JSS MAHAVIDYAPEETHA

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

(AUTONOMOUS), MYSURU-570006

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

Project Report on

GESTURE RECOGNITION FOR COMPUTER APPLICATIONS AND PHOTOGRAPHY

Semester: VI

Section: A

Submitted by:

Rohan Kuntoji	42	4JC14EC082
S Ajith Kumar	44	4JC14EC085
Sneha C Morge	49	4JC14EC107

Under the Guidance of

Dr. T SHREEKANTH

and

CHANDRASHEKAR MURTHY B N

ACKNOWLEDGEMENT

We would like to take this opportunity to express our gratitude and sincere thanks to our respected supervisors Prof. T. SHREEKANTH and CHANDRASHEKAR MURTHY. B .N for their continuous guidance, insight, and the mental support they have provided us throughout the course of this work. We would like to articulate our profound gratitude and indebtedness to Prof. S. P. KULKARNI and Prof. A. THYAGARAJMURTHY for providing us with such an exquisite opportunity of developing this mini-project with the availability of the practical labs at any time, for our help that enabled us to achieve this success. We would like to thank all the faculty members and the staff of the Department of Electronics and Communication Engineering, JSS Science and Technology University for their extreme help throughout course. An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in the reference section. We acknowledge our indebtedness to all of them. Last but not the least our sincere thanks to all our friends, who have patiently extended all sorts of help for accomplishing this undertaking.

Date: 26th April 2017

Sri Jayachamarajendra College of Engineering

Rohan Kuntoji (4JC14EC082)
S Ajith Kumar (4JC14EC085)
Sneha C Morge (4JC14EC107)

Contents

List of Figures

Abstract

Gesture recognition is the fast growing field in image processing and artificial technology. The gesture recognition is a process in which the gestures or postures of human body parts are identified and are used to control computers and other electronic appliances. The most contributing reason for the emerging gesture recognition is that they can create a simple communication path between human and computer called HCI (Human Computer Interaction).[2] This project is confined to identification of hand postures and to establish a man-machine interaction.

The main objective of this project is to understand and demonstrate how gesture recognition can be used in various aspects and applications. We also, at the same time, develop knowledge on how this technology could be utilized in various areas in the coming years, as an alternative over the existing methods.

Chapter 1

Introduction

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 and image processing technique to interpret sign language. However, the identification and recognition of posture, gait, proxemics, and human behaviors is also the subject of gesture recognition techniques.[1] 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.

In present days a lot of research has been taking place to develop a simple and natural interaction between human and computer. The ways in which one can interact with computer are either by using devices like keyboard, mouse or via audio signals, while the former always needs a physical contact and the latter is prone to noise and disturbances, with the advancement in image processing a simple and efficient mode of communication with computer can be developed. Gesture recognition enables humans to communicate with the machine (HMI) and interact naturally without any mechanical devices. Using the concept of gesture recognition, it is possible to point a finger at the computer screen so that the cursor will move accordingly. This could make conventional input devices such as mouse, keyboards and even touch-screens redundant.

It is hard to clinch on a useful definition of gestures because of its wide variety of applications. A statement concerning it can only specify a particular domain of gestures. Many researchers had tried to define gestures but their actual meaning is still uncertain. Bobick and Wilson have defined gestures as the motion of the body intended to communicate with other agents. As per the context of the project, a gesture is defined as an expressive movement of hand which has a particular message that is communicated precisely between a sender and a receiver. [9] [10] A sender and a receiver should have the same set of information for a particular hand gesture for a successful communication. A gesture can be categorized as dynamic and static. A dynamic gesture is intended to change over a 3 period of time whereas a static gesture is observed at the spurt of time. A waving hand meaning goodbye is an example of dynamic gesture and a still hand sign is an example of static gesture. All the static and dynamic gestures are interpreted over a period of time to understand a full message. [16] This complicated process is termed as gesture recognition. Gesture recognition deals with a process of recognition and interpretation of a stream of continuous sequential gesture from the given set of input data.

Master Hand Technology uses different hand gestures and colors to give various commands for the Human-Machine (here Computer) Interfacing. Using hand to create gestures and color codes attached to it gives additional flexibility in usage and applications.

Using the hand gestures and hand movement, many applications can be developed. It can also be interfaced with computer and can work as a mouse. Similarly a no. of mouse interfacing applications can be developed. Many virtual applications can be developed that creates a virtual reality. This works only when the application runs. All the computer applications can be interfaced with hand gesture control and can be controlled directly from hand gesture.

Mouse-Interfacing: It's an example of dynamic gestures, where hand movement (essentially finger in this case) is tracked. Mouse pointer is interfaced to this movement and mouse clicks are initiated through static gestures.

Media-Control: This hand gesture recognition technique can be aptly used for easy functioning and controlling of various media devices, without any physical contact with them respectively. This enhances the feasibility of the devices and thereby also provides easy and smooth handling. We could easily control functions such as pause, volume up and volume down with just mere hand gestures.

Easy Power-point Presentation: With the help of these hand gestures, we could easily monitor the basic functioning of power-point presentation such as slide-show, scrolling and switching over to other slides easily with these basic hand gestures. Hence we can control it.

Quick Photography: With the use of a good quality camera or a web-camera, we could take pictures and good snapshots with just the shake of our hands, within no time, or posing some gestures, as designed so that image will be captured by the camera. [2]

Chapter 2

Background

Hand gesture recognition making use of digital images has been a research topic for many years now. Direct use of hands as an input device is an innovative method for providing natural Human Computer Interaction which has its inheritance from text-based interfaces through 2D graphical-based interfaces, multimedia-supported interfaces, to full-fledged multi-participant Virtual Environment (VE) systems [2]. According to P Premaratne the history of hand gesture recognition for computer control started with the invention of glove-based control interfaces. Researchers realized that gestures inspired by sign language can be used to offer simple commands for a computer interface. This gradually evolved with the development of much accurate accelerometers, infrared cameras and even fiber optic bend-sensors. Some of those developments in glove based systems eventually offered the ability to realize computer vision based recognition without any sensors attached to the glove.

2.1 Color Space:

By using primary colors of pigment (cyan, magenta, yellow, and black), a wide range of colors are created. Those colors then define a specific color space. To create a three dimensional representation of a color space, the amount of magenta color is assigned to X axis, the amount of cyan to Y axis, and the amount of yellow to Z axis. This resulting 3-D space can provide a unique position for every possible color being created by combining those three pigments. On the contrary, this is not the only possible color space. For example, when colors are displayed on a computer monitor, they are usually in the RGB (red, green and blue) color space. [3] This is another way of making nearly the same colors (limited by the reproduction medium, such as the phosphor (CRT) or filters and backlight (LCD)), and in this case red, green and blue can be considered as the X, Y and Z axes respectively. However, another way of making the same colors is to use their Hue, Saturation and their brightness Value as X axis, Y axis and Z axis respectively. Such a kind of 3D space is called the HSV color space. Many color spaces are represented as three-dimensional (X, Y, Z) values in this manner, but some may have more, or fewer dimensions, and some, like Pantone, cannot be represented in this way at all. [5]

2.2 Generic Color Models:

2.2.1 RGB

RGB uses additive color mixing, because it describes what kind of light needs to be emitted to produce a given color. Light is added together to create form from out of the darkness. RGB stores individual values for red, green and blue. RGBA is RGB with an additional channel, alpha, to indicate transparency. [5]

2.2.2 CMYK

CMYK uses subtractive color mixing used in the printing process, because it describes what kind of inks need to be applied so the light reflected from the substrate and through the inks produces a given color.[5]

2.2.3 YIQ

YIQ was formerly used in NTSC (North America, Japan and elsewhere) television broadcasts for historical reasons. This system stores a luminance value with two chrominance values, corresponding approximately to the amounts of blue and red in the color. It is similar to the YUV scheme used in most video capture systems and in PAL (Australia, Europe, except France, which uses SECAM) television, except that the YIQ color space is rotated 33 deg with respect to the YUV color space. The YDbDr scheme used by SECAM television is rotated in another way. [5]

2.2.4 YPbPr

YPbPr is a scaled version of YUV. It is most commonly seen in its digital form, YCbCr, used widely in video and image compression schemes such as MPEG and JPEG. [5]

2.2.5 HSV

HSV (hue, saturation, value), also known as HSB (hue, saturation, brightness) is often used by artists because it is often more natural to think about a color in terms of hue and saturation than in terms of additive or subtractive color components. HSV is a transformation of an RGB color space, and its components and colorimetry are relative to the RGB color space from which it was derived. [5]

2.2.6 HSL

HSL (hue, saturation, lightness/luminance), also known as HLS or HSI (hue, saturation, intensity) is quite similar to HSV, with "lightness" replacing "brightness". The difference is that the brightness of a pure color is equal to the brightness of white, while the lightness of a pure color is equal to the lightness of a medium gray. [15]

2.3 Components:

2.3.1 Camera:

A portable camera interfaced to a device or an inbuilt camera in any device, capable of sending data, is used to capture the image of a hand gesture and transmit via networking devices, hard wired or wireless, to main control computer, where the image is processed and necessary task is executed.

2.3.2 Control Computer:

A computer device that will receive the data and process using MATLAB and generate control signals to carry out necessary tasks as programmed.

2.3.3 Color Tapes:

Red, Green and Yellow color tapes or color markers are used respectively in this project for color mapping and recognition part, so that various gestures can be fed through this color code recognition. These specific colors are used because they form the integral colors of the entire color system.

2.3.4 Software Requirements:

MATLAB is the one and only software platform required for this project. Here, we make use of the well furbished and popular built in Image Processing Tool (IPT) for the processing of image, control and recognition. This tool supports the necessary application requirements,thus proving to be an efficient platform for developing this particular project. Using MATLAB software, we can also invoke various classes, functions such as Robot() and Activex, which are very important and necessary add-ons and are explained in detail in the later part of the document.

Clearly, not much of expensive hardware is needed. Infact, this project hardly uses any hardware components, making it a much more cost efficient and ideal project.

Chapter 3

Literature Review

3.1 Important MATLAB functions used:

3.1.1 imaqreset

SYNTAX imaqreset

imaqreset deletes any image acquisition objects that exist in memory and unloads all adaptors loaded by the toolbox. As a result, the image acquisition hardware is reset.

imaqreset is the image acquisition command that returns MATLAB® to the known state of having no image acquisition objects and no loaded image acquisition adaptors.

You can use imaqreset to force the toolbox to search for new hardware that might have been installed while MATLAB was running.

Note that imaqreset should not be called from any of the callbacks of a videoinput object, such as the StartFcn or FramesAcquiredFcn.

3.1.2 videoinput

SYNTAX
obj = videoinput(adaptorname)
obj = videoinput(adaptorname,deviceID)
obj = videoinput(adaptorname,deviceID,format)
obj = videoinput(adaptorname,deviceID,format,P1,V1,...)

obj = videoinput(adaptorname) constructs the video input object obj. A video input object represents the connection between MATLAB® and a particular image acquisition device. adaptorname is a text string that specifies the name of the adaptor used to communicate with the device. Use the imaqhwinfo function to determine the adaptors available on your system.

obj = videoinput(adaptorname,deviceID) constructs a video input object obj, where deviceID is a numeric scalar value that identifies a particular device available through the specified adaptor, adaptorname. Use the imaqhwinfo(adaptorname) syntax to determine the devices available through the specified adaptor. If deviceID is not specified, the first available device ID is used. As a convenience, a device's name can be used in place of the deviceID. If multiple devices have the same name, the first available device is used.

obj = videoinput(adaptorname,deviceID,format) constructs a video input object, where format is a text string that specifies a particular video format supported by the device or the full path of a device configuration file (also known as a camera file). To get a list of the formats supported by a particular device, view the DeviceInfo structure for the device that is returned by the imaqhwinfo function. Each DeviceInfo structure contains a SupportedFormats field. If format is not specified, the device's default format is used.

When the video input object is created, its VideoFormat field contains the format name or device configuration file that you specify.

obj = videoinput(adaptorname,deviceID,format,P1,V1,...) creates a video input object obj with the specified property values. If an invalid property name or property value is specified, the object is not created. The property name and property value pairs can be in any format supported by the set function, i.e., parameter/value string pairs, structures, or parameter/value cell array pairs.

To view a complete listing of video input object functions and properties, use the imaqhelp function.

imaqhelp videoinput

3.1.3 getsnapshot

SYNTAX
frame = getsnapshot(obj)

frame = getsnapshot(obj) immediately returns one single image frame, frame, from the video input object obj. The frame of data returned is independent of the video input object FramesPerTrigger property and has no effect on the value of the FramesAvailable or FramesAcquired property.

3.1.4 actxcontrol

SYNTAX
$h = actxcontrol('progid', position, fig_handle, event_handler,' filename')$

It creates an ActiveX control with the first four arguments, and sets its initial state to that of a previously saved control. MATLAB loads the initial state from the file specified in the string filename.
$Ifyoudonotwanttospecifyanevent_handler, youcanuseanemptystring('')asthefourthargument.$

3.1.5 getdata

SYNTAX
data = getdata(obj)

data = getdata(obj) returns data, which contains the number of frames specified in the FramesPerTrigger property of the video input object obj. obj must be a 1-by-1 video input object.

data is returned to the MATLAB® workspace in its native data type using the color space specified by the ReturnedColorSpace property.

You can use the MATLAB image or imagesc functions to view the returned data. Use imaqmontage to view multiple frames at once.

3.1.6 triggerconfig

SYNTAX
triggerconfig(obj,type)

triggerconfig(obj,type) configures the value of the TriggerType property of the video input object obj to the value specified by the text string type. For a list of valid TriggerType values, use triggerinfo(obj). type must specify a unique trigger configuration.

obj can be either a single video input object or an array of video input objects. If an error occurs, any video input objects in the array that have already been configured are returned to their original configurations.

3.1.7 impixel

Pixel color values.
SYNTAX
P = impixel(I,c,r)

P = impixel(I,c,r) returns the values of pixels specified by the row and column vectors r and c. r and c must be equal-length vectors. The kth row of P contains the RGB values for the pixel (r(k),c(k)).

Use normal button clicks to select pixels. Press Backspace or Delete to remove the previously selected pixel. To finish selecting pixels, adding a final pixel, press shift-click, right-click, or double-click. To finish selecting pixels without adding a final pixel, press Return.

3.2 Image Processing Toolbox

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB ® numeric computing environment. The toolbox supports a wide range of image processing operations, including:
Geometric operations
Neighborhood and block operations
Linear filtering and filter design Transforms
Image analysis and enhancement
Binary image operations
Region of interest operations

Many of the toolbox functions are MATLAB M-files, series of MATLAB statements that implement specialized image processing algorithms. You can view the MATLAB code for these functions using the statement:

$typefunctionname$

You can extend the capabilities of the Image Processing Toolbox by writing your own M-files, or by using the toolbox in combination with with other toolboxes, such as the Signal Processing Toolbox and the Wavelet Toolbox.

3.2.1 Images in MATLAB and the Image Processing Toolbox

The basic data structure in MATLAB is the array, an ordered set of real or complex elements. This object is naturally suited to the representation of images, real-valued, ordered sets of color or intensity data. (MATLAB does not support complex-valued images.)

MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image. (Pixel is derived from picture element and usually denotes a single dot on a computer display.) For example, an image composed of 200 rows and 300 columns of different colored dots would be stored in MATLAB as a 200-by-300 matrix.

This convention makes working with images in MATLAB similar to working with any other type of matrix data, and makes the full power of MATLAB available for image processing applications.

3.2.2 Image Types in the Toolbox

The Image Processing Toolbox supports four basic types of images:
Indexed images
Intensity images
Binary images

RGB images

This section discusses how MATLAB and the Image Processing Toolbox represent each of these image types.

Indexed Images

An indexed image consists of two arrays, an image matrix and a colormap. The colormap is an ordered set of values that represent the colors in the image. For each image pixel, the image matrix contains a value that is an index into the colormap. The colormap is an m-by-3 matrix of class double. Each row of the colormap matrix specifies the red, green, and blue (RGB) values for a single color:

color = [R G B]

R, G, and B are real scalars that range from 0 (black) to 1.0 (full intensity). The figure below illustrates the structure of an indexed image.

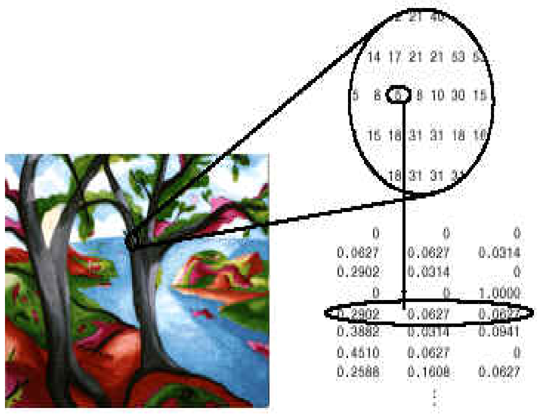


Figure 3.1: Indexed Image

The pixels in the image are represented by integers, which are pointers (indices) to color values stored in the colormap. The relationship between the values in the image matrix and the colormap depends on whether the image matrix is of class double or uint8. If the image matrix is of class double, the value 1 points to the first row in the colormap, the value 2 points to the second row, and so on. If the image matrix is of class uint8, there is an offset; the value 0 points to the first row in the colormap, the value 1 points to the second row, and so on. The uint8 convention is also used in graphics file formats, and enables 8-bit indexed images to support up to 256 colors. In the image above, the image matrix is of class double, so there is no offset. For example, the value 5 points to the fifth row of the colormap.

Intensity Images

MATLAB stores an intensity image as a single matrix, with each element of the matrix corresponding to one image pixel. The matrix can be of class double, in which case it contains values in the range [0,1], or of class uint8, in which case the data range is [0,255]. The elements in the intensity matrix represent various intensities, or gray levels, where the intensity 0 represents black and the intensity 1 (or 255) represents full intensity, or white. This figure depicts an intensity image of class double.

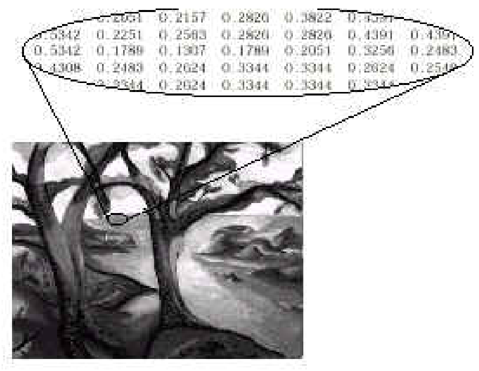


Figure 3.2: Gray-scale Image

Binary Images

In a binary image, each pixel assumes one of only two discrete values. Essentially, these two values correspond to on and off. A binary image is stored as a two-dimensional matrix of 0's (off pixels) and 1's (on pixels). A binary image can be considered a special kind of intensity image, containing only black and white. Other interpretations are possible, however; you can also think of a binary image as an indexed image with only two colors. A binary image can be stored in an array of class double or uint8. However, a uint8 array is preferable, because it uses far less memory. In the Image Processing Toolbox, any function that returns a binary image returns it as a uint8 logical array. The toolbox uses the presence of the logical flag to signify that the data range is [0,1]. (If the logical flag is off, the toolbox assumes the data range is [0,255].) This figure shows an example of a binary image.

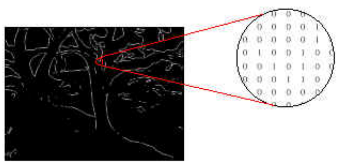


Figure 3.3: Binary Image

RGB Images

Like an indexed image, an RGB image represents each pixel color as a set of three values, representing the red, green, and blue intensities that make up the color. Unlike an indexed image, however, these intensity values are stored directly in the image array, not indirectly in a colormap. In MATLAB, the red, green, and blue components of an RGB image reside in a single m-by-n-by-3 array. m and n are the numbers of rows and columns of pixels in the image, and the third dimension consists of three planes, containing red, green, and blue intensity values. For each pixel in the image, the red, green, and blue elements combine to create the pixel's actual color. For example, to determine the color of the pixel (112,86), look at the RGB triplet stored in (112,86,1:3). Suppose (112,86,1) contains the value 0.1238, (112,86,2) contains 0.9874, and (112,86,3) contains 0.2543. The color for the pixel at (112,86) is:
0.1238 0.9874 0.2543
An RGB array can be of class double, in which case it contains values in the range [0,1], or of class uint8, in which case the data range is [0,255]. The figure below shows an RGB image of class double:

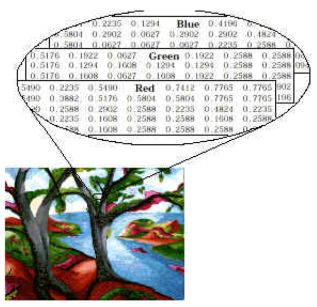


Figure 3.4: RGB Image

3.2.3 Converting Images to Other Types

For certain operations, it is helpful to convert an image to a different image type. For example, if you want to filter a color image that is stored as an indexed image, you should first convert it to RGB format. When you apply the filter to the RGB image, MATLAB filters the intensity values in the image, as is appropriate. If you attempt to filter the indexed image, MATLAB simply applies the filter to the indices in the indexed image matrix, and the results may not be meaningful. The Image Processing Toolbox provides several functions that enable you to convert any image to another image type. These functions have mnemonic names; for example, ind2gray converts an indexed image to a grayscale intensity format. Note that when you convert an image from one format to another, the resulting image may look different from the original. For example, if you convert a color indexed image to an intensity image, the resulting image is grayscale, not color.

Chapter 4

Processing

4.1 Block Diagram Representation:

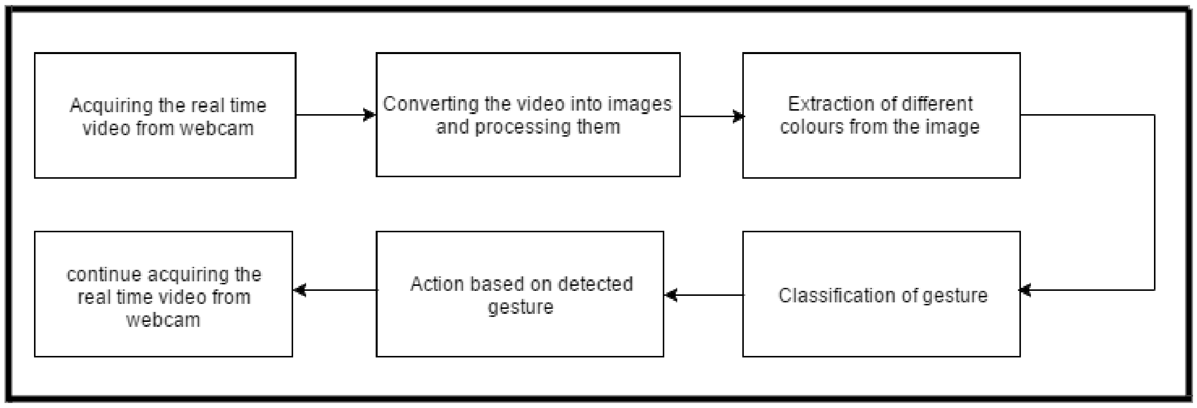

Acquiring the real time video from webcam
Converting the video into images and processing them
Extraction of different colours from the image
Classification of gesture
Action based on detected gesture
continue acquiring the real time video from webcam

Figure 4.1: Block Diagram

When we perform some gestures in front of the webcam, the video tracked by the webcam is converted to images. The images consist of various color components and we extract these color components from the image and process it. Each gesture is defined by various color components and we classify the gesture based on the extracted color. Depending on the classified gesture the corresponding action is executed. At the same time we keep on acquiring the video input though the webcam.

4.2 Working Methodology:

The image processing pipeline used for hand gesture recognition system involved following:

4.2.1 Hand image acquisition:

The picture is taken by camera, digitized and prepared for further processing. It is usually done with the help of frame grabber. A frame grabber can be described as a GUI or interface which allows user to take snapshots by initializing the camera hardware and trigger manually for taking snaps.

4.2.2 Hand image segmentation:

In this step those area in the image which represents the skin part of hand are separated from the background by applying different approaches using different color models. Using different color models helps in comparing the efficiency of segmentation on basis of models respectively.

4.2.3 Feature extraction:

Aim of this step is to derive smallest possible amount of features out of the segmented hand region, in order to differentiate the different gestures. As in this we use output of previous step and for effective utilization, result image is converted into binary and then parameters as pixel intensities and quantity helps in forming base for feature extraction followed by cropping and standardization.

4.2.4 Recognition and Matching:

By this step we conclude the exact gesture and by applying suitable matching technique results the recognition of gesture based on database images. For efficient matching algorithm we can implement neural networks, skeleton matching, chain coding, or can define algorithm which is capable of fetching best match for the input one. Here basically we applied the matching/recognition phase on the basis of presence of number of white pixels which are of the binary image obtained from image segmentation operation on input image. Then this magnitude of number of pixels is matched with those previously stored images in database. As the match is found out then user is supplied with all in between processed images along with the execution of application associated with gestures.

4.2.5 Gesture based interaction:

Finally after obtaining the suitable match with respect to the gesture, associates the gesture with the assigned functionality i.e. launching the application.

4.3 Working of System:

A gesture recognition system is one which is capable of perceiving gesture by a means and then makes the system to act accordingly. So, here introducing a system which will recognize the gesture on the basis of image and after processing on image the associated application is executed. The image on which processing is being carried out is a static part of human hand gesture in the form of signals by fingers. The following steps are being carried out:

1. Firstly the image is being capture/acquired with the help of webcam or external acquisition device, the condition applied is the background must be uniform.

2. Then after the image is processed and the human hand is being separated out from the background i.e. segmentation is done.

3. Then in order to match the supplied gesture with that in the database certain features are to be extracted out. Here we are concerned about the number of white pixels as the outcome from the previous step we get a binary image.

4. So, on the basis of number of white pixels the particular gesture is recognized to that of images present in database.

5. As soon as the gesture is recognized the associated application is being executed.

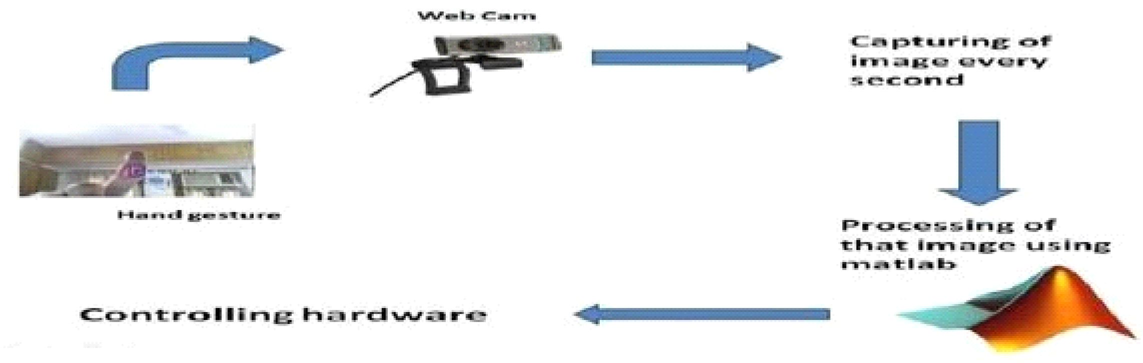

Web Cam
Capturing of image every second
Hand gesture
Processing of that image using matlab
Controlling hardware

Figure 4.2: Flow Cycle of the process

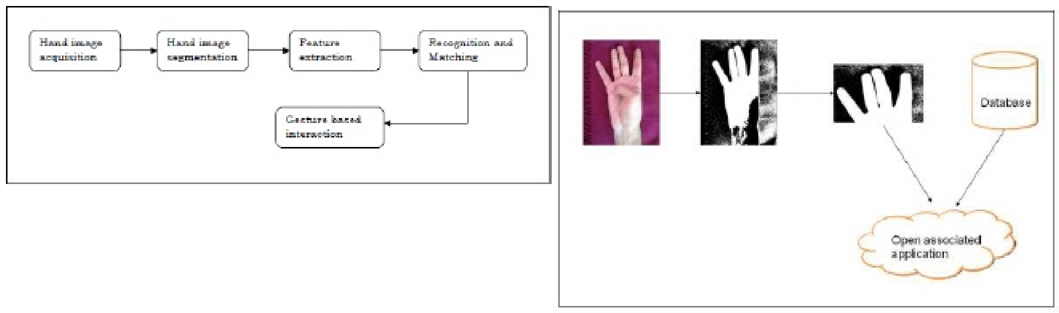


Figure 4.3: Flow diagram of gesture recognition

Chapter 5

Application Development

Here in our project we have employed this gesture recognition technology to develop and demonstrate the following applications mainly. All the applications were developed by coding with the help of MATLAB software itself and even the screenshots of the working and results are shown below.

5.1 Power Point Presentation Control:

With respect to the power point operations, here we are dealing with moving to a next slide or a previous slide. It can be observed that we can move to the next slide or the previous slide using the mouse wheel. Based on which direction we roll the mouse wheel, the slide changes accordingly. So here what is needed is that we should simulate the mouse wheel control using MATLAB. First of all, this requires the control of the mouse pointer in the first place. For this we import the java utility toolkit into the MATLAB environment that provides us with the control of the mouse. Based on which direction the mouse pointer is moved, the slide changes accordingly. So each time we move the mouse pointer in a particular direction, with the help of colored tapes, we are simulating a mouse wheel control that changes the slides accordingly. It is important to note that we are calculating the displacement of the mouse pointer based on the centroid of the detected region.

Working

Initially a video object with set properties is triggered which continually obtains a stream from the camera that is being used. Then we are importing the java utility toolkit and taking control of the mouse. The user then selects the desired region and a bitmap image is created and corresponding to the selected color, the user can observe the movement of the mouse pointer corresponding to the movement in the colored tape. Each instant the centroid of the colored region is calculated and all the mouse pointer movements are taking place with respect to this calculation. If the centroid value exceeds a certain threshold, then a corresponding mouse wheel roll action is stimulated which changes the slide accordingly.

5.2 Media Player Control:

In the media player control, we are dealing with function such as pausing the currently playing file, increasing the volume of the currently playing file, and decreasing the volume of the currently playing file. We are creating an instance of the media player object using the ActiveX control feature. Here we are detecting the number of pixels and looking at the value of the centroid to take necessary actions such as increasing the volume or decreasing it or making the current file to pause.

WORKING

As before initially a video object with set properties is triggered which continually obtains a stream from the camera that is being used. As before the user selects the region of interest from the obtained snapshot so as to detect the particular color. Then we use the actxcontrol function to open a media player object with specific properties. Then the uigetfile function is used to open a dialog box that allows the user to select a particular audio or video file. Once the user selects the file, the media player starts playing it. When the user brings his fingers in front of the webcam, the total number of detected pixels is calculated and if it exceeds a threshold then the media file is paused. Else if the centroid of the region of interest exceeds certain thresholds then the volume of the currently playing file either increases or decreases depending on the centroid value obtained.

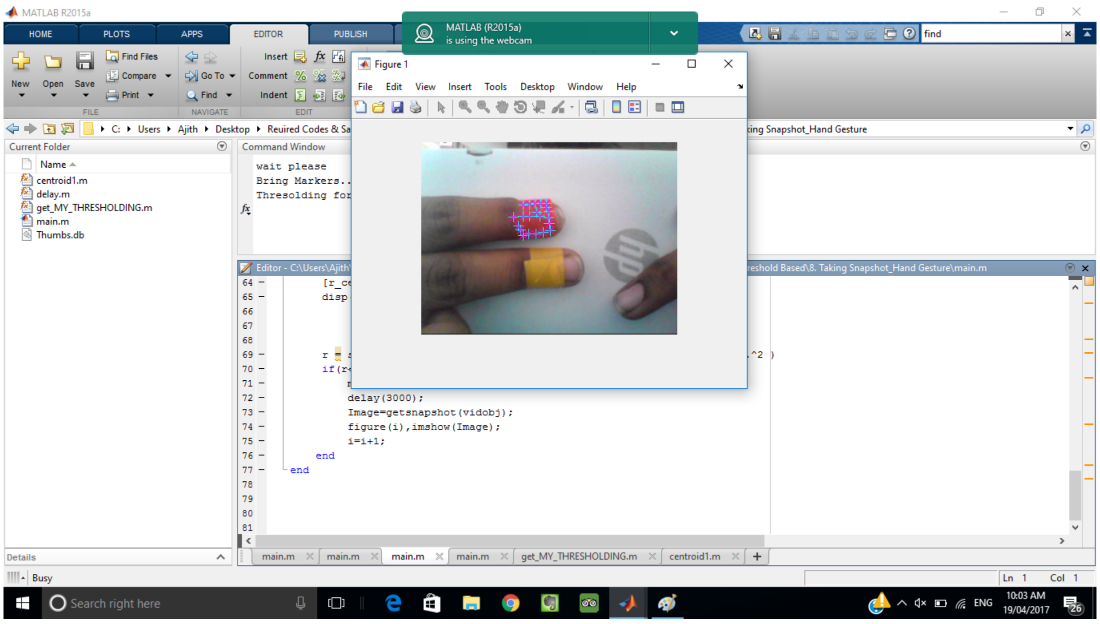
5.3 Photography Using Hand Gestures:

Here by photography, we mean obtaining a snapshot. This application is very much similar to a camera wherein it is used to obtain a picture. There is a major difference between the earlier applications and this one since it requires no external dependencies. All we need is the access to the camera as in the earlier applications. Two colored tapes are used rather than one. Here, we are calculating the distance between the two centroids of the two colored tapes region and if it becomes less than a certain threshold, the image capture process is initiated.

WORKING

As in earlier applications, initially a video object with set properties is triggered which continually obtains a stream from the camera that is being used. Here there are two snapshots obtained to select two colored regions instead of one. Each time the different colored regions are selected with the help of the impixel function. The selected colors are detected each time the user brings the tapes on his/her fingers nearer to the camera. The centroid corresponding to each colored regions is calculated. If the distance between the two centroids becomes less than a set threshold, then the image capture is initiated with a certain delay of convenience so that it gives the user some time to make adjustments that are necessary. The image capture is initiated with the help of the getsnapshot function which captures an image object from the currently streaming video object after the delay duration has passed. The captured images will be present in the MATLAB window so that the user can come and checkout the captured images later and save them if required.

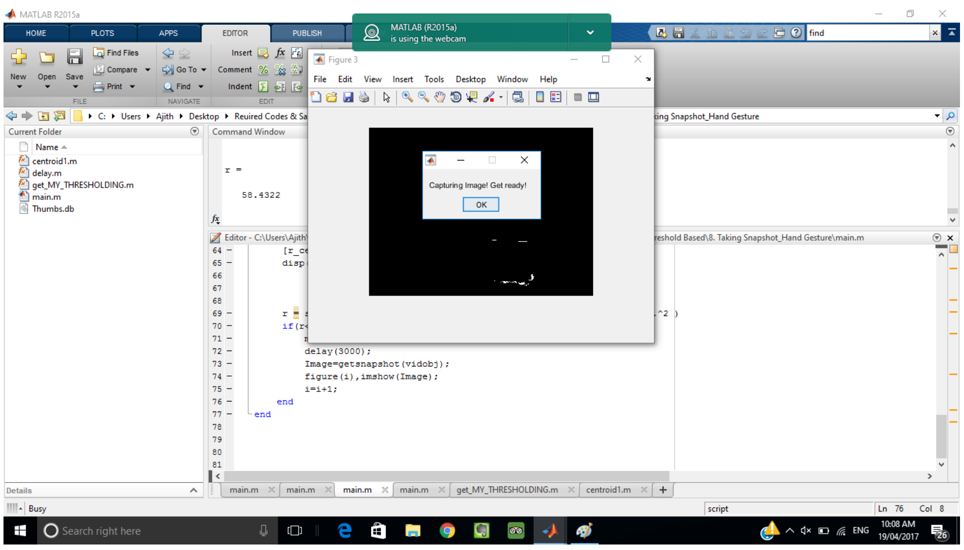
A few images below demonstrate the processes involved in the above mentioned applications.



```
wait please
Bring Markers..
Thresolding for
```

```
        [r_ce
        disp



        r = s                                        .^2 )
        if(r<
            r
            delay(3000);
            Image=getsnapshot(vidobj);
            figure(i),imshow(Image);
            i=i+1;
        end
    end
```

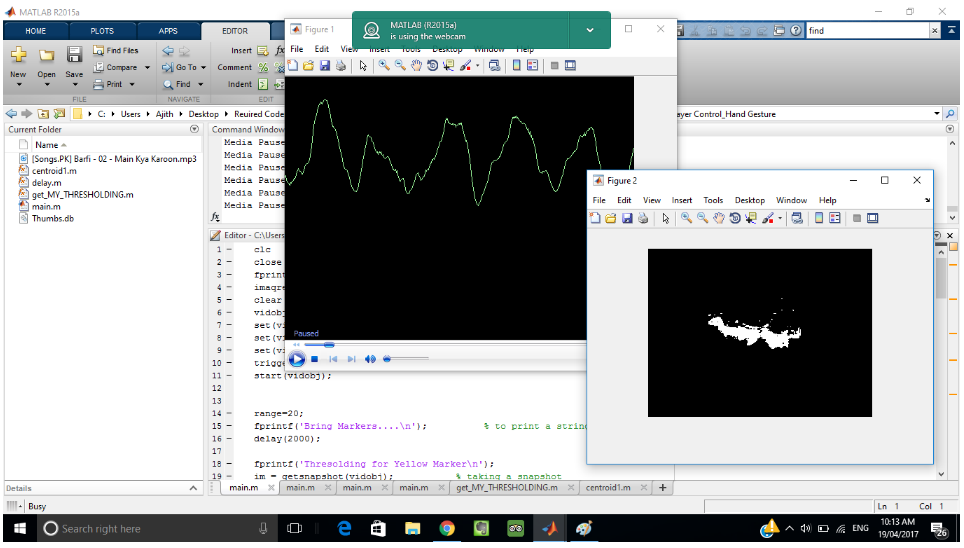
Figure 5.1: Selecting the desired region (Color Masking)



```
r =

   58.4322
```

```
        [r_ce
        disp(



        r = s                                                    .^2 )
        if(r<
            r
            delay(3000);
            Image=getsnapshot(vidobj);
            figure(i),imshow(Image);
            i=i+1;
        end
    end
```

Figure 5.2: Demonstration of Photography application



```
Media Pause
Media Pause
Media Pause
Media Pause
Media Pause
Media Pause
```

```
clc
close
fprint
imaqre
clear
vidobj
set(vi
set(vi
set(vi
trigge
start(vidobj);


range=20;
fprintf('Bring Markers....\n');          % to print a string
delay(2000);

fprintf('Thresolding for Yellow Marker\n');
im = getsnapshot(vidobj);             % taking a snapshot
```

Figure 5.3: Demonstration of Media Player Control

5.4 Mouse Interfacing:

MATLAB directly does not support mouse pointer control. Using Matlab only cursor can be moved.

Set(0,'PointerLocation',[X,Y])

But using only Matlab right click and left click can not be done. For this Java is used. Java Robot class can be easily interfaced with Matlab which is used for Left Click and Right Click operation.

java.awt.Robot

This class is used to generate native system input events for the purpose of test automation, self-running demos and other applications where the control of mouse and keyboard is needed.

robot = java.awt.Robot;

robot.mouseMove(X,Y)

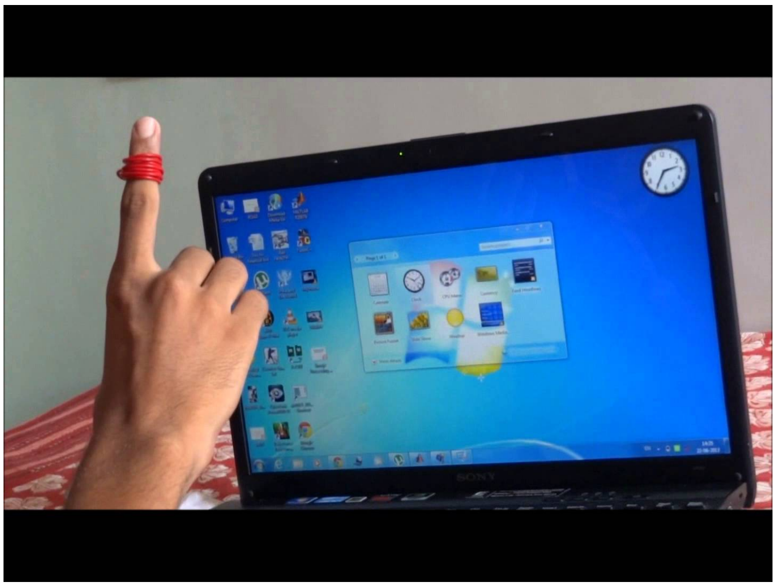


Figure 5.4: Interaction with the PC

Chapter 6

Merits and Demerits

As every other project implemented, this project has it's own pros and cons too. These are discussed in the section below:

6.1 Merits:

Few advantages of using the gesture language are given below:

(a) Low power requirement.

(b) Simple circuitry as it does not require special hardware.

(c) Higher security as directionality of the beam helps ensure that data isn't leaked or spilled to nearby devices as it is transmitted.

(d) Devices can be controlled more comfortably.

6.2 Demerits and Challenges:

Some basic limitations of using such communication are given below:

(a) It has distance limit in controlling devices.

(b) Line of sight communication as transmitters and receivers must be almost directly aligned (i.e. able to see each other) to communicate.

(c) The data rate or speed of transmission is lower than a typical wired transmission.

There are many challenges associated with the accuracy and usefulness of gesture recognition software. For image-based gesture recognition there are limitations on the equipment used and image noise. Images or video may not be under consistent lighting, or in the same location. Items in the background or distinct features of the users may make recognition more difficult.

The variety of implementations for image-based gesture recognition may also cause issue for viability of the technology to general usage. For example, an algorithm calibrated for one camera may not work for a different camera. The amount of background noise also causes tracking and recognition difficulties, especially when occlusions (partial and full) occur. Furthermore, the distance from the camera, and the camera's resolution and quality, also cause variations in recognition accuracy. In order to capture human gestures by visual sensors, robust computer vision methods are also required, for example for hand tracking and hand posture recognition or for capturing movements of the head, facial expressions or gaze direction.

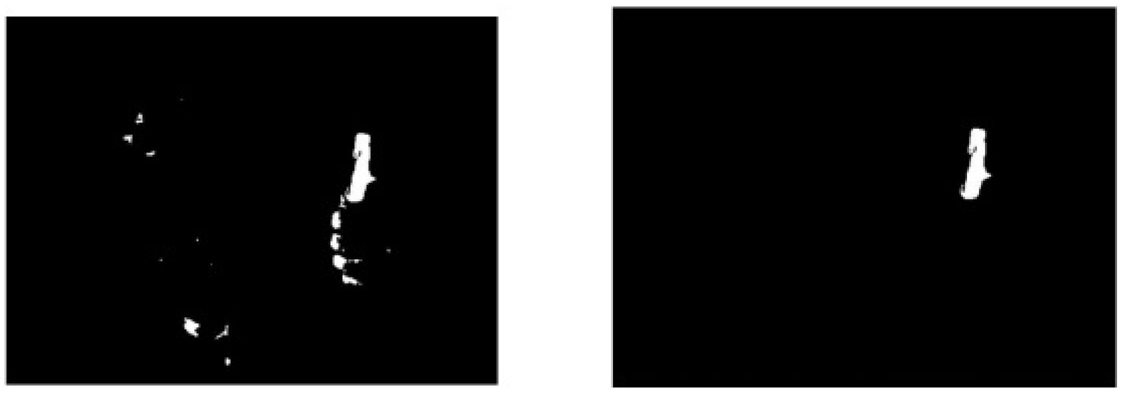


Figure 6.1: Converted Bitmap Images

Chapter 7

Conclusion and Future Work

The objectives and goal of this project are achieved successfully. We successfully implemented the image segmentation and recognition part in MATLAB. Where the images are static and provides interface which is user friendly. And the most important thing is that there is no extra hardware to perform all these tasks.

But taking the dynamic environment in mind this project is not enough robust and safe to guarantee result. Here we only considered limited set of gestures but if we take into account more number of gestures then we can facilitate more commands and the system will become more interactive. As the system can serve in vital ways so the system should not involve any other external part/hardware except computer system equipped with webcam. This helps in keeping the cost minimum and everyone is able to use this application also able to own it. Using different colors and with different hand gestures, different types of interfacing can be done. In case of hand-mouse interfacing, the computer can be controlled from a distant point depending on the resolution of camera. If the camera is linked to the computer either by direct connection or by online; then system can be controlled remotely.

Future Work:

1. Hand Gesture recognition can be optimized using developed techniques.

2. Instead of using RGB space , HSV colour space can be used to avoid light intensity fluctuations and thereby giving better results.

3. HAND can be used as a remote control to TV if subject to a camera is embedded.

4. Many entertainment applications can be developed e.g. virtual piano playing , video game etc.

5. Using high resolution camera or better sensor like a Kinect; high end application can be developed. This can be also utilised in industries machine controlling applications.

REFERENCES

[1]. Henrik Birk and Thomas Baltzer Moeslund, “Recognizing Gestures From the Hand Alphabet Using Principal Component Analysis, Master’s Thesis, Laboratory of Image Analysis, Aalborg University, Denmark, 1996.

[2]. Tosas, M.,Visual Articulated Hand Tracking for Interactive Surfaces.,University of Nottingham, 2006

[3]. Gonzalez, Rafael C., and Woods, Richard E., Digital Image Processing, Pearson Education, 2003

[4]. Gonzalez, Rafael C., and Woods, Richard E., Digital Image Processing using MATLAB, 2nd Edition, 2009

[5]. Zarit, B. D., Super, B. J. and Quek, F. K. H. (1999). Comparison of five color models in skin pixel classification. In ICCV’99 Int’l Workshop on recognition, analysis and tracking of faces and gestures in Real-Time systems, 58- 63.

[6]. Zhang, Z., Wu, Y., Shan, Y. and Shafer. S. (2001). Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In Workshop on Perceptive User Interfaces. ACM Digital Library, ISBN 1-58113-448-7.

[7]. Stafford, Q. and Robinson, P. (1996). BrightBoard: A Video-Augmented Environment. In Proc. of the CHI96, pp. 134-141.

[8]. http://channel9.msdn.com/coding4fun/articles.

[9]. Panwar, M., "Hand Gesture Recognition based on Shape Parameters", Computing, Communication and Applications (ICCCA), 2012 International Conference, Page(s): 1 - 6

[10]. Tomita A., Ishii R., "hand Shape Extraction from a Sequence of digitized grayscale Images", IEEE Journals

[11]. http://www.mathworks.com

[12]. www.alldatasheet.com

[13]. http://www.digi.com/support/productdetl.jsp?pid=4091

[14]. Burande, C.A.; Tugnayat, R.M.; Choudhary, N.K., "Advanced recognition techniques for human computer interaction", Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference ,Volume: 2 , Page(s): 480 - 483

[15]. Wang X., "A Six-Degree-of-Freedom Virtual Mouse Based on Hand Gestures", Electrical and Control Engineering (ICECE), 2010 International Conference, Page(s): 257 - 260

[16]. Isard, M. and MacCormick, J. (2000). Hand tracking for vision-based drawing. Technical report, Visual Dynamics Group, Dept. Eng. Science, University of Oxford.