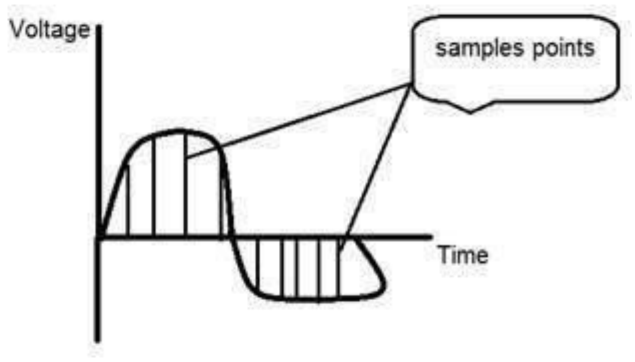
samples

We digitize x axis in sampling

It is done on independent variable

In case of equation $y = \sin(x)$, it is done on x variable

It is further divided into two parts , up sampling and down sampling



If you will look at the above figure, you will see that there are some random variations in the signal. These variations are due to noise. In sampling we reduce this noise by taking samples. It is obvious that more samples we take, the quality of the image would be more better, the noise would be more removed and same happens vice versa.

However, if you take sampling on the x axis, the signal is not converted to digital format, unless you take sampling of the y-axis too which is known as quantization. The more samples eventually means you are collecting more data, and in case of image, it means more pixels.

What is a mask

A mask is a filter. Concept of masking is also known as spatial filtering. Masking is also known as filtering. In this concept we just deal with the filtering operation that is performed directly on the image.

A sample mask has been shown below

-1	0	1
-1	0	1
-1	0	1

What is filtering

The process of filtering is also known as convolving a mask with an image. As this process is same of convolution so filter masks are also known as convolution masks.

How it is done

The general process of filtering and applying masks is consists of moving the filter mask from point to point in an image. At each point (x,y) of the original image, the response of a filter is calculated by a pre defined relationship. All the filters values are pre defined and are a standard.

Types of filters

Generally there are two types of filters. One is called as linear filters or smoothing filters and others are called as frequency domain filters.

Why filters are used?

Filters are applied on image for multiple purposes. The two most common uses are as following:

- Filters are used for Blurring and noise reduction
- Filters are used or edge detection and sharpness

Blurring and noise reduction

Filters are most commonly used for blurring and for noise reduction. Blurring is used in pre processing steps, such as removal of small details from an image prior to large object extraction.

Masks for blurring

The common masks for blurring are.

- Box filter
- Weighted average filter

In the process of blurring we reduce the edge content in an image and try to make the transitions between different pixel intensities as smooth as possible.

Noise reduction is also possible with the help of blurring.

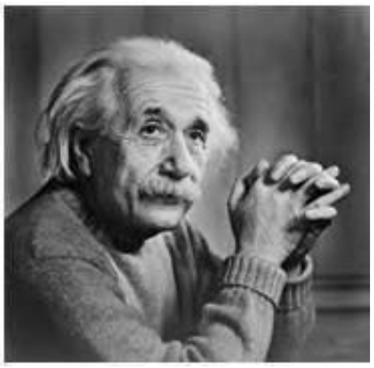
Edge Detection and sharpness

Masks or filters can also be used for edge detection in an image and to increase sharpness of an image.

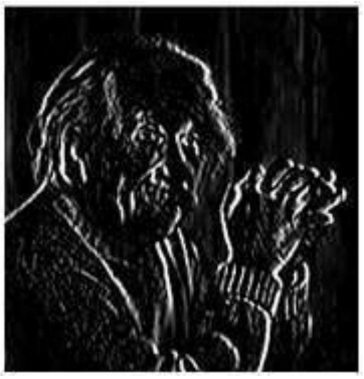
What are edges?

We can also say that sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.A picture with edges is shown below.

Original picture



Same picture with edges



Blurring

In blurring, we simple blur an image. An image looks more sharp or more detailed if we are able to perceive all the objects and their shapes correctly in it. For example. An image with a face, looks clear when we are able to identify eyes, ears, nose, lips, forehead e.t.c very clear. This shape of an object is due to its edges. So in blurring, we simple reduce the edge content and makes the transition form one color to the other very smooth.

Convolution Theorem

The relationship between the spatial domain and the frequency domain can be established by convolution theorem.

The convolution theorem can be represented as.

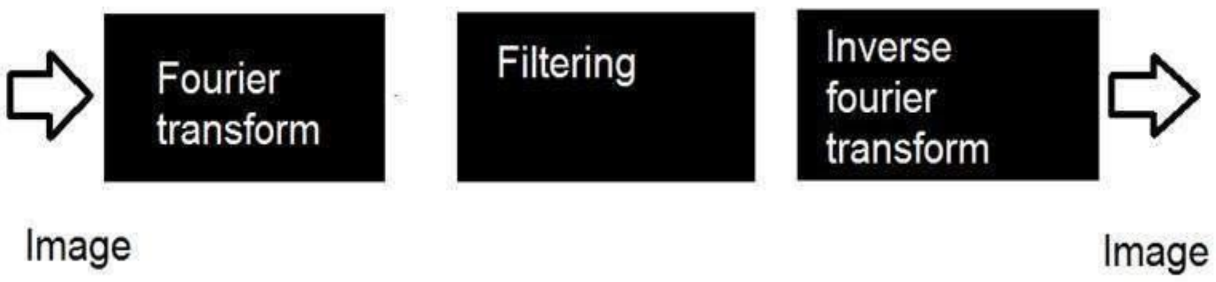
$$f(x,y)*h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

$$f(x,y)h(x,y) \Leftrightarrow F(u,v)*H(u,v)$$

$$h(x,y) \Leftrightarrow H(u,v)$$

It can be stated as the convolution in spatial domain is equal to filtering in frequency domain and vice versa.

The filtering in frequency domain can be represented as following:



The steps in filtering are given below.

- At first step we have to do some pre – processing an image in spatial domain, means increase its contrast or brightness
- Then we will take discrete Fourier transform of the image
- Then we will center the discrete Fourier transform, as we will bring the discrete Fourier transform in center from corners

- Then we will apply filtering, means we will multiply the Fourier transform by a filter function
- Then we will again shift the DFT from center to the corners
- Last step would be take to inverse discrete Fourier transform, to bring the result back from frequency domain to spatial domain
- And this step of post processing is optional, just like pre processing , in which we just increase the appearance of image.

Filters

The concept of filter in frequency domain is same as the concept of a mask in convolution.

After converting an image to frequency domain, some filters are applied in filtering process to perform different kind of processing on an image. The processing include blurring an image, sharpening an image e.t.c.

The common type of filters for these purposes are:

- high pass filter
- low pass filter

Low-pass Filter

A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The exact frequency response of the filter depends on thefilter design. • Used for smoothing and blurring.

High-pass Filter

A high-pass filter is an electronic filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency. The amount of attenuation for each frequency depends on the filter design.

- Used for sharpening and edge detection.

Both Low-pass and High-pass Filter are of three types

- Ideal

- Butterworth
- Gaussian

BIOMETRICS

Biometric systems are systems that recognize or verify human beings. Some of the most important biometric features are based physical features like hand, finger, face and eye. For instance, finger print recognition utilizes of ridges and furrows on skin surface of the palm and fingertips. Hand gesture detection is related to the location of the presence of a hand in still image or in sequence of images i.e. moving images. Other biometric features are determined by human behaviour like voice, signature and walk. The way humans generate sound for mouth, nasal cavities and lips is used for voice recognition. Signature recognition looks at the pattern, speed of the pen when writing one’s signature.

Biometric identifiers are the distinctive, measurable characteristics used to label and describe individuals. Biometric identifiers are often categorized as physiological versus behavioral characteristics. Physiological characteristics are related to the shape of the body. Examples include, but are not limited to fingerprint, palm veins, face recognition, DNA, palm print, hand geometry, iris recognition, retina and odour/scent. Behavioral characteristics are related to the pattern of behavior of a person, including but not limited to typing rhythm, gait, and voice.[5][note 2] Some researchers have coined the term behaviometrics to describe the latter class of biometrics.

More traditional means of access control include token-based identification systems, such as a driver's license or passport, and knowledge-based identification systems, such as a password or personal identification number.[3] Since biometric identifiers are unique to individuals, they are more reliable in verifying identity than token and knowledge-based methods; however, the collection of biometric identifiers raises privacy concerns about the ultimate use of this information

HAND GESTURE DETECTION AND RECOGNITION

DETECTION

Hand detection is related to the location of the presence of a hand in a still image or sequence of images i.e. moving images. In case of moving sequences, it can be followed by tracking of the hand in the scene but this is more relevant to the applications such as sign language. The underlying concept of hand detection is that human eyes can detect objects which machines cannot with that much accuracy as that of a human. From a machine point of view, it is just like a man fumble around with his senses to find an object.

The factors, which make the hand detection task difficult to solve, are:

Variations in image plane and pose

The hands in the image vary due to rotation, translation and scaling of the camera pose or the hand itself. The rotation can be both in and out of the plane.

Skin Color and Other Structure Components

The appearance of a hand is largely affected by skin colour, size and also the presence or absence of additional features like hairs on the hand further adds to this variability.

Lighting Condition and Background

As shown in Figure 1.1 light source properties affect the appearance of the hand. Also, the background, which defines the profile of the hand, is important and cannot be ignored.

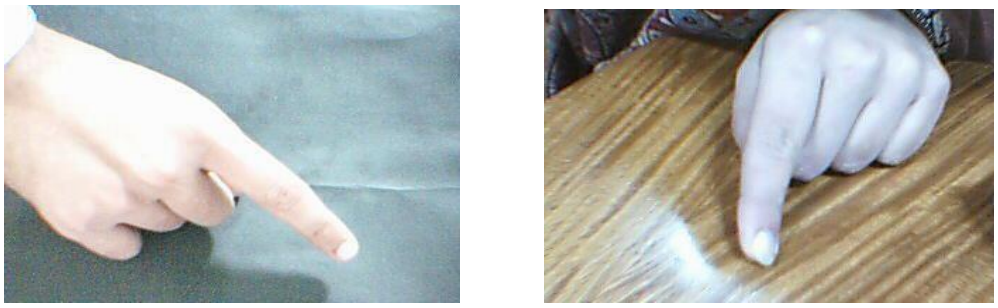
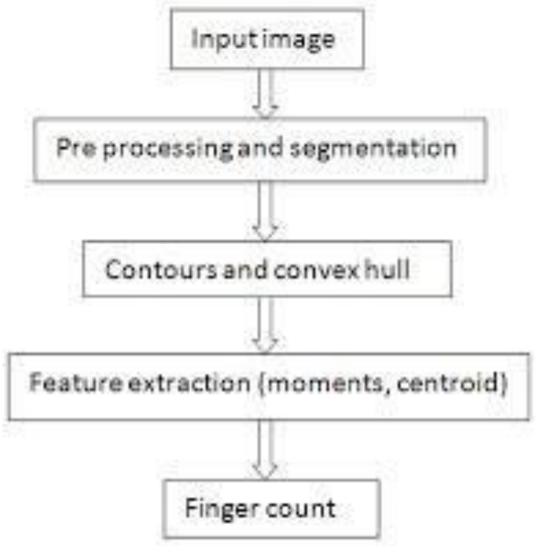


Figure: 1.8: Lighting Condition and Background

RECOGNITION

Hand detection and recognition have been significant subjects in the field of computer vision and image processing during the past 30 years. There have been considerable achievements in these fields and numerous approaches have been proposed. However, the typical procedure of a fully automated hand gesture recognition system can be illustrated in the **Figure 1.2** below:



Input image
Pre processing and segmentation
Contours and convex hull
Feature extraction (moments, centroid)
Finger count

Figure 1.9: Hand Gesture Recognition Flow Chart

MOTIVATION

Biometric technologies make use of various physical and behavioural characteristics of human such as fingerprints, expression, face, hand gestures and movement. These features are then processed using sophisticated machines for detection and recognition and hence used for security purposes. Unlike common security measures such as passwords, security cards that can easily be lost, copied or stolen; these biometric features are unique to individuals and there is little possibility that these pictures can be replaced or altered.

Among the biometric sector hand gesture recognition are gaining more and more attention because of their demand regarding security for law enforcement agency as well as in private sectors such as surveillance systems.

In video conferencing system, there is a need to automatically control the camera in such a way that the current speaker always has the focus. One simple approach to this is to guide the camera based on sound or simple cues such as motion and skin colour.

Hand gestures are important to intelligent human and computer interaction to build fully automated systems that analyse information contained in images, fast and efficient hand gesture recognition algorithms are required.

SCOPE

The scope of this project is to build a real time gesture classification system that can automatically detect gestures in natural lighting condition. In order to accomplish this objective, a real time gesture-based system is developed to identify gestures.

This system will work as one of futuristic of Artificial Intelligence and computer vision with user interface. It creates method to recognize hand gesture based on different parameters. The main priority of this system is to simple, easy and user friendly without making any special hardware. All computation will occur on single PC or workstation. Only special hardware will use to capture frames (Web Camera) or Laptop camera.

SOFTWARE TOOLS

Due to the simplicity and easy availability of various libraries in python, the aim was to build a program that uses OpenCV (Open Source Computer Vision) to recognize hand and perform specified task.

OBJECTIVES

The objective of this project is to create a complete system to detect, recognize and interpret the hand gestures through computer vision.

CHAPTER 2

Software Requirements

- Python
- OpenCV
- Java GUI
- NetBeans IDE

Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Syntax and semantics

Main article: Python syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation

Main article: Python syntax and semantics § Indentation
Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule.

Statements and control flow
Python's statements include (among others):

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.
The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

The while statement, which executes a block of code as long as its condition is true.

The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.

The raise statement, used to raise a specified exception or re-raise a caught exception. The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming. The def statement, which defines a function or method.

The with statement, from Python 2.5 released on September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.

The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.

The assert statement, used during debugging to check for conditions that ought to apply. The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import * or from <module name> import
<definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ...

The print statement was changed to the print() function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutinelike functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Methods

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self-parameter to access instance data, in contrast to the implicit self (or this) in some other objectoriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

OpenCV

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell, and Ruby have been developed to encourage adoption by a wider audience.

Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform. All of the new developments and algorithms in OpenCV are now developed in the C++ interface

Java GUI

Swing is a GUI widget toolkit for Java.[1] It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

NetBeans

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5, and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

Hardware Requirement • THE

WEBCAM SYSTEM (USB PORT)

Below is the summary of the specifications of the camera which this system required:

Resolution:	**640**x480
Video frame rate:	30fps @640x480
Pixel depth:	Minimum 1.3-mega pixels
Connection port:	USB

In my project web cam was attached via USB port of the computer. The web cam worked by continually capturing the frames. In order to capture a particular frame, the user just need to select the particular Algorithm METHOD button on the interface and the hand was detected in the particular frame. The web cam took color pictures, which were then converted into grayscale format. The main reason of sticking to grayscale was the extra amount of processing required to deal with color images.

CHAPTER 3

LITERATURE REVIEW

INTRODUCTION

Hand gesture recognition research is classified in three categories. First "**Glove based Analysis**" attaching sensor with gloves mechanical or optical to transduces flexion of fingers into electrical signals for hand posture determination and additional sensor for position of the hand. This sensor is usually an acoustic or a magnetic that attached to the glove. Look-up table software toolkit provided for some applications to recognize hand posture.

The second approach is "**Vision based Analysis**" that human beings get information from their surroundings, and this is probably most difficult approach to employ in satisfactory way. Many different implementations have been tested so far. One is to deploy 3-D model for the human hand. Several cameras attached to this model to determine parameters corresponding for matching images of the hand, palm orientation and joint angles to perform hand gesture classification. Lee and Kunii developed a hand gesture analysis system based on a threedimensional hand skeleton model with 27 degrees of freedom. They incorporated five major constraints based on the human hand kinematics to reduce the model parameter space search. To simplify the model matching, specially marked gloves were used.

The Third implementation is "**Analysis of drawing gesture**" use stylus as an input device. These drawing analysis lead to recognition of written text. Mechanical sensing work has used for hand gesture recognition at vast level for direct and virtual environment manipulation. Mechanically sensing hand posture has many problems like electromagnetic noise, reliability and accuracy. By visual sensing gesture interaction can be made potentially practical but it is most difficult problem for machines.

Full American Sign Language recognition systems (words, phrases) incorporate data gloves. *Takashi and Kishino* discuss a Data glove-based system that could recognize 34 of the 46 Japanese gestures (user dependent) using a joint angle and hand orientation coding technique. From their paper, it seems the test user made each of the 46 gestures 10 times to provide data for principle component and cluster analysis. The user created a separate test from five iterations of the alphabet, with each gesture well separated in time. While these systems are technically interesting, they suffer from a lack of training.

Excellent work has been done in support of machine sign language recognition by *Sperling and Parish*, who has done careful studies on the bandwidth necessary for a sign conversation using spatially and temporally sub-sampled images. Point light experiments (where "lights" are attached to significant locations on the body and just these points are used for recognition), have been carried out by *Poizner.* Most systems to date study isolate/static gestures. In most of the cases those are fingerspelling signs.

LIGHTING

The task of differentiating the skin pixels from those of the background is made considerably easier by a careful choice of lighting. According to Ray Lockton, if the lighting is constant across the view of the camera then the effects of selfshadowing can be reduced to a minimum. (See Figure 2.1)

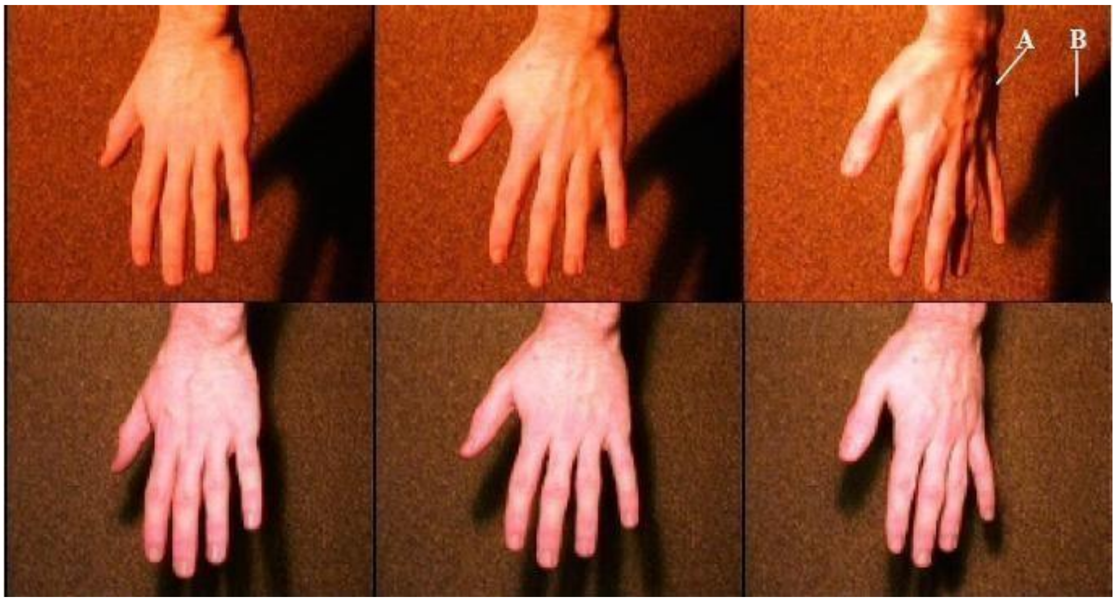


Figure 3.1: The effect of self-shadowing (A) and cast shadowing (B)

The top three images were lit by a single light source situated off to the left. A self- shadowing effect can be seen on all three, especially marked on the right image where the hand is angled away from the source. The bottom three images are more uniformly lit, with little self-shadowing. Cast shadows do not affect the skin for any of the images and therefore should not degrade detection. Note how an increase of illumination in the bottom three images results in a greater contrast between skin and background.

The intensity should also be set to provide sufficient light for the CCD in the

camera. However, since this system is intended to be used by the consumer
it

would be a disadvantage if special lighting equipment were required. It was decided to attempt to extract the hand information using standard room lighting. This would permit the system to be used in a non-specialist environment.

CAMERA ORIENTATIONS AND DISTANCE

It is very important to careful about direction of camera to permit easy choice of background. Two good and more effective approaches are to point the camera towards wall or floor. Lighting was standard room; intensity of light would be higher and shadowing effects lower because camera was pointed downwards. The distance of the camera from the hand should be such that it covers the entire gesture mainly. There is no effect found on the accuracy of the system if the image is a zoomed one or not; the principle is to cover the entire hand area majorly.

BACKGROUND SELECTION

Another important aspect is to maximize differentiation that the colour of background must be different as possible from skin colour. The floor colour in the work used was black. It was decided to use this colour because it offered minimum self-shadowing problem as compared to other background colours.

VISION-BASED GESTURE RECOGNITION

The most significant disadvantage of the tracker-based systems is that they are cumbersome. This detracts from the immerse nature of a virtual environment by requiring the user to put on an unnatural device that cannot easily be ignored, and which often requires significant effort to put on and calibrate. Even optical systems with markers applied to the body suffer from these shortcomings, albeit not as severely. What many have wished for is a technology that provides realtime data useful for analysing and recognizing human motion that is passive and non-obtrusive. Computer vision techniques have the potential to meet these requirements.

Vision-based interfaces use one or more cameras to capture images, at a frame rate of 30 Hz or more, and interpret those images to produce visual features that can be used to interpret human activity and recognize gestures.

Typically, the camera locations are fixed in the environment, although they may also be mounted on moving platforms or on other people. For the past decade, there has been a significant amount of research in the computer vision community on detecting and recognizing faces, analysing facial expression, extracting lip and facial motion to aid speech recognition, interpreting human activity, and recognizing particular gestures.

Unlike sensors worn on the body, vision approaches to body tracking have to contend with occlusions. From the point of view of a given camera, there are always parts of the user’s body that are occluded and therefore not visible – e.g., the backside of the user is not visible when the camera is in front. More significantly, self-occlusion often prevents a full view of the fingers, hands, arms, and body from a single view. Multiple cameras can be used, but this adds correspondence and integration problems.

The occlusion problem makes full body tracking difficult, if not impossible, without a strong model of body kinematics and perhaps dynamics. However, recovering all the parameters of body motion may not be a prerequisite for gesture recognition. The fact that people can recognize gestures leads to three possible conclusions: we infer the parameters that we cannot directly observe, we don’t need these parameters to accomplish the task, and we infer some and ignore others.

Unlike special devices, which measure human position and motion, vision uses a multipurpose sensor; the same device used to recognize gestures can be used to recognize other objects in the environment and also to transmit video for teleconferencing, surveillance, and other purposes. There is a growing interest in CMOS-based cameras, which promise miniaturized, low cost, low power cameras integrated with processing circuitry on a single chip.

Currently, most computer vision systems use cameras for recognition. Analog cameras feed their signal into a digitizer board, or frame grabber, which may do a DMA transfer directly to host memory. Digital cameras bypass the analog-todigital conversion and go straight to memory. There may be a pre-processing step, where images are normalized, enhanced, or transformed in some manner, and then a feature extraction step. The features
– which may be any of a variety of two- or three-dimensional features, statistical properties, or estimated body parameters – are analysed and classified as a particular gesture if appropriate.

This technique was also used by us for recognizing hand gestures in real time. With the help of a web camera, I took pictures of hand on a prescribed background and then applied the classification algorithm for recognition.

CHAPTER 4

METHODOLGY

INTRODUCTION

There have been numerous researches in this field and several methodologies were proposed like Principle Component Analysis (PCA) method, gradient method, subtraction method etc. PCA relates to Linear transformation consist on statistical approach. This gives us powerful tool for pattern recognition and data analysis which mostly used in image processing techniques for data (compression, dimension and correlation). Gradient method is also another image processing technique that detect colour patches applying low pass filters is also known as edge detection method. Subtraction method is very simple that subtract input image pixel to another image or constant value to provide output. I have also studied different approaches to hand gesture recognition and came to know that implementation of such techniques like PCA and Gradient method is complicated, we can produce same output as these techniques gives us by simple and easy implementation. So, I have tried four different algorithms and finally selected the one, which was most efficient i.e. diagonal sum algorithm. This algorithm is able to recognize maximum gestures correctly.

PROJECT CONSTRAINTS

I propose a vision-based approach to accomplish the task of hand gesture detection. As discussed above, the task of hand gesture recognition with any machine learning technique suffers from the variability problem. To reduce the variability in hand recognition task we assume the following assumptions:

- Single colored camera mounted above a neutral colored desk.
- User will interact by gesturing in the view of the camera.
- Training is must.
- Hand will not be rotated while image is capturing.

The real time gesture classification system depends on the hardware and software.

THE WEBCAM SYSTEM (USB PORT)

Below is the summary of the specifications of the camera which this system required:

Resolution:	640x480
Video frame rate:	30fps @640x480
Pixel depth:	Minimum 1.3-mega pixels
Connection port:	USB

In my project web cam was attached via USB port of the computer. The web cam worked by continually capturing the frames. In order to capture a particular frame, the user just need to select the particular Algorithm METHOD button on the interface and the hand was detected in the particular frame. The web cam took color pictures, which were then converted into grayscale format. The main reason of sticking to grayscale was the extra amount of processing required to deal with color images.

BRIEF OUTLINE OF THE IMPLEMENTED SYSTEM

Hand gesture recognition system can be divided into following modules:

Pre-processing
Feature extraction of the processed image
Real Time classification

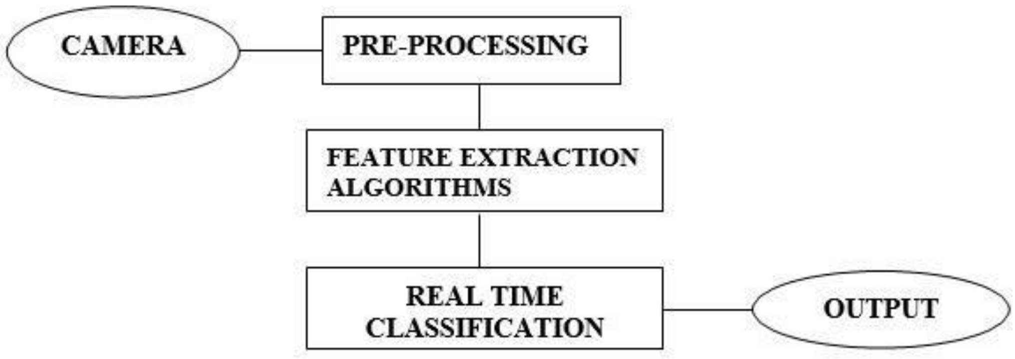


Figure 4.1: System Implementation

PRE-PROCESSING

Like many other pattern recognition tasks, pre-processing is necessary for enhancing robustness and recognition accuracy.

The pre-processing prepares the image sequence for the recognition, so before calculating the diagonal Sum and other algorithms, a pre-processing step is performed to get the appropriate image, which is required for real time classification. So, it consists of some steps. The net effect of this processing is to extract the hand only from the given input because once the hand is detected from the given input it can be recognized easily. So, pre-processing step mainly consists of following tasks:

- **Skin Modeling**
- **Removal of Background**
- **Conversion from BGR to binary**
- **Hand Detection**

Skin Modelling

There are numerous methods used for skin detection such as RGB (Red, Green, Blue), YCbCr (Luminance Chrominance) and HSV (Hue, Saturation, Value).

- ### BGR:

BGR is a 3D color space pixel where each pixel has combination of three colors Red, Green and Blue at specific location. This technique widely used in image processing for identifying skin region.

- ### YCbCr (Luminance Chrominance):

This color space is used in digital video color information represent two color Cb and Cr. Cb is difference between Blue and Cr is difference between Red component references of value. This is basically RGB transformation to YCbCr for separation of luminance and chrominance for color modelling.

- ### HSV (Hue, Saturation and Value):

In HSV, Hue detect dominant color and Saturation define colourfulness whilst Value measure intensity or brightness. This is well enough to choose single color but it ignores complexity of color appearance. It trade off computation speed mean computationally expensive and perceptual relevance.

Removal of Background

I have found that background greatly affects the results of hand detection that's why I have decided to remove it. For this I have written our own code in spite of using any built-in ones.

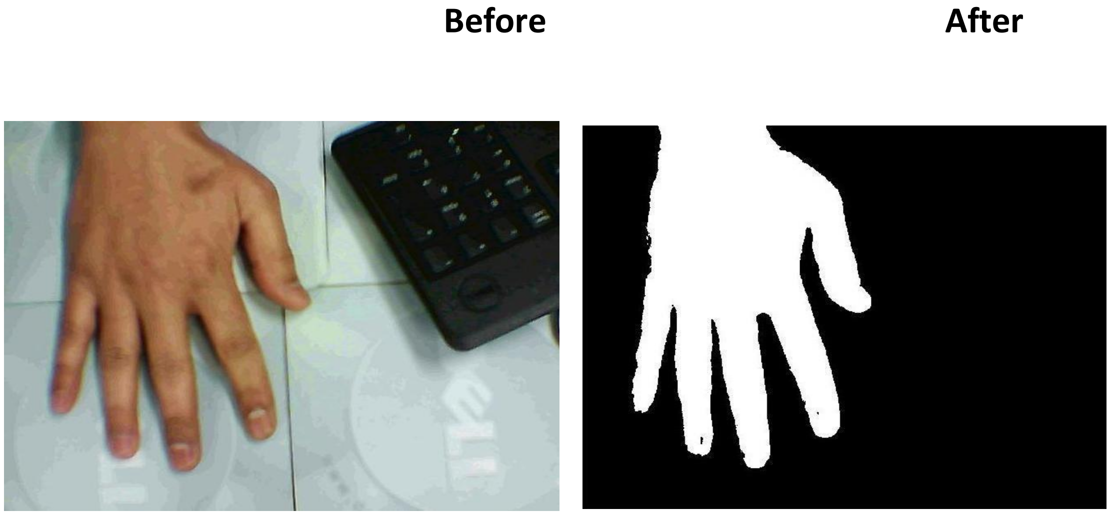


Figure: 4.2: Removal of Background

Conversion from BGR to Binary

All algorithms accept an input in RGB form and then convert it into binary format in order to provide ease in recognizing any gesture and also retaining the luminance factor in an image.

FEATURE EXTRACTION ALGORITHMS

There are four types of algorithms that I studied and implemented namely as followings:

- **Row vector algorithm**
- **Edging and row vector passing**
- **Diagonal sum algorithm**

CHAPTER 5

FEATURE EXTRACTIONS

INTRODUCTION

In this chapter, I described the detail of all the four features extraction algorithms. First, I would like to discuss neural network used in first three algorithms.

NEURAL NETWORKS

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Figure 4.1 below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this supervised learning to train a network.

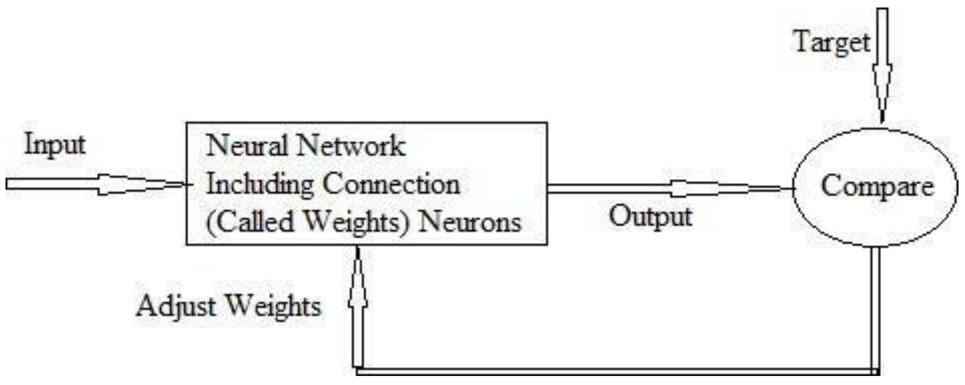
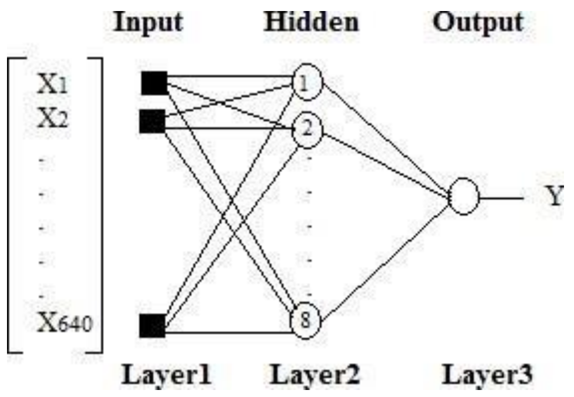


Figure 5.1 Neural network Block Diagram

Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, and vision and control systems. Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings.

Once data ready for representation then next step is to design NN for training and testing data. In first two algorithms Row Vector and Edging and Row Vector passing algorithm have three layers feed forward network: Input, Hidden and Output. Number of neurons in Input is 640 which are equal to number of features extracted from each of algorithm and one neuron for Output layer for skin class to be recognized. But for Mean and standard deviation there are only two input which is also equal to extracted features from this algorithm. Neural network Architecture has number of parameters such as learning rate (lr), number of epochs and stopping criteria which is based on validation of data. Training of Mean Square Error at output layer



which is set trial values and which is set by several experiments.

Figure 5.2: NN for Row Vector and Edging Row Vector

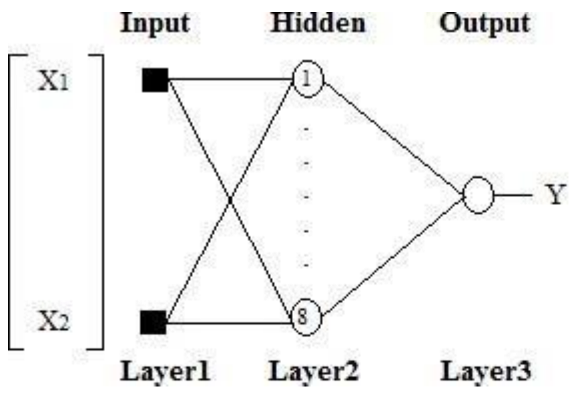


Figure 5.3: NN for Mean and S.D

ROW VECTOR ALGORITHM

We know that behind every image is a matrix of numbers with which we do manipulations to derive some conclusion in computer vision. For example, we can calculate a row vector of the matrix. A row vector is basically a single row of numbers with resolution 1*Y, where Y is the total no ofcolumns in the image matrix. Each element in the row vector represents the sum of its respective column entries as illustrated in **Figure 5.4:**

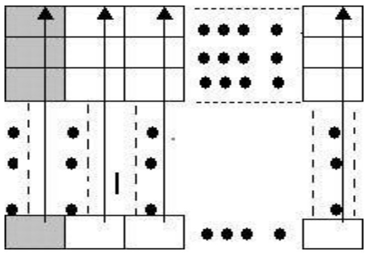


Figure 5.4 Row vector of an image

The first algorithm I studied and implemented makes use of the row vector of the hand gestures. For each type of hand gesture, I took several hand images, do skin modelling, labelling, removed their background and RGB to binary conversion in the pre-processing phase, calculated their row vectors and then trained the neural network with these row vectors. Ultimately, the neural network was able to recognize the row vectors that each gesture count can possibly have. Hence, after training, the system was tested to see the recognition power it had achieved.

Mathematically, we can describe the image for training or testing purpose given to the neural network as:

Input to neural network =Row vector (After image Preprocessing)

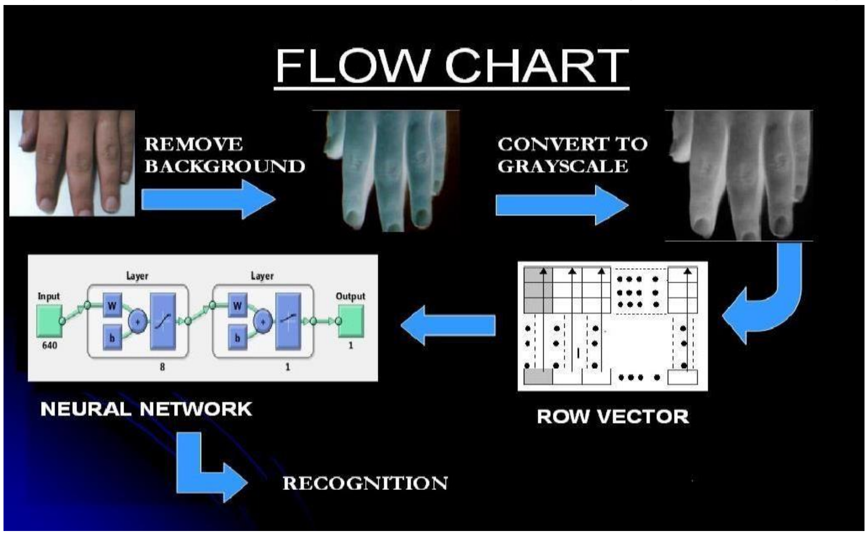


Figure 5.5: Row Vector Flow Chart

EDGING AND ROW VECTOR PASSING ALGORITHM

In the pre-processing phase of this algorithm, I do pre-processing, skin modelling and removed the background etc. of the gesture image taken. This image was converted from RGB into grayscale type. grey scale images represent an image as a matrix where every element has a value corresponding to how bright/dark the pixel at the corresponding position should be colored.

For representing the brightness of pixels there are two ways for represent numbers, first class called Double class that assign floating numbers (“decimals”) between 0 and 1 for each pixel. The zero (0) value represent black and value one
(1) corresponds to white. The second class known as unit8 that assign integer between 0 and 255 for brightness of pixel, zero (0) correspond to black and 255 for white. The unit8 class requires less storage than double roughly 1/8.

After the conversion of the image into grayscale, I took the edge of the image with a fixed threshold i.e. 0.5. This threshold helped us in removing the noise in the image. In the next step, a row vector of the edged image was calculated. This row vector is then passed on to the neural network for training. The neural network (NN) is later on tested for the classification of the gestures.

Mathematically, the input to the neural network is given as:

Input to NN= Row vector [Edge (Grayscale image)]

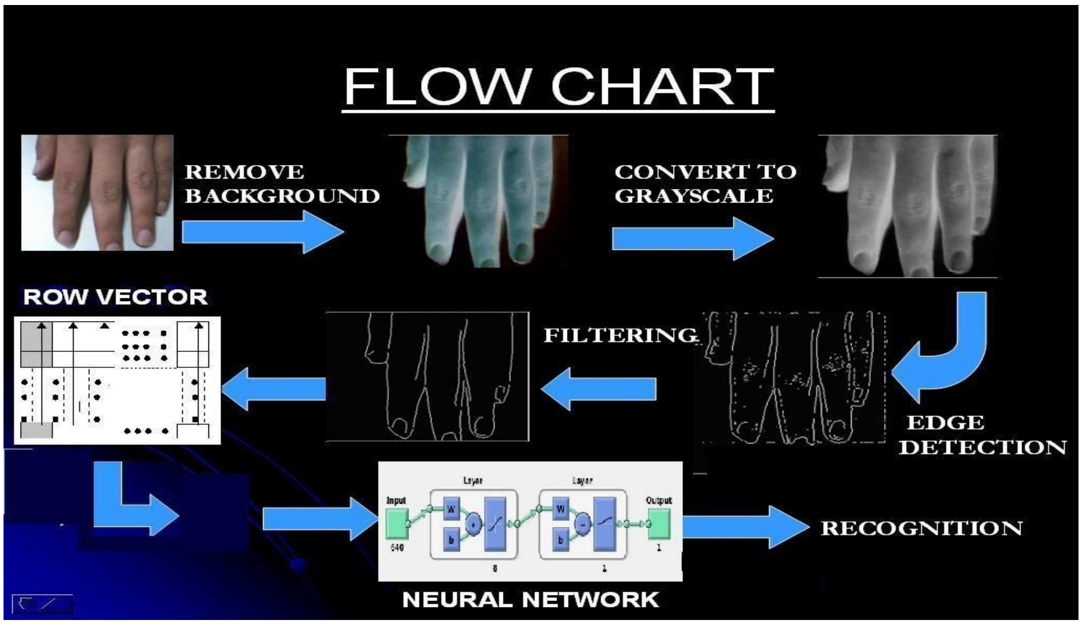
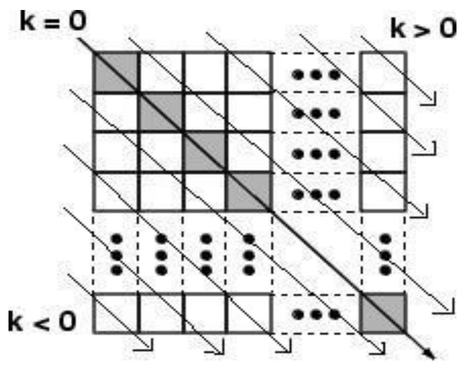


Figure 5.6: Edging and Row Vector Flow Chart

DIAGONAL SUM ALGORITHM



In the pre-processing phase, doing mentioned steps in methodology, skin modeling removal of the background, conversion of RGB to binary and labeling. The binary image format also stores an image as a matrix but can only color a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white. In the next step, the sum of all the elements in every diagonal is calculated. The main diagonal is represented as k=0 in Figure 4.7 given below; the diagonals below the main diagonal are represented by k<0 and those above it **Figure: 5.7: Diagonal Sum** are represented as k>0

The gesture recognition system developed through this algorithm first train itself with the diagonals sum of each type of gesture count at least once, and then its power could be tested by providing it with a test gesture at real time. Mathematically, the input given to the system at real time is given as:

$$X_i \quad \square_{i\,\boxed{1}} \quad \textit{Diagonals}$$

$$\text{Input} = \square_{i-1} \; X_i$$

The flowchart of the algorithm is given below in **Figure 4.9:**

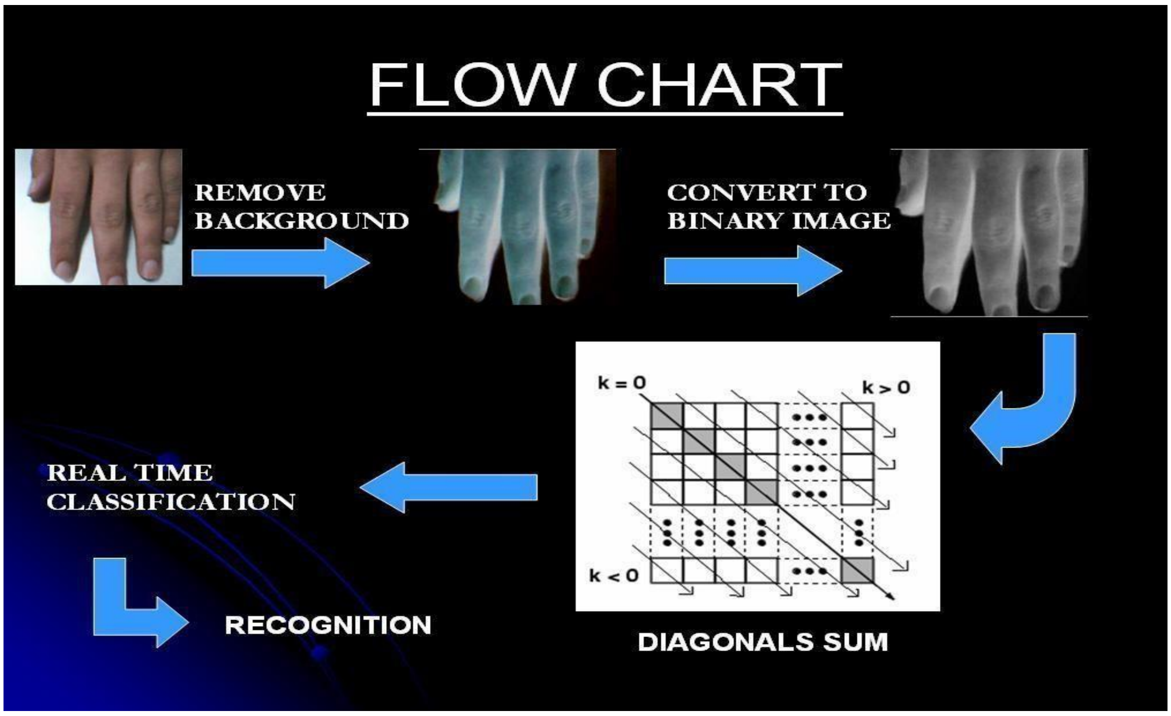


Figure 5.8: Diagonal Sum Flow Chart

Chapter 6 Coding

Counting Fingers

```
 import cv2
import numpy as
np import
```

pyautogui import
time

```
cap=cv2.VideoCapture(0)

while(cap.isOpened()):
time.sleep(.00001)
_,feed=cap.read()
        cv2.rectangle(feed,(50,100),(300,400),(0,0,255),0)
image=feed[100:400,50:300]
            img=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        blur=cv2.GaussianBlur(img,(35,35),0)           ret,thresh =
              cv2.threshold(blur,0,255,1+cv2.THRESH_OTSU)
                                                                contours,
hierarchy = cv2.findContours(thresh,1,1)         max_area=0
pos=0         for i in contours:                 area=cv2.contourArea(i)
if area>max_area:                        max_area=area
                        pos=i
        peri=cv2.arcLength(pos,True)
approx=cv2.approxPolyDP(pos,0.02*peri,True)
        hull=cv2.convexHull(pos)
            #print len(hull)
    #cv2.polylines(image,[approx],True,(0,255,255))
    #cv2.drawContours(image,[approx],-1,(255,100,50),2)
cv2.drawContours(image,[hull],-1,(0,255,0),2)          hull =
cv2.convexHull(pos,returnPoints = False)          defects =
cv2.convexityDefects(pos,hull)          num=0
l=defects.shape[0]         for i in range(1,defects.shape[0]):
                s,e,f,d = defects[i,0]
far = tuple(pos[f][0])
if d>10000:
num+=1
                        cv2.circle(image,far,3,[0,0,255],-1)
        num+=1;
if num==1:
s='One'         elif
num==2:
    s='Two'
elif num==3:
    s='Three'
 elif num==4:
    s='Four'
 elif num==5:
    s='Five'
else:
```

```
                s='HELLO WOLRD'
feed[100:400,50:300]=image          font =
cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(feed,s,(100,450),        font,        2,(255,10,10),2,cv2.LINE_AA)
cv2.imshow('Feed',feed)
        cv2.imshow('image',thresh)

        k=cv2.waitKey(10)
if k==27:
break

        # Close the camera if 'q' is
pressed if cv2.waitKey(1) == ord('q'):
break

cap.release() cv2.destroyAllWindows()
```

Scrolling/ Volume UP/DOWN

```
import cv2
import numpy as
np import
pyautogui import
time

cap=cv2.VideoCapture(0)

while(cap.isOpened()):
time.sleep(.01)
_,feed=cap.read()
        cv2.rectangle(feed,(50,100),(300,400),(0,255,0),0)
image=feed[100:400,50:300]
```

```
img=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
blur=cv2.GaussianBlur(img,(35,35),0)            ret,thresh =
cv2.threshold(blur,0,255,1+cv2.THRESH_OTSU)
                                                contours,
hierarchy = cv2.findContours(thresh,1,1)         max_area=0
pos=0         for i in contours:                 area=cv2.contourArea(i)
if area>max_area:                        max_area=area
                        pos=i
        peri=cv2.arcLength(pos,True)
approx=cv2.approxPolyDP(pos,0.02*peri,True)
        hull=cv2.convexHull(pos)
            #print len(hull)
    #cv2.polylines(image,[approx],True,(0,255,255))
    #cv2.drawContours(image,[approx],-1,(255,100,50),2)
cv2.drawContours(image,[hull],-1,(0,0,255),2)         hull =
cv2.convexHull(pos,returnPoints = False)         defects =
cv2.convexityDefects(pos,hull)         num=0
l=defects.shape[0]         for i in range(1,defects.shape[0]):
                s,e,f,d = defects[i,0]
far = tuple(pos[f][0])
if d>10000:
num+=1
cv2.circle(image,far,3,[0,0,255
],-1)
        num+=1;
if num==1:
                s=''         elif
num==2:
    pyautogui.press('down')
    s='Down'         elif num==3:
    pyautogui.press('up'
    ) s='Up'       else:
            s='HELLO WOLRD'
        feed[100:400,50:300]=image           font =
cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(feed,s,(100,450),       font,       2,(255,10,10),2,cv2.LINE_AA)
cv2.imshow('Feed',feed)
        cv2.imshow('image',thresh)

        k=cv2.waitKey(10)
if k==27:
break

        # Close the camera if 'q' is
pressed if cv2.waitKey(1) == ord('q'):
```

break

```
cap.release() cv2.destroyAllWindows()
```

Playing Game

```
import cv2
import numpy as
np import
pyautogui import
time

cap=cv2.VideoCapture(0)

while(cap.isOpened()):
time.sleep(.0000001)
                                    _,feed=cap.read(
)
       cv2.rectangle(feed,(50,100),(300,400),(0,0,255),0)
image=feed[100:400,50:300]
          img=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
      blur=cv2.GaussianBlur(img,(35,35),0)          ret,thresh =
            cv2.threshold(blur,0,255,1+cv2.THRESH_OTSU)
                                                                   contours,
hierarchy = cv2.findContours(thresh,1,1)         max_area=0
pos=0         for i in contours:                area=cv2.contourArea(i)
if area>max_area:                        max_area=area
pos=i
       peri=cv2.arcLength(pos,True)
approx=cv2.approxPolyDP(pos,0.02*peri,True)
       hull=cv2.convexHull(pos)
          #print len(hull)
   #cv2.polylines(image,[approx],True,(0,255,255))
   #cv2.drawContours(image,[approx],-1,(255,100,50),2)
cv2.drawContours(image,[hull],-1,(0,255,0),2)         hull =
cv2.convexHull(pos,returnPoints = False)         defects =
cv2.convexityDefects(pos,hull)         num=0
l=defects.shape[0]         for i in range(1,defects.shape[0]):
             s,e,f,d = defects[i,0]
far = tuple(pos[f][0])
if d>10000:
```

```
num+=1
                        cv2.circle(image,far,3,[0,0,255],-1)
        num+=1;
if num==1:
                s=''         elif
num==2:
    pyautogui.press('up')
    s='Two'          elif num==4:
    pyautogui.press('space')
                s='Start'
else:
            s='HELLO WOLRD'
        feed[100:400,50:300]=image           font =
cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(feed,s,(100,450),        font,        2,(255,10,10),2,cv2.LINE_AA)
cv2.imshow('Feed',feed)
        cv2.imshow('image',thresh)

        k=cv2.waitKey(10)
if k==27:
break

        # Close the camera if 'q' is
pressed if cv2.waitKey(1) == ord('q'):
break

cap.release()
cv2.destroyAllWindows()
```

Front Page GUI

```
package Gui; import java.io.*;
import java.util.logging.Level;
import
java.util.logging.Logger;
import static javafx.application.Platform.exit;
/**
 *
* @author AFFAN AHMAD
 */
public class Gesture extends javax.swing.JFrame {

    /**
* Creates new form Gesture
     */
    public Gesture()
```

```
{          initComponents();
setLocationRelativeTo(null);
```

```
    }

    /**
* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
   * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    //     <editor-fold     defaultstate="collapsed"     desc="Generated     Code">
private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jPanel3 = new javax.swing.JPanel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jButton5 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBackground(new java.awt.Color(153, 153, 255));

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel1.setText("Hand Gesture Recognition");

        jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ExtraFiles/gnit.jpg"))); // NOI18N

        jLabel3.setBackground(new java.awt.Color(204, 204, 204));
jLabel3.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setText("GREATER NOIDA INSTITUTE OF TECHNOLOGY");

        javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
```

```
jPanel3Layout.setHorizontalGroup(
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(jPanel3Layout.createSequentialGroup()
             .addComponent(jLabel2) .addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED)                  .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
             .addContainerGap())
      );
      jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
             .addComponent(jLabel2)
             .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
149, javax.swing.GroupLayout.PREFERRED_SIZE))
      );

      jButton1.setBackground(new java.awt.Color(153, 153, 0));
jButton1.setFont(new java.awt.Font("Trebuchet MS", 1, 18)); // NOI18N
jButton1.setText("COUNT FINGERS"); jButton1.addActionListener(new
java.awt.event.ActionListener()
{            public void actionPerformed(java.awt.event.ActionEvent evt)
{               jButton1ActionPerformed(evt);
          }
       });

      jButton2.setBackground(new java.awt.Color(0, 0, 204));
      jButton2.setFont(new java.awt.Font("Trebuchet MS", 1, 18)); // NOI18N
jButton2.setForeground(new java.awt.Color(255, 255, 255));
jButton2.setText("PLAY GAME");            jButton2.addActionListener(new
java.awt.event.ActionListener() {             public void
actionPerformed(java.awt.event.ActionEvent evt)
{               jButton2ActionPerformed(evt);
          }
       });

      jButton3.setBackground(new java.awt.Color(204, 0, 204));
jButton3.setFont(new java.awt.Font("Trebuchet MS", 1, 18)); // NOI18N
jButton3.setText("SCROLLING");            jButton3.addActionListener(new
java.awt.event.ActionListener() {               public void
```

```
actionPerformed(java.awt.event.ActionEvent evt)
{                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setBackground(new java.awt.Color(0, 153, 0));
        jButton4.setFont(new java.awt.Font("Trebuchet MS", 1, 18)); // NOI18N
jButton4.setText("Play/Pause");        jButton4.addActionListener(new
java.awt.event.ActionListener() {            public void
actionPerformed(java.awt.event.ActionEvent evt)
{                jButton4ActionPerformed(evt);
            }
        });

        jButton5.setBackground(new java.awt.Color(255, 0, 0));
        jButton5.setFont(new java.awt.Font("Trebuchet MS", 1, 18)); // NOI18N
jButton5.setText("EXIT");          jButton5.addActionListener(new
java.awt.event.ActionListener() {            public void
actionPerformed(java.awt.event.ActionEvent evt)
{                jButton5ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)                .addGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, jPanel1Layout.createSequentialGroup()
                .addContainerGap(161, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                    .addComponent(jButton5,
javax.swing.GroupLayout.PREFERRED_SIZE,                                        119,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(109, 109, 109))
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()                    .addComponent(j
Button1,
javax.swing.GroupLayout.PREFERRED_SIZE,                                        199,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(jButton2)
                    .addGap(18, 18, 18)
                    .addComponent(jButton3)
                    .addGap(18, 18, 18)
                    .addComponent(jButton4)
                    .addGap(148, 148, 148))))
        );

        jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {jButton1, jButton2, jButton3, jButton4});

        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(126, 126, 126)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B ASELINE)
                    .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 79,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2)
                    .addComponent(jButton3)
                    .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,                148,
Short.MAX_VALUE)
            .addComponent(jButton5,       javax.swing.GroupLayout.PREFERRED_SIZE,             52,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(76, 76, 76))
      );

      jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL,            new
java.awt.Component[] {jButton1, jButton2, jButton3, jButton4});

      javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(           layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)              .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
      );

layout.setVerticalGroup(           layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)              .addComponent(jPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
      );

      pack();
   }// </editor-fold>

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      //Play     game
try {
         Process p=Runtime.getRuntime().exec("C:\\Python27\\python.exe
Files\\game.py");
      } catch (IOException ex) {
         ex.printStackTrace();
      }
   }

   private    void    jButton3ActionPerformed(java.awt.event.ActionEvent    evt)
{                                      //Scrolling        try {
        Process
p=Runtime.getRuntime().exec("C:\\Python27\\python.exe
Files\\scrolling.py");
```

```
        }              catch              (IOException              ex)
{              ex.printStackTrace();
        }
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        //Play/Pause
try {
          Process          p=Runtime.getRuntime().exec("C:\\Python27\\python.exe
Files\\pause.py");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    private    void    jButton1ActionPerformed(java.awt.event.ActionEvent    evt)
{                                        try {
            //Count Fingers
            Process
                              p=Runtime.getRuntime().exec("C:\\Python27\\python.ex
e Files\\count.py");
        }              catch              (IOException              ex)
{              ex.printStackTrace();
        }
    }

    private    void    jButton5ActionPerformed(java.awt.event.ActionEvent    evt)
{                                        System.exit(0);
    }

    /**
* @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
* For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.h
```

```
tml
        */                          try   {                                    for
(javax.swing.UIManager.LookAndFeelInfo info :
```

```
javax.swing.UIManager.getInstalledLookAndFeels())
{                 if ("Nimbus".equals(info.getName())) {
                   javax.swing.UIManager.setLookAndFeel(info.getClassName());
break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Gesture.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Gesture.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Gesture.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Gesture.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new              Runnable()
{             public void run() {
                new Gesture().setVisible(true);
            }
         });
    }

    // Variables declaration - do not
    modify private javax.swing.JButton
    jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;       private
```

```
javax.swing.JLabel jLabel2;       private
javax.swing.JLabel jLabel3;       private
javax.swing.JPanel jPanel1;       private
javax.swing.JPanel jPanel3;      // End of
variables declaration
```

FUTURE SCOPE

The Hand Gesture recognition is moving at tremendous speed for the futuristic products and services and major companies are developing technology based on the hand gesture system and that includes companies like Microsoft, Samsung, Sony and it includes the devices like Laptop, Hand held devices, Professional and LED lights. The verticals include where the Gesture technology is and will be evident are Entertainment, Artificial Intelligence, Education and Medical and Automation fields. And with lot of Research and Development in the field of Gesture Recognition Field, the use and adoption will become more cost effective and cheaper. It’s a brilliant feature turning data into features with mix of technology and Human wave. Smart phones have been experiencing enormous amount of Gesture Recognition Technology with look and views and working to manage the Smartphone in reading, viewing and that includes what we call touch less gestures. Google Glass has been also in the same cadre. And the Technology has also been embedded into smart televisions nowadays as well, which can easily control and managed by Voice and Hand options. In the medical fields Hand Gesture may also be experienced in terms of Robotic Nurse and medical assistance. As the Technology is always revolving and changing the future is quite unpredictable but we have to be certain the future of Gesture Recognition is here to stay with more and eventful and Life touching experiences.

CONCLUSION

The importance of gesture recognition lies in building efficient human machine interaction. Its applications range from sign language recognition through medical rehabilitation to virtual reality. Gesture recognition soft computing tools pose another promising application to static hand gestures identification. Thus, gesture recognition promises wide-range applications in fields from photojournalism through medical technology to biometrics.

REFERENCES

- https://en.wikipedia.org
- https://www.tutorialspoint.com
- https://opencv.org
- https://www.w3schools.com
- https://stackoverflow.com
- https://www.slideshare.net