Master Hand Technology For The HMI Using Hand Gesture And Colour Detection

A thesis submitted in partial fulfilment of the
requirement for the degree of

Bachelor of technology
In
Electronics & Instrumentation Engineering

by

Alok Nanda
Roll No- 108EI005
&
Asutosh Mishra
Roll No- 108EI001

Under the guidance of
Prof U.C.Pati

Department of Electronics and communication Engineering
National Institute of Technology, Rourkela

Master Hand Technology For The HMI Using Hand Gesture And Colour Detection

A thesis submitted in partial fulfilment of the
requirement for the degree of

Bachelor of technology
In
Electronics & Instrumentation Engineering

by

Alok Nanda
Roll No- 108EI005
&
Asutosh Mishra
Roll No- 108EI001

Under the guidance of
Prof U.C.Pati

Department of Electronics and communication Engineering
National Institute of Technology, Rourkela

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the project report titled "Master Hand Technology for the HMI Using Hand Gesture and Colour Detection " submitted by Alok Nanda (Roll No: 108EI005) and Asutosh Mishra (Roll No: 108EI001) in the partial fulfilment of the requirements for the award of Bachelor of Technology Degree in Electronics and Instrumentation Engineering during session 2008-2012 at National Institute of Technology, Rourkela (Deemed University) and is an authentic work carried out by them under my supervision and guidance.

Prof. U. C. Pati

Department of Electronics and Communication Engg.

Date: National Institute of Technology

Rourkela-769008

ACKNOWLEDGEMENT

We would like to take this opportunity to express our gratitude and sincere thanks to our respected supervisor **Prof. U. C. PATI** for his guidance, insight, and support he has provided throughout the course of this work. We would like to articulate our profound gratitude and indebtedness to **Prof. L. P. ROY and Prof. S. MEHER** for teaching "Digital Image Processing" in such depth and setting up a strong fundamental base in Image/Video Processing that enabled us achieve this success. We would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela 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: 9th May 2010 **ALOK NANDA (108EI005)**

NIT Rourkela **ASUTOSH MISHRA (108EI001)**

ABSTRACT

Master Hand Technology uses different hand gestures and colors to give various commands for the Human-Machine(here Computer) Interfacing. Gestures recognition deals with the goal of interpreting human gestures via mathematical algorithm. Gestures made by users with the help of a color band and/or body pose , in two or three dimensions , get translated by software/image processing into predefined commands .The computer then acts according to the command. There have been a lot work already developed in this field either by extracting hand gesture only or extracting hand with the help of color segmentation. In this project, both hand gesture extraction and color detection used for better, faster, robust, accurate and real-time applications. Red, Green, Blue colors are most efficiently detected if RGB color space used. Using HSV color space, it can be extended to any no of colors. For hand gesture detection, the default background is captured and stored for further processing. Comparing the new captured image with background image and doing necessary extraction and filtering, hand portion can be extracted. Then applying different mathematical algorithms different hand gestures detected. All this work done using MATLAB software. By interfacing a portion of Master hand or/and color to mouse of a Computer, the computer can be controlled same as the mouse. And then many virtual (Augmented reality) or PC based application can be developed (e.g. Calculator, Paint). It does not matter whether the system is within your reach or not; but a camera that is linked with the system must have to be near-by . Showing different gestures by your Master-Hand , the computer can be controlled remotely. If the camera can be set-up online, then the computer can be controlled even from a very far place online.

Contents

CERTIFICATE i
ACKNOWLEDGEMENT ii
ABSTRACT iii
1. INTRODUCTION 1
1.1 HUMAN COMPUTER INTERFACE SYSTEM 2
1.2 GESTURES 2
1.3 MASTER HAND TECHNOLOGY 3
1.4 GESTURE BASED APPLICATIONS 3
1.5 COLOUR SPACE 4
1.6 COMPONENTS 5
2. LITERATURE REVIEW 6
2.1 Important Matlab Functions Used : 7
3. PROCESSING AND APPLICATION DEVELOPMENT 18
3.1 Algorithm: (For Red Color Detection) 19
3.2 Mouse Interfacing : 21
3.4 Algorithm for Hand Extraction And Finger Count : 22
3.5 Virtual Calculator : 24
3.6 PC Calculator: 25
3.7 PC Paint : 26
3.8 HCI(Human-Computer Interface) : 27
3.9 Remote Controlling Using Cloud : 28
3.10 Gesture based Hardware Interfacing Applications: 29
3.10.1 Hardware 29
3.10.2 Software 29
3.10.3 ATMEGA 640 Features- 29
3.10.4 Program- 30
4. CONCLUSION AND FUTURE WORK 37
4.1 CONCLUSION: 38
4.2 FUTURE WORKS: 38
REFERENCES 39

LIST OF FIGURES

Fig 2.1.a- Input Colored RGB Image Frame	7
Fig 2.1.b- Converted Gray Image	7
Fig 2.2.a- Color Image	8
Fig 2.2.b- Gray Image	8
Fig 2.2.c- Black-White Image	8
Fig 2.3.a- Before area opening	9
Fig 2.3.b- After area opening	9
Fig 2.4.a- Before image filling	10
Fig 2.4.b- After image filling	10
Fig 2.5.a, 2.5.b- Image perimeter example	11
Fig 3.1.a- Input Image Frame	20
Fig 3.1.b- Mirrored Image	20
Fig 3.1.c- Red portion extracted	20
Fig 3.1.d-Median filtered Image	20
Fig 3.1.e-Binary Image	20
Fig 3.1.f-Small noise/objects extracted	20
Fig 3.1.g-Final Output	20
Fig 3.2.a- Finger Count 5	23
Fig 3.2.b- Finger Count 4	23
Fig 3.2.c- Finger Count 3	24
Fig 3.2.d- Finger Count 2	24
Fig 3.3.a- Virtual Calculator	24

Fig 3.3.b- Calculator design 25

Fig 3.3.c- PC Calculator 25

Fig 3.4.a- PC Paint Snapshot 26

Fig 3.4.b- Final Paint 26

Fig 3.5 HCI snapshot 27

Fig 3.6 Remote Controlling Application 28

Fig 3.7 ATmega 640 Development Board 30

Fig 3.8 Block Diagram of Window Control Hardware interfacing application 36

1. INTRODUCTION

1.1 HUMAN COMPUTER INTERFACE SYSTEM

Computer is used by many people either at their work or home or in their spare-time. Special input and output devices have been designed over the years with the purpose of easing the communication gap between computers and humans. Keyboard and mouse are frequently used for this purpose. Each new device is seen as an attempt to increase the intelligence-level of computer and making humans able to perform more complicated communication with the computer. This has been possible due to the result oriented efforts made by computer professionals for creating successful human computer interfaces . As the complexities of human needs have turned into many folds and continues to grow so, the need for Complex programming ability and intuitiveness are critical attributes of computer programmers to survive in a competitive environment. The computer programmers have been immensely successful in easing the communication between computers and human.[1] With the emergence of every new product in the market; it attempts to ease the complexity of jobs performed. For instance, it has helped in facilitating tele operating, robotic use, better human control over complex work systems like cars, planes and monitoring systems. Earlier, Computer programmers were avoiding such kind of complex programs as the focus was more on speed than other modifiable features. However, a shift towards a user friendly environment has driven them to revisit the focus area .The idea is to make computers understand human interactions and develop a user friendly human computer interfaces (HCI).[1] Making a computer understand speech, facial expressions and human gestures are some steps towards it. Gestures are a type of non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. The project aims to understand and utilize human hand gestures by creating an HCI. Coding of these hand gestures into machine language requires a complicated programming algorithm.

1.2 GESTURES

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. [17] [18] 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

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. [26] 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.

1.3 MASTER HAND TECHNOLOGY

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 colour codes attached to it gives additional flexibility in usage and applications.

1.4 GESTURE BASED 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 : Its 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.

Virtual Calculator: Virtual Calculator is like an augmented reality. It is designed in Matlab in the same interfacing program. Depending on the position of finger- peak of Master hand, application gives input to the calculator. It checks for 5 frames. If for 5 frames, the finger-peak remain in a particular region or number region, then that number or symbol is given as input.

PC Calculator: PC Calculator is the original calculator application given in windows platform by Microsoft. In PC Calculator, finger-peak of Master hand is interfaced with mouse cursor. If the mouse cursor remains within +-20 pixels for 5 frames, then Left Click event activated by java Robot class and if the cursor remains within +-20 pixels for 8 frames, then Right Click event activated by java Robot class. By this method, input is given to the calculator.

PC Paint : Above paint is done with the help of hand and without using mouse. In above application, two color pins used (Red, Blue). Red is used for left click and cursor movement

and blue color is used for dragging. If the cursor remains within +-20 pixels for 5 continuous frames, then Left click gets triggered

Tele presence : There may raise the need of manual operations in some cases such as system failure or emergency hostile conditions or inaccessible remote areas. Often it is impossible for human operators to be physically present near the machines. Telepresence is that area of technical intelligence which aims to provide physical operation support that maps the operator to carry out the necessary task.

Remote Controlling of Hardware: Using dropbox, google drive; a captured image or video can be sent to any location. Processing the image, the given command or gesture can be found out. Then interfacing the computer to any hardware e.g motor through Atmega or any other medium, any hardware interfacing application can be executed.

1.5 COLOUR 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]

Generic Color Models:

1. **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. **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]
3. **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° with respect to the YUV color space. The YDbDr scheme used by SECAM television is rotated in another way. [5]
4. **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]
5. **HSV** (**h**ue, **s**aturation, **v**alue), also known as HSB (hue, saturation, **b**rightness) 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 colorspace, and its components and colorimetry are relative to the RGB colorspace from which it was derived. [5]
6. **HSL** (**h**ue, **s**aturation, **l**ightness/**l**uminance), also known as HLS or HSI (hue, saturation, **i**ntensity) 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]

1.6 COMPONENTS

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 , hardwired or wireless,to main control computer,where the image is processed and necessary task is executed ***.***

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. LITERATURE REVIEW

2.1 Important Matlab Functions Used :

2.1.1 rgb2gray : Convert RGB image or colormap to grayscale.

Syntax :

I = rgb2gray(RGB)

newmap = rgb2gray(map)

Description :

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.newmap = rgb2gray(map) returns a grayscale colormap equivalent to map. If the input is an RGB image, it can be of class uint8, uint16, single, or double. [11] The output image I is of the same class as the input image. If the input is a colormap, the input and output colormaps are both of class double.

e.g *RGBImage = getsnapshot(vid);*

grayImage = rgb2gray(RGBImage);

Fig- 2.1.a

Fig- 2.1.b

2.1.2 Im2bw : Convert image to binary image, based on threshold

Syntax :

BW = im2bw(I, level)

BW = im2bw(X, map, level)

BW = im2bw(RGB, level)

Description :

BW = im2bw(I, level) converts the grayscale image I to a binary image. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). Specify level in the range [0,1].

This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 is midway between black and white, regardless of class. To compute the level argument, you can use the function graythresh. If you do not specify level, im2bw uses the value 0.5.BW = im2bw(X, map, level) converts the indexed image X with colormap map to a binary image.BW = im2bw(RGB, level) converts the true color image RGB to a binary image.If the input image is not a grayscale image, im2bw converts the input image to grayscale, and then converts this grayscale image to binary by thresholding.

bw = im2bw(grayImage,0.4);

Fig 2.2.a

Fig 2.2.b

Fig 2.2.c

2.1.3 medfilt2 : 2-D median filtering

Syntax :

B = medfilt2(A, [m n])

B = medfilt2(A)

Description :

Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.B = medfilt2(A, [m n]) performs median filtering of the matrix A in two dimensions. Each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image. medfilt2 pads the image with 0s on the edges, so the median values for the points within [m n]/2 of the edges might appear distorted.B = medfilt2(A) performs median filtering of the matrix A using the default 3-by-3 neighborhood. [11]

diff_im = medfilt2(diff_im, [3 3]);

2.1.4 bwareaopen : Morphologically open binary image (remove small objects)

Syntax :

BW2 = bwareaopen(BW, P)

Description :

BW2 = bwareaopen(BW, P) removes from a binary image all connected components (objects) that have fewer than P pixels, producing another binary image, BW2. The default connectivity is 8 for two dimensions, 26 for three dimensions, and conndef(ndims(BW), 'maximal') for higher dimensions. BW can be a logical or numeric array of any dimension, and it must be nonsparse. [11] The return value BW2 is of class logical.

diff_im = bwareaopen(diff_im,250);

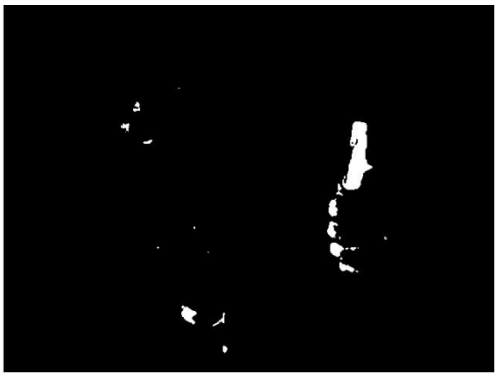


Fig 2.3.a

Fig 2.3.b

All the small objects of size below 250 pixels have been eliminated.

2.1.5 imfill : Fill image regions and holes

Syntax :

BW2 = imfill(BW)

Description :

BW2 = imfill(BW) displays the binary image BW on the screen and lets you define the region to fill by selecting points interactively by using the mouse. To use this interactive syntax, BW must be a 2-D image. Press Backspace or Delete to remove the previously selected point. A shift-click, right-click, or double-click selects a final point and starts the fill operation. Pressing Return finishes the selection without adding a point. The input image can be numeric or logical, and it must be real and non-sparse. It can have any dimension. The output image has the same class as the input image. [11]

BW5 = imfill(BW4,'holes');

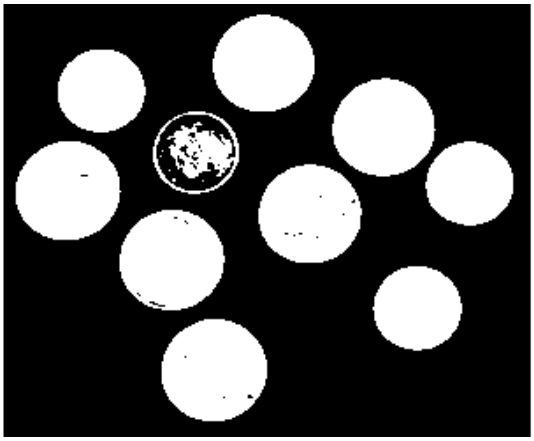


Fig 2.4.a

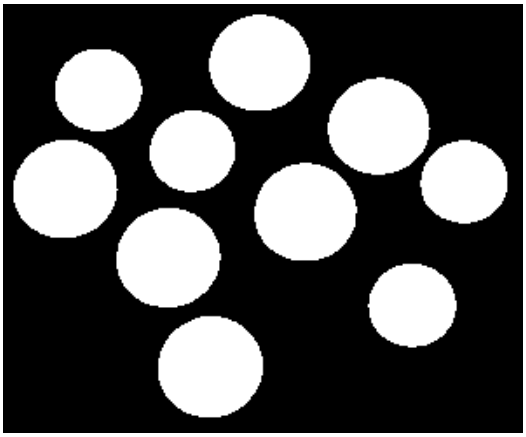


Fig 2.4.b

2.1.6 bwlabel : Label connected components in 2-D binary image

Syntax :

L = bwlabel(BW, n)

Description :

L = bwlabel(BW, n) returns a matrix L, of the same size as BW, containing labels for the connected objects in BW. The variable n can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects. If the argument is omitted, it defaults to 8.The elements of L are integer values greater than or equal to 0. The pixels labeled 0 are the background. [11] The pixels labeled 1 make up one object; the pixels labeled 2 make up a second object; and so on. BW can be logical or numeric, and it must be real, two-dimensional, and nonsparse. L is of class double.

bw = bwlabel(diff_im, 8);

2.1.7 bwperim : Find perimeter of objects in binary image

Syntax :

BW2 = bwperim(BW1)

Description :

BW2 = bwperim(BW1) returns a binary image containing only the perimeter pixels of objects in the input image BW1. A pixel is part of the perimeter if it is nonzero and it is connected to at least one zero-valued pixel. The default connectivity is 4 for two dimensions, 6 for three

dimensions, and conndef(ndims(BW), 'minimal') for higher dimensions. BW1 must be logical or numeric, and it must be nonsparse. BW2 is of class logical.

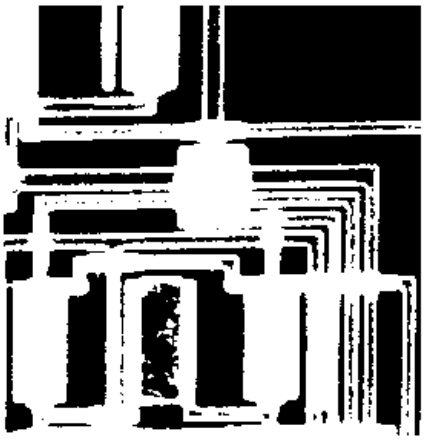


Fig 2.5.a

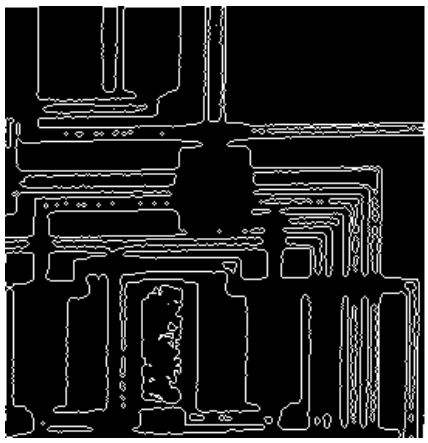


Fig 2.5.b

2.1.8 bwboundaries : Trace region boundaries in binary image

Syntax :

B = bwboundaries(BW)

Description :

B = bwboundaries(BW) traces the exterior boundaries of objects, as well as boundaries of holes inside these objects, in the binary image BW. bwboundaries also descends into the outermost objects (parents) and traces their children (objects completely enclosed by the parents). BW must be a binary image where nonzero pixels belong to an object and 0 pixels constitute the background. The following figure illustrates these components. bwboundaries returns B, a P-by-1 cell array, where P is the number of objects and holes. Each cell in the cell array contains a Q-by-2 matrix. [11] Each row in the matrix contains the row and column coordinates of a boundary pixel. Q is the number of boundary pixels for the corresponding region. BW can be logical or numeric and it must be real, two-dimensional, and nonsparse. L and N are double. A is sparse logical.

2.1.9 regionprops : Measure properties of image regions

Syntax :

STATS = regionprops(BW, properties)

Descriptions :

STATS = regionprops(BW, properties) measures a set of properties for each connected component (object) in the binary image, BW. The image BW is a logical array; it can have any dimension. [11] STATS is a structure array with length equal to the number of objects in BW, . The fields of the structure array denote different properties for each region, as specified by
properties.

Shape Measurement –

Area, EulerNumber, Orientation, BoundingBox, Extent, Perimeter, Centroid, Extrema, PixelIdxList, ConvexArea, FilledArea, PixelList, ConvexHull, FilledImage, Solidity, ConvexImage, Image, SubarrayIdx, Eccentricity, MajorAxisLength, EquivDiameter, MinorAxisLength

Pixel Value Measurement –

MaxIntensity, MinIntensity, WeightedCentroid, MeanIntensity, PixelValue

2.1.10 rectangle : Create 2-D rectangle object

Syntax :

rectangle
rectangle('Position',[x,y,w,h])
rectangle('Curvature',[x,y])
rectangle('PropertyName',propertyvalue,...)
h = rectangle(...)

Description :

rectangle draws a rectangle with Position [0,0,1,1] and Curvature [0,0] (i.e., no curvature).rectangle('Position',[x,y,w,h]) draws the rectangle from the point x,y and having a width of w and a height of h. Specify values in axes data units.Note that, to display a rectangle in the specified proportions, you need to set the axes data aspect ratio so that one unit is of equal length along both the x and y axes. You can do this with the command axis equal or daspect([1,1,1]).rectangle('Curvature',[x,y]) specifies the curvature of the rectangle sides, enabling it to vary from a rectangle to an ellipse. The horizontal curvature x is the fraction of width of the rectangle that is curved along the top and bottom edges.[11]The vertical curvature y is the fraction of the height of the rectangle that is curved along the left and right edges.The values of x and y can range from 0 (no curvature) to 1 (maximum curvature). A value of [0,0] creates a rectangle with square sides. A value of [1,1] creates an

ellipse. If you specify only one value for Curvature, then the same length (in axes data units) is curved along both horizontal and vertical sides. The amount of curvature is determined by the shorter dimension. [11]

rectangle('PropertyName',propertyvalue,...) draws the rectangle using the values for the property name/property value pairs specified and default values for all other properties. For a description of the properties, see Rectangle Properties.h = rectangle(...) returns the handle of the rectangle object created.

2.1.11 serial : Create serial port object

Syntax :

obj = serial('port')

Description :

obj = serial('port') creates a serial port object associated with the serial port specified by port. If port does not exist, or if it is in use, you will not be able to connect the serial port object to the device.

s = serial('COM1');

Port object name will depend upon the platform that the serial port is on. instrhwinfo ('serial') provides a list of available serial ports.

Before you can communicate with the device, it must be connected to obj with the fopen function. A connected serial port object has a Status property value of open. An error is returned if you attempt a read or write operation while the object is not connected to the device. You can connect only one serial port object to a given serial port. [11]

Specifications of a serial connection are given by:

ByteOrder = littleEndian
BytesAvailable = 0
BytesAvailableFcn =
BytesAvailableFcnCount = 48
BytesAvailableFcnMode = terminator
BytesToOutput = 0
ErrorFcn =
InputBufferSize = 512
Name = Serial-COM1
ObjectVisibility = on
OutputBufferSize = 512
OutputEmptyFcn =

RecordDetail = compact
RecordMode = overwrite
RecordName = record.txt
RecordStatus = off
Status = closed
Tag =
Timeout = 10
TimerFcn =
TimerPeriod = 1
TransferStatus = idle
Type = serial
UserData = []
ValuesReceived = 0
ValuesSent = 0

SERIAL specific properties:
BaudRate = 9600
BreakInterruptFcn =
DataBits = 8
DataTerminalReady = on
FlowControl = none
Parity = none
PinStatus = [1x1 struct]
PinStatusFcn =
Port = COM1
ReadAsyncMode = continuous
RequestToSend = on
StopBits = 1
Terminator = LF

2.1.12 fopen : Connect serial port object to device

Syntax :

fopen(obj)

Description :

fopen(obj) connects the serial port object, obj to the device. An error is returned if you attempt to perform a read or write operation while obj is not connected to the device. You can connect only one serial port object to a given device.

Some properties are read-only while the serial port object is open (connected), and must be configured before using fopen. Examples include InputBufferSize and OutputBufferSize. Refer to the property reference pages to determine which properties have this constraint.

The values for some properties are verified only after obj is connected to the device. If any of these properties are incorrectly configured, then an error is returned when fopen is issued and obj is not connected to the device. Properties of this type include BaudRate, and are associated with device settings. [11]

fopen(s)

open : Open file in appropriate application

Syntax :

open(name)

output = open(name)

Description :

open(name) opens the specified file or variable in the appropriate application.
output = open(name) returns an empty output ([]) for most cases. If opening a MAT-file, output is a structure that contains the variables in the file. If opening a figure, output is a handle to that figure. The open function opens files based on their extension. You can extend the functionality of open by defining your own file handling function of the form openxxx, where xxx is a file extension. For example, if you create a function openlog, the open function calls openlog to process any files with the .log extension. The open function returns any single output defined by your function. [11]

2.1.13 fprintf : Write data to text file

Syntax :

fprintf(fileID, format, A, ...)

fprintf(format, A, ...)

Description :

fprintf(fileID, format, A, ...) applies the format to all elements of array A and any additional array arguments in column order, and writes the data to a text file. fprintf uses the encoding scheme specified in the call to fopen. [11]

fprintf(format, A, ...) formats data and displays the results on the screen.

fprintf(s,'s')

2.1.14 fscanf : Read data from a text file

Syntax :

A = fscanf(fileID, format)

A = fscanf(fileID, format, sizeA)

Description :

A = fscanf(fileID, format) reads and converts data from a text file into array A in column order. To convert, fscanf uses the format and the encoding scheme associated with the file. To set the encoding scheme, use fopen. The fscanf function reapplies the format throughout the entire file, and positions the file pointer at the end-of-file marker. If fscanf cannot match the format to the data, it reads only the portion that matches into A and stops processing.[11]

A = fscanf(fileID, format, sizeA) reads sizeA elements into A, and positions the file pointer after the last element read. sizeA can be an integer, or can have the form [m,n].

out = fscanf(s);

2.1.15 wavplay : Play recorded sound on PC-based audio output device.

Syntax :

wavplay(y,Fs)

wavplay(...,'mode')

Description :

wavplay(y,Fs) plays the audio signal stored in the vector y on a PC-based audio output device. Fs is the integer sample rate in Hz (samples per second). The default value for Fs is 11025 Hz. wavplay supports only 1- or 2-channel (mono or stereo) audio signals. To play in stereo, y must be a two-column matrix.[11]

wavplay(...,'mode') specifies how wavplay interacts with the command line, according to the string 'mode'. The string 'mode' can be

'async': You have immediate access to the command line as soon as the sound begins to play on the audio output device (a nonblocking device call).

'sync' (default value): You don't have access to the command line until the sound has finished playing (a blocking device call).

The audio signal y can be one of four data types. The number of bits used to quantize and play back each sample depends on the data type.

The wavplay function is for use only with 32-bit Microsoft Windows operating systems.

3. PROCESSING AND APPLICATION DEVELOPMENT

3.1 Algorithm: (For Red Color Detection)

1. Initialization of capture device and capture format.

   Default Adapter Settings in MATLAB is imaqhwinfo
   Installed Adapters - ('Coreco' , 'Winvideo')
   MATLAB Version - '7.10(r2010a)'
   Toolbox Name – 'Image Acquisition Toolbox'
   Toolbox Version – 3.5(r2011a)

   Camera and Format Initialization
   Vid = videoinput('Winvideo',1,YUY2 640x480)
   Set(vid,'Framepertrigger',inf)
   Set(vid,'Returncolourspace','rgb')
   Framegrabinterval = 5

2. Start video capture.

3. Take an instance/frame

4. Mirror the Image

   Row Values remains the same Column Values of RGB space reversed with respect to input video As Webcam is in our opposite direction,so to synchronize with input hand movement and for better user interface , mirror of image is used for this case.

5. Extract Red Portion.

   R vaue of RGB image-grey scale image gives the red portion
   Imsubstract(RGB image (R value) .greyscale image)

6. Noise reduction using Median filter.

7. Convert to binary image

8. Opening of image (to remove small objects/noises).

   For this bwareaopen function is used

9. Labeling and image blob analysis

   Here Boundary and centroid of the red portion is found out.

10. Display the marked red portion/ Interface to mouse

11. Check If more FrameCount.
    a. If YES, go to step-3
    b. If NO, Stop Video Capture and erase data.

Fig 3.1.a-Input Image Frame

Fig 3.1.b-Mirrored Image

Fig 3.1.c-Red portion extracted

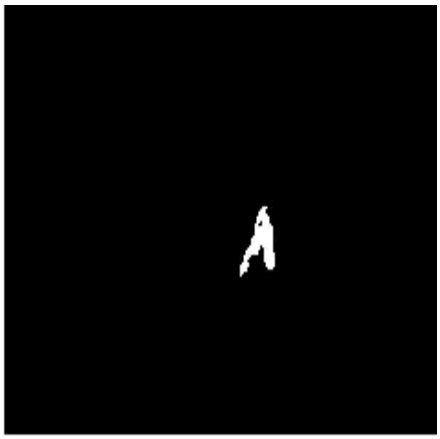


Fig 3.1.d-Median filtered Image

Fig 3.1.e-Binary Image

Fig 3.1.f-Small noise/objects extracted

Fig 3.1.g-Final Output(detected red portion and centroid of the area shown)

This centroid of the detected red portion can be interfaced with mouse. If more than one moving red objects are there, then it may create ambiguity. To eliminate this ambiguity , hand gesture also included in the features. Finally the mirrored image displayed for better user interface; as the camera displays the mirror image. So mirror of mirror image is the original image i.e in accordance with the user. Using other region properties (angle, major axis length, minor axis length) more no of applications can be developed.

3.2 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) % For mouse pointer movement
```

3.2.1 For Left Click-

```
robot.mousePress(java.awt.event.InputEvent.BUTTON1_MASK)
robot.mouseRelease(java.awt.event.InputEvent.BUTTON1_MASK)
```

3.2.2 For Middle Button Click-

```
robot.mousePress(java.awt.event.InputEvent.BUTTON2_MASK)
robot.mouseRelease(java.awt.event.InputEvent.BUTTON2_MASK)
```

3.2.3 For Right Click-

```
robot.mousePress(java.awt.event.InputEvent.BUTTON3_MASK)
robot.mouseRelease(java.awt.event.InputEvent.BUTTON3_MASK)
```

3.3 Keyboard Interfacing:

Similarly java.awt.Robot class can also be used for keyboard interfacing.

For pressing 'D', following commands need to be given

```
robot.keyPress(KeyEvent.VK_D);
```

robot.keyRelease(KeyEvent.VK_D);

For ‘Blankspace’ robot.keyPress(KeyEvent.VK_SPACE);
robot.keyRelease(KeyEvent.VK_SPACE);

Any key can be entered in this way.

3.4 Algorithm for Hand Extraction And Finger Count :

Method-1: (In this method, it is needed to wear full-sleeve black shirt)

1. Initialization of capture device and capture format.
2. Start video capture.
3. Initialize Background , Mirror it and save mirrored background image
4. Take an instance/frame
5. Mirror the Image
6. Subtract the mirrored image from the saved mirrored initial background.
7. Noise reduction using Median filter.
8. Convert to binary image
9. Opening of image (to remove small objects/noises).
10. Display only the extracted Image
11. Hand Extraction by avoiding the black pixels (perform Step-6-9 again)
12. Labeling and image blob analysis using region properties.
13. Display the marked red portion/ Interface to mouse
14. Check If more FrameCount.
    a. If YES, go to step-4
    b. If NO, Stop Video Capture and erase data

Method-2: (No need of Black-shirt. Any color will work.)

1. Initialization of capture device and capture format.
2. Start video capture.
3. Initialize Background , Mirror it and save mirrored background image
4. Take an instance/frame

5. Mirror the Image
6. Subtract the mirrored image from the saved mirrored initial background.
7. Noise reduction using Median filter.
8. Convert to binary image
9. Opening of image (to remove small objects/noises).
10. Labeling and image blob analysis using region properties.
11. Using boundary properties, check change in slope of contours in anti-clockwise direction.
12. If Slope change from negative to positive, bottom gap of finger.
    If slope change from positive to negative, peak of a finger.
13. No. of peaks gives the no of fingers counted.
14. Depending on the count, a program is activated.
15. Interfacing the hand with mouse, any application can be used.
16. Check If more FrameCount.
    a. If YES, go to step-4
    b. If NO, Stop Video Capture and erase data

Fig 3.2a-i Fig 3.2a-ii Fig 3.2a-iii Fig3.2a-iv Fig3.2a-v Fig3.2a-vi

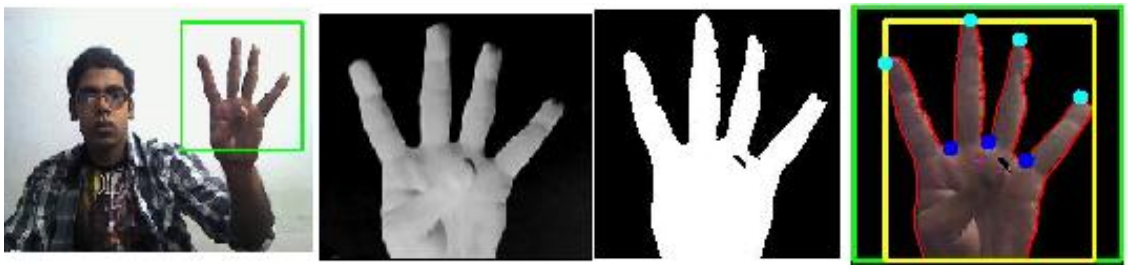


Fig3.2b-i Fig3.2b-iii Fig3.2b-v Fig3.2b-vi

Fig3.2c-i Fig3.2c-iii Fig3.2c-v Fig3.2c-vi

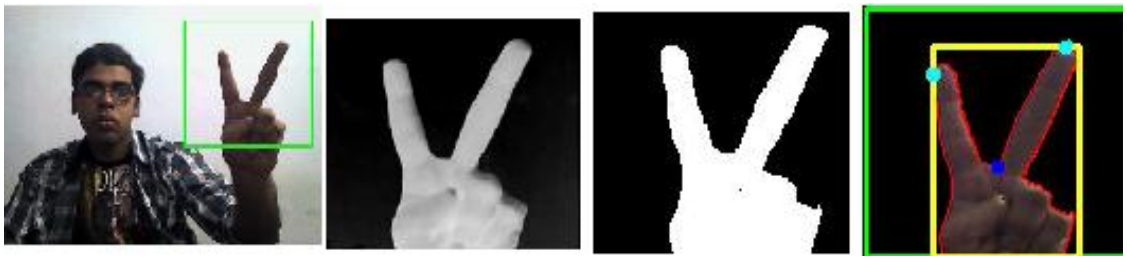


Fig3.2d-i Fig3.2d-iii Fig3.2d-v Fig3.2d-vi

[Fig3.2a,b,c,d- No of fingers 5,4,3,2 respectively]

[i-Input Image, ii-Background eliminated image, iii-Filtered Image, iv-Binary Image, v-Opening Image, vi-Final Image with Finger peaks and gaps detected]

3.5 Virtual Calculator :

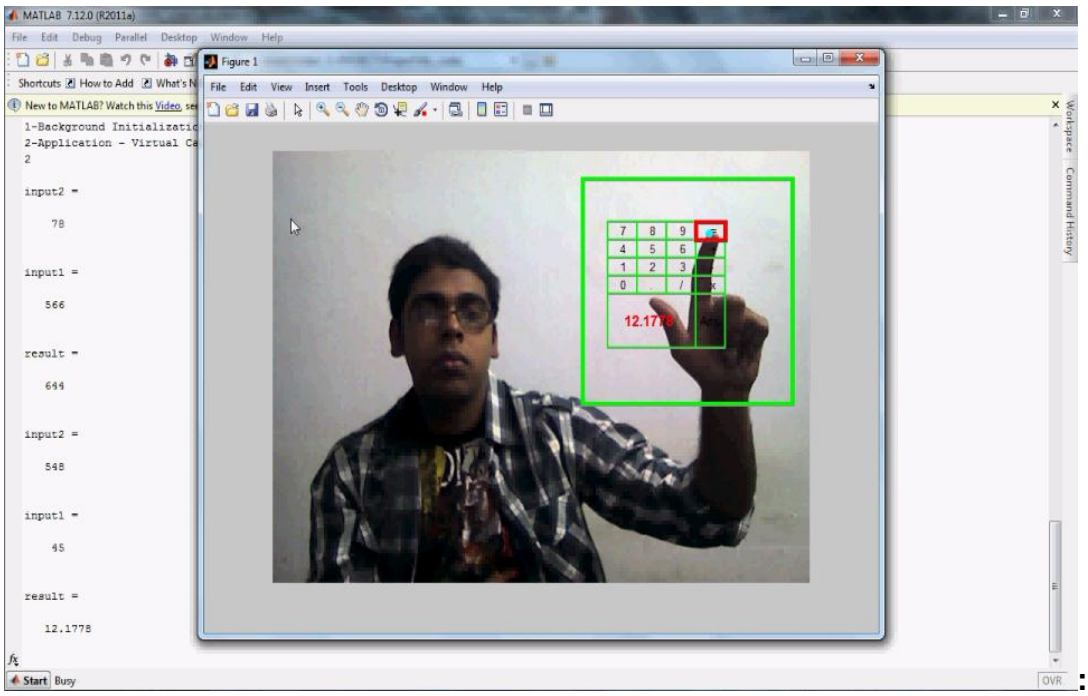


Fig 3.3.a-Virtual Calculator

Virtual Calculator is like an augmented reality. It is designed in Matlab in the same interfacing program. Depending on the position of finger- peak of Master hand, application

gives input to the calculator. It checks for 5 frames. If for 5 frames, the finger-peak remain in a particular region or number region, then that number or symbol is given as input.

Checking for '5'

if(point_y > 437 && point_y < 468)

And

if(point_x > 102 && point_x < 118)

Then check if it is in the region for

Five continuous frame

If Yes, take 5 as input and reset all Counter.

If No, continue process.

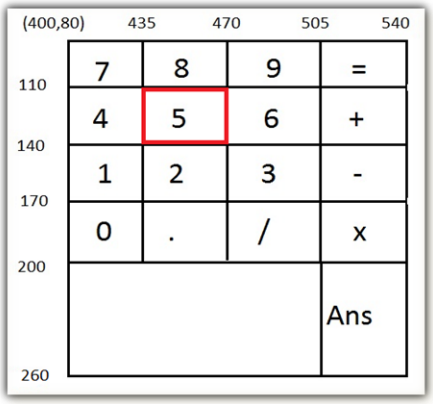


Fig 3.3.b- Calculator design

3.6 PC Calculator:

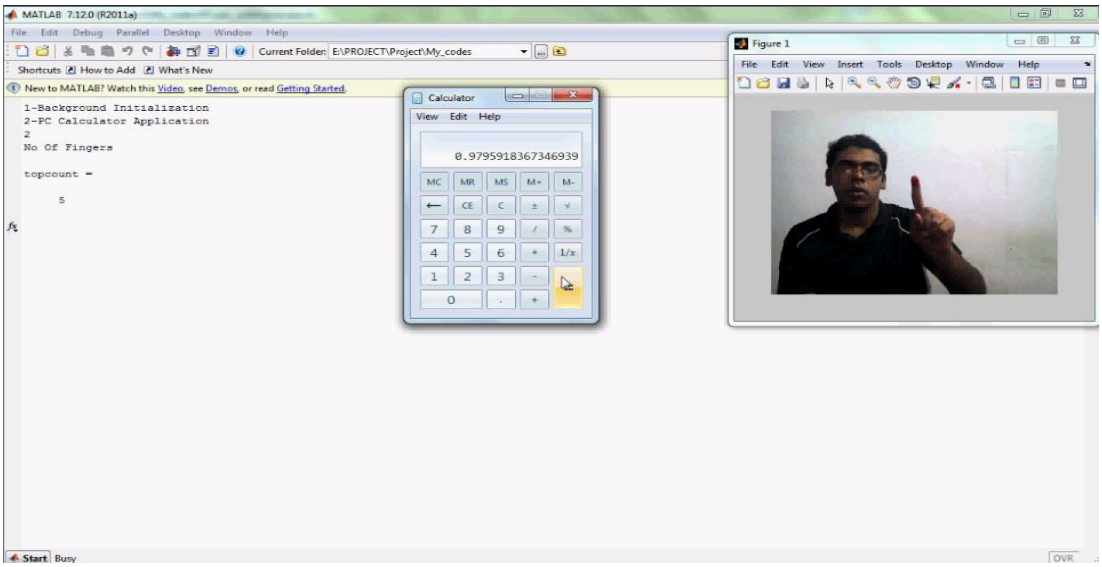


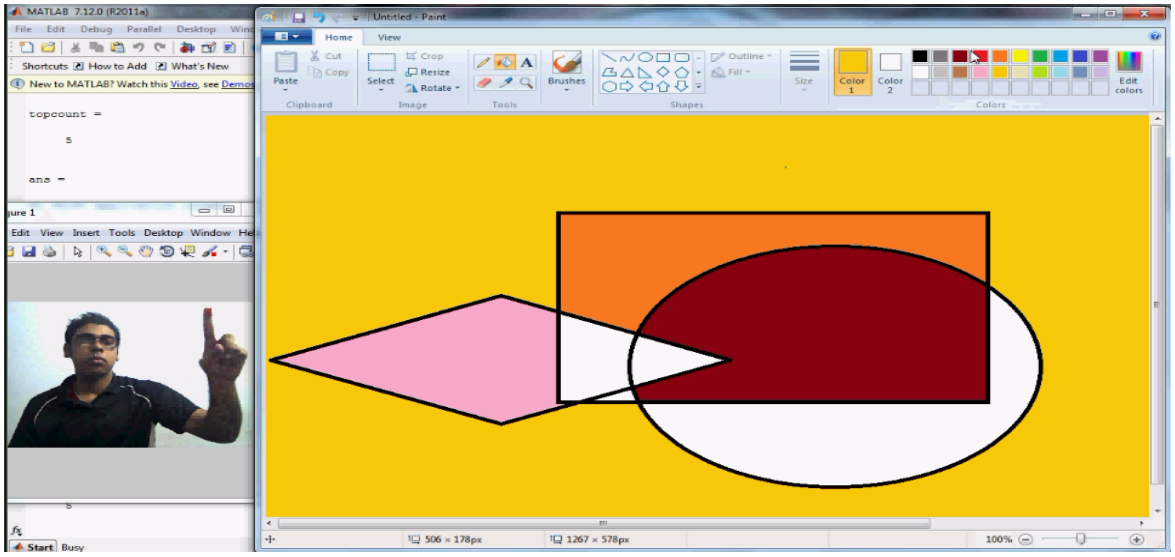
Fig 3.3.c -PC Calculator

PC Calculator is the original calculator application given in windows platform by Microsoft. In PC Calculator, finger-peak of Master hand is interfaced with mouse cursor. Ifthe mouse cursor remains within +-20 pixels for 5 frames, then Left Click event activated by java Robot class and if the cursor remains within +-20 pixels for 8 frames, then Right Click event activated by java Robot class. By this method, input is given to the calculator.

If tip of hand(tip_x,tip_y) remains within +- 20 pixels for five continuous frames, then trigger "Left Click" by using java.awt.Robot class

```
robot = java.awt.Robot;
robot.mousePress(java.awt.event.InputEvent.BUTTON1_MASK);
pause(0.1)
robot.mouseRelease(java.awt.event.InputEvent.BUTTON1_MASK)
```

3.7 PC Paint :



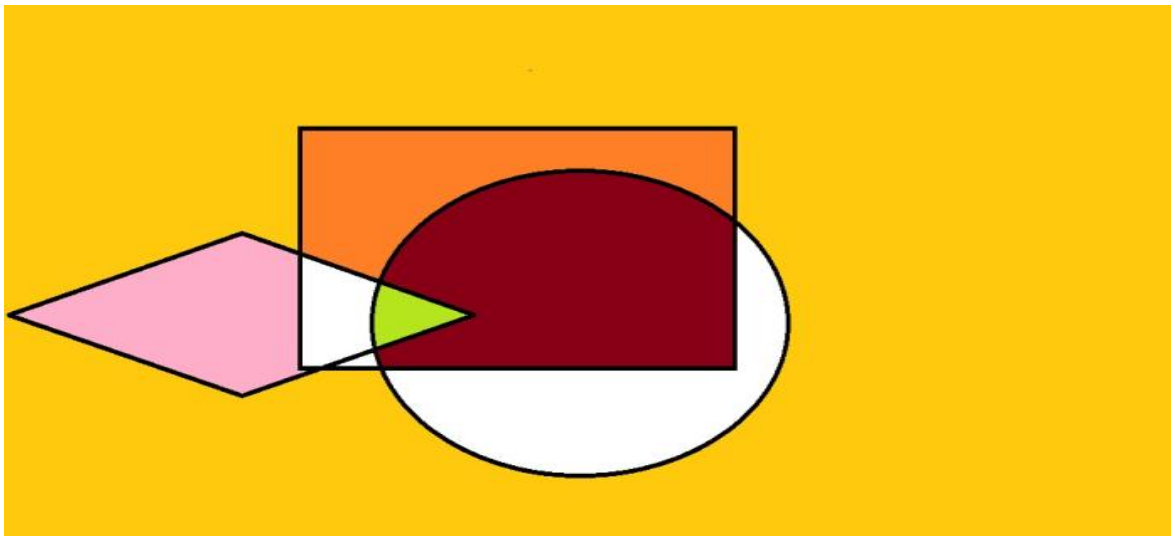


Fig 3.4.a- PC Paint Snapshot Fig 3.4.b- Final Paint

Above paint is done with the help of hand and without using mouse. In above application, two color pins used (Red, Blue). Red is used for left click and cursor movement and blue color is used for dragging. If the cursor remains within +-20 pixels for 5 continuous frames, then Left click gets triggered. When the blue color code is shown, the left press is triggered until the blue color code vanishes. When the blue color code vanishes, the left

button release triggerd. Hereby, it works as dragging event. Fig 3.4.b shows the final saved paint file. Like this any PC application can be interfaced.

3.8 HCI(Human-Computer Interface) :

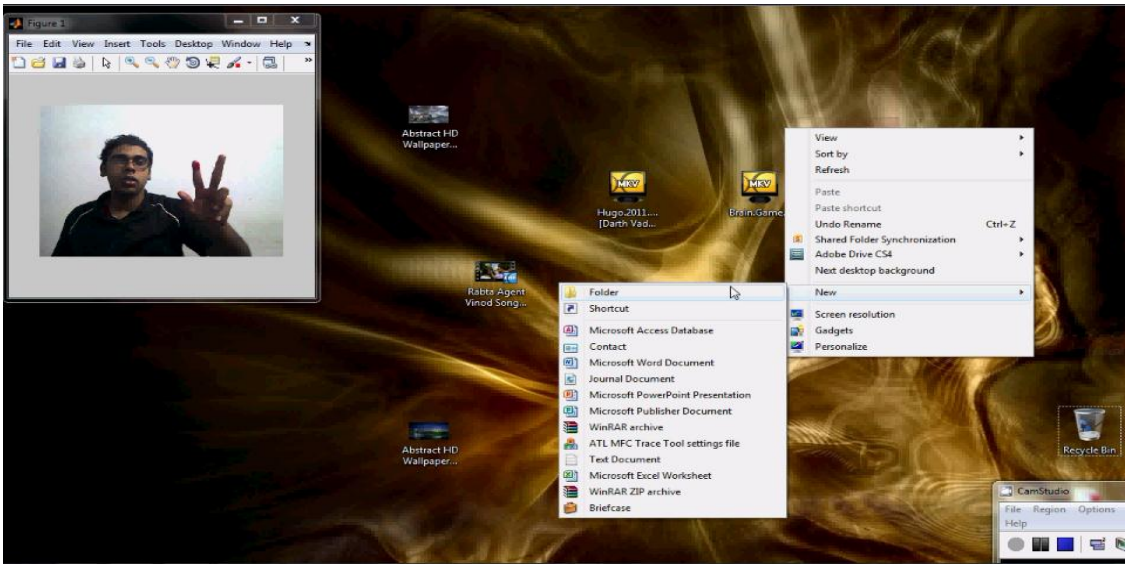


Fig 3.5 HCI snapshot

In above application, Master-Hand has been interfaced with mouse cursor. It can perform Left click, Right Click, Double Left Click with a precision of 20 pixels.

If tip of hand(tip_x,tip_y) remains within +- 20 pixels for five continuous frames, then trigger “Left Click” by using java.awt.Robot class

robot = java.awt.Robot;

robot.mousePress(java.awt.event.InputEvent.BUTTON1_MASK);

pause(0.1)

robot.mouseRelease(java.awt.event.InputEvent.BUTTON1_MASK);

If tip of hand(tip_x,tip_y) remains within +- 20 pixels for eight continuous frames, then trigger “Double Left Click” by using java.awt.Robot class

robot = java.awt.Robot;

robot.mousePress(java.awt.event.InputEvent.BUTTON1_MASK);pause(0.1)
robot.mouseRelease(java.awt.event.InputEvent.BUTTON1_MASK);pause(0.1)

robot.mousePress(java.awt.event.InputEvent.BUTTON1_MASK); pause(0.1)

robot.mouseRelease(java.awt.event.InputEvent.BUTTON1_MASK);

3.9 Remote Controlling Using Cloud :

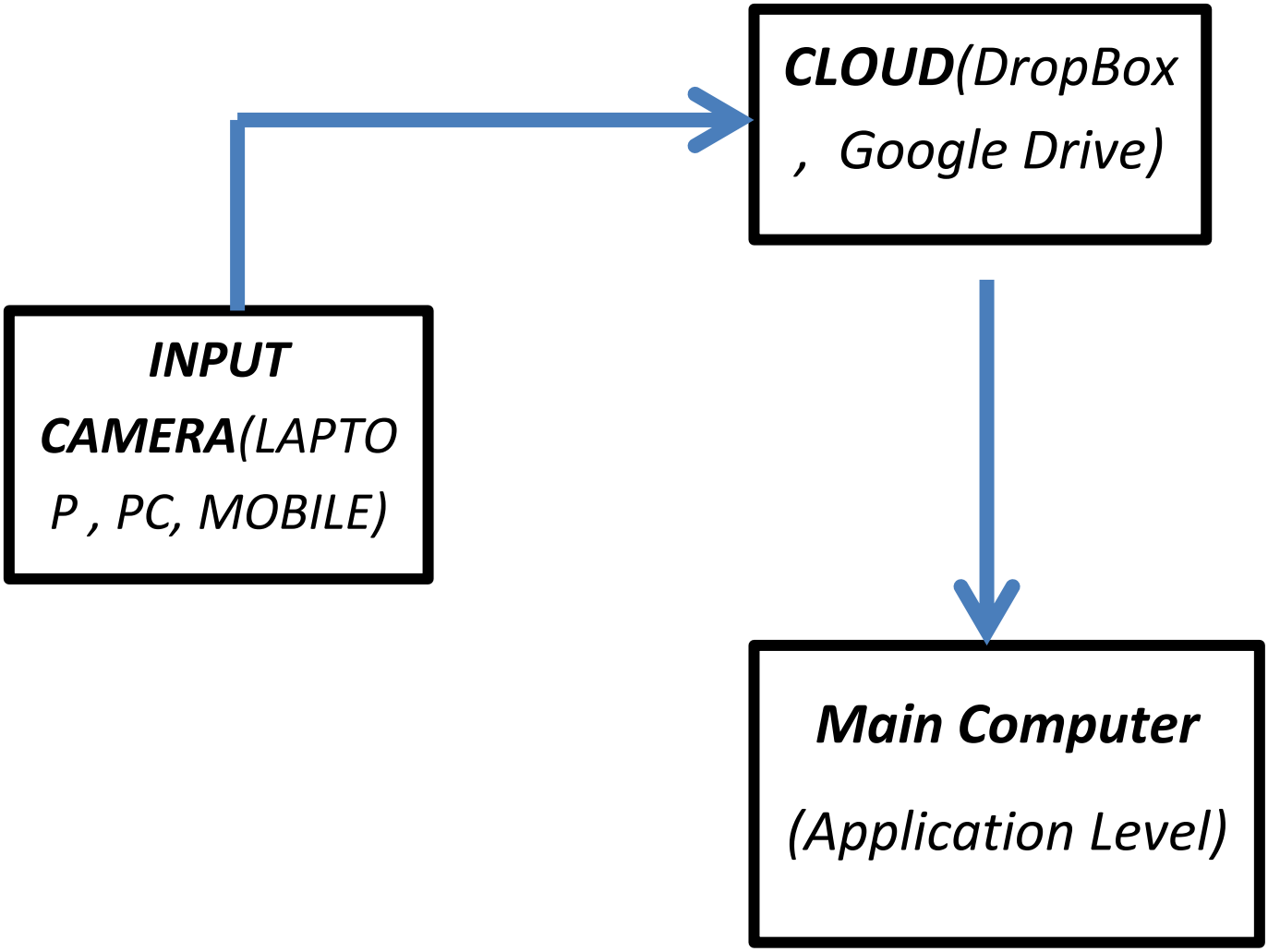


Fig 3.6 Remote Controlling Application

Must have Internet connection both at client side and server side.DROPBOX used as cloud Hand gesture captured through any camera (Mobile or PC or Laptop), then stored at a particular location giving a particular name.

At server side PC, the image gets updated automatically where matlab program uses the above saved image and image procesisng done to recognize the gesture. Then it checks the corresponding command for the gesture and acts accordingly.

For better user Interface, a small application can be developed in Visual Studio or Android platform. Its work is only to capture hand gesture and store in dropbox.

It helps in controlling PC(activating/deactivating any application mainly) from any where in the world only that you should have a camera and internet connection.

3.10 Gesture based Hardware Interfacing Applications:

1. Remote control of home-window

Let's take a case, somebody has left his home window opened. When he reaches office, he notices that it is going to rain. Then he recalls leaving home window opened. There is nobody at home . At this condition, he would have to run back to home and close the window otherwise all household objects will get wet and few may get damaged. But using this application, he will just have to take his mobile out , take a snapshot of a predefined hand-gesture(i.e. linked to closing the window) and save it in dropbox.

Then at home, the dropbox or any cloud app will automatically get updated. If a motor can be interfaced with the PC, then giving command from PC, the window can be closed. Here serial communication is used. Atmega640 Development board used for the purpose.

3.10.1 Hardware-

1) Atmega640 Development Board
2) Program Burner(ISP)
3) Motor
4) Adapter
5) Connector

3.10.2 Software-

1) AVR Studio
2) USB-Serial Driver
3) Winavr
4) X-CTU Configuration & Test Utility Software
5) Dropbox/ google drive/ any cloud application

3.10.3 ATMEGA 640 Features-

- High Performance, Low Power AVR® 8-Bit Microcontroller
- 32 x 8 General Purpose Working Registers
- Up to 16 MIPS Throughput at 16 MHz
- 64K Bytes of In-System Self-Programmable Flash
- In-System Programming by On-chip Boot Program
- 4K Bytes EEPROM

- 8K Bytes Internal SRAM
- Up to 64K Bytes Optional External Memory Space
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
- Four 8-bit PWM Channels
- Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits
- 16-channel, 10-bit ADC
- Two/Four Programmable Serial USART
- Master/Slave SPI Serial Interface
- Byte Oriented 2-wire Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

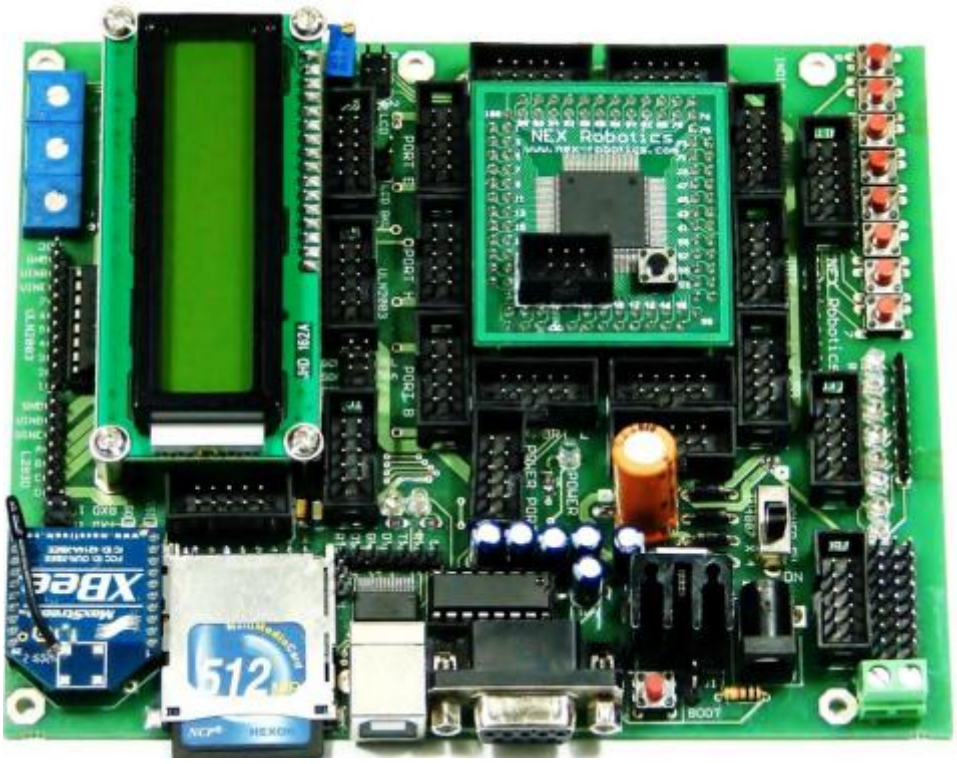


Fig 3.7 ATmega 640 Development Board[12]

3.10.4 Program-

UART Program-

```
#include<avr/io.h>
#include<util/delay.h>
```

```
/*Macros definition*/
#define CHECKBIT(x,b) x&(1<<b)     //Checks bit status
#define SETBIT(x,b) x|=(1<<b)      //Sets the particular bit
#define CLEARBIT(x,b) x&=~(1<<b) //Sets the particular bit
#define TOGGLEBIT(x,b) x^=(1<<b) //Toggles the particular bit
void uart0_init(void)
{
UCSR0B = 0x18;
UCSR0C = 0x86;
UBRR0L = 95;
}
void send0(char data)
{
UDR0 = data;
while(!(CHECKBIT(UCSR0A,UDRE0)))
{
}
_delay_ms(10);
}
char recv0(void)
{
while(!(CHECKBIT(UCSR0A,RXC0)))
{
}
_delay_ms(5);
char atad = UDR0;
return (atad);

void uart1_init(void)
{  UCSR1A = 0x00;
UCSR1B = 0x18;
UCSR1C = 0x86;
UBRR1L = 95;
}
void send1(char data)
{
UDR1 = data;
while(!(CHECKBIT(UCSR1A,UDRE1)))
{
}
_delay_ms(10);
}
char recv1(void)
{
while(!(CHECKBIT(UCSR1A,RXC1)))
{
}
_delay_ms(5);
char atad = UDR1;
return (atad);
}
void uart2_init(void)
{
UCSR2B = 0x18;
UCSR2C = 0x86;
```

```
UBRR2L = 95;
}
void send2(char data)
{
UDR2 = data;
while(!(CHECKBIT(UCSR2A,UDRE2)))
{
}
_delay_ms(0);
}
char recv2(void)
{
while(!(CHECKBIT(UCSR2A,RXC2)))
{
}
_delay_ms(0);
char atad = UDR2;
return (atad);
}
void uart_init()
{
uart0_init();
uart1_init();
uart2_init();
}
```

LCD Display Program-

```
#include<avr/io.h>
#include<util/delay.h>
#define P1 PORTC   //data port n controller
#define P3 DDRC  //data port
#define rs 4
#define e  5
#define LCD_EN 1<<e
#define LCD_RS 1<<rs

                    void delay(int a,int b)
                    {
                  for(;b>0;b--)
                      _delay_ms(a);
                    }

          void port()
{

P3=0xFF;P1=0x00;
}
                    void data1 (unsigned char dat)
{
P1 = (((dat >> 4) & 0x0F)|LCD_EN|LCD_RS);
P1 = (((dat >> 4) & 0x0F)|LCD_RS);

P1 = ((dat & 0x0F)|LCD_EN|LCD_RS);
```

```
P1 = ((dat & 0x0F)|LCD_RS);

_delay_us(200);
_delay_us(200);
}
                void instruction(char cmd)
{
P1= ((cmd >> 4) & 0x0F)|LCD_EN;
P1 = ((cmd >> 4) & 0x0F);

P1 = (cmd & 0x0F)|LCD_EN;
P1 = (cmd & 0x0F);

_delay_us(200);
_delay_us(200);
}

                void lcd_cursor(short int r,short int c)          //sets position of the cursor to
the Rth roe and Cth column
                {
                  if(r==1)
                {  instruction(127+c);
                   }
                  if(r==2)
                   { instruction(191+c);
                   }
                }
                void character(char c)
                {
                   data1(c);

                }
                void lcd_string(char c[])
                {
                                    while(*c)
                   {

                      character(*c++);
                    }

                }

                void digit(int a)
                {
                  data1(48+a);
                }

                void lcd_number(int a,int b,int num,int n)
                {  lcd_cursor(a,b);
                   int i=b+n-1;

              while(num)
               { lcd_cursor(a,i--);
                 digit(num%10);
                 num/=10;
```

```
}
            while(i>=b)
            {
lcd_cursor(a,i--);
digit(0);}
                        }
                         void number2(float n,int p)
                        {    int a,c;
                             a=n;
                             n=n-a;
                            for(c=1;c<=p;c++)
                             n=n*10;
                        }

void clr()
{
instruction(0x01);
}
void reset()
{
P1= 0xFF;
_delay_ms(20);
P1 = 0x03+LCD_EN;
P1 = 0x03;
_delay_ms(10);
P1 = 0x03+LCD_EN;
P1 = 0x03;
_delay_ms(1);
P1 = 0x03+LCD_EN;
P1 = 0x03;
_delay_ms(1);
P1 = 0x02+LCD_EN;
P1 = 0x02;
_delay_ms(1);
}
                    void initiate()
{port();
reset(); // Call LCD reset
_delay_us(400);
instruction(0x28); // 4-bit mode - 2 line - 5x7 font.
instruction(0x0C); // Display no cursor - no blink.
instruction(0x06); // Automatic Increment - No Display shift.
instruction(0x80); // Address DDRAM with 0 offset 80h.
}

void clear()
{
lcd_cursor(1,1);
lcd_string("               ");

lcd_cursor(2,1);
lcd_string("               ");
lcd_cursor(1,1);
}
```

Program for Motor Control-
(To close the window)

```
#include "lcd4.c"     //lcd include
#include"uart (2).c"
#include<avr/io.h>
int HIGH;
int data;
void go()
{

PORTA|=0b00000100;
PORTA&=0b11011111;

}
void stop()
{
PORTA&=0b11011011;
}
void back()
{
PORTA|=0b01000000;
PORTA&=0b01111111;

}
void fwd()
{
PORTA&=0b00111111;

}
void main()
{
initiate();
   uart_init();  //serial port configs
   DDRA=0xFF;PORTA=0xFF; // Relay control to l293d
lcd_string("lplp");
while(1)
 {

            data=recv0();
                    if(data == 'e')
                    {         lcd_cursor(2,5);lcd_string("stop ");
          stop();

          }
                    if(data == 's')
                    {         lcd_cursor(2,5);lcd_string("forward");
go();
          }

 }

}
```

Hand gesture captured through any camera (Mobile or PC or Laptop), then stored at a particular location giving a particular name. At server side PC, the image gets updated automatically where matlab program uses the above saved image and image procesisng done to recognize the gesture. Then it checks the corresponding command for the gesture and acts accordingly. The motor is connected to computer via USB-Serial Connector through Atmega640. When the respective gesture for window-dooring closing detected, matlab activates serial port and start signal send to Atmega. Then the motor runs i.e connected to the Atmega. When the the stop signal is sent, motor stops. By this way, the motor, closes the window-door.

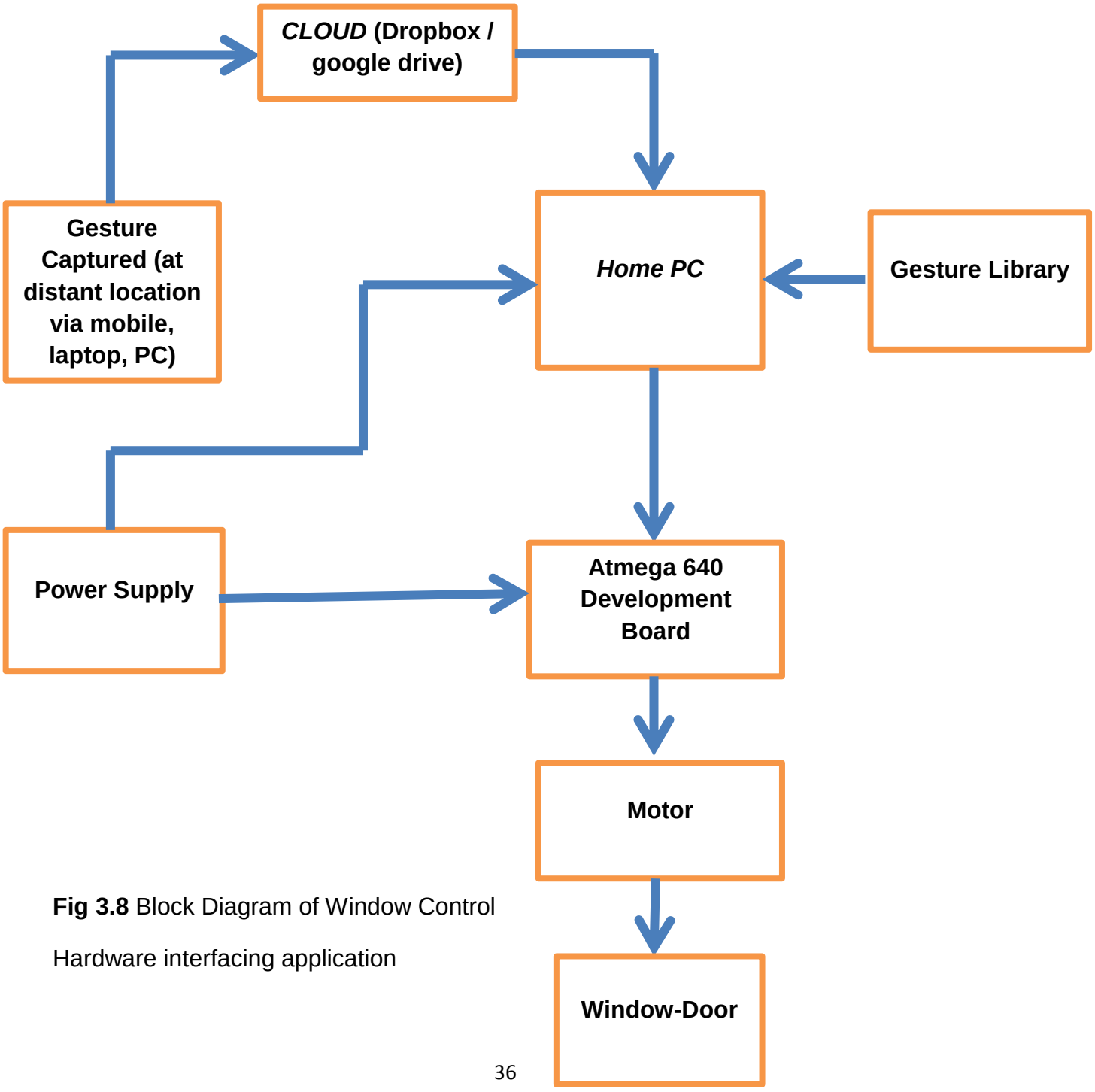


Fig 3.8 Block Diagram of Window Control Hardware interfacing application

4. CONCLUSION AND FUTURE WORK

4.1 CONCLUSION:

- The proposed technique is simple , efficient and faster (SEF).
- In simple case, using even only 5 hand gestures, if combined with colors (say 5) gives 5x5! no of combinations. So 600 no of commands can be given through image processing.
- In completed case, if two levels of colour used of 5 different types, then total no of combinations can be 5x5!x5! = 72,000 which is way enough for use in applications.
- If both hand used, then the no of combinations increases exponentially.
- Using different colours 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
- Using the cloud feature the hand gesture command can be sent to the main computer from a faraway area.
- In Hardware interfacing applications , many application can be developed in a similar way like emergency window closing application. This can have a vast utilisation in future smart homes.

4.2 FUTURE WORKS:

- Hand Gesture recognition can be optimized using developed techniques
- Instead of using RGB space , HSV colour space can be used to avoid light intensity fluctuations and thereby giving better results.
- HAND can be used as a remote control to TV if subject to a camera is embedded.
- Many entertainment applications can be developed e.g. virtual piano playing , video game etc.
- 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]. http://technologyinterface.nmsu.edu/5_1/5_1.html

[10]. http://www.embedded-vision.com/applications-embedded-vision

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

[12]. www.alldatasheet.com

[13]. http://www.digi.com/support/productdetl.jsp?pid=4091&osvid=62&tp=5&hit=XCTU%20ver.%205.1.4.1%20installer#utilities

[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]. Rautaray, S.S., "Interaction with Virtual Game through Hand Gesture Recognition ", Multimedia, Signal Processing and Communication Technologies (IMPACT), 2011 International Conference, Page(s): 244 - 247

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

[18]. Tomita A., Ishii R., "hand Shape Extraction from a Sequence of digitized gray-scale Images", IEEE Journals

[19]. Abe, K., Saito, H. and Ozawa, S. (2000). 3-D Drawing System via Hand Motion Recognition from Two Cameras. In Proceeding of the 6th Korea-Japan Joint Workshop on Computer Vision, pp. 138-143.

[20]. Assan, M. and Grobel, K. (1997). Video Based Sign language Recognition using Hidden Markov Models. Gesture and Sign Language in Human-Computer Interaction, Intl. In Proc. of Gesture Workshop, vol. 1371 of Lecture Notes in Computer Science, pp. 97-110.

[21]. Bowden, R., Heap, A., and Hart, C. (1996). Virtual Datagloves: Interacting with Virtual Environments Through Computer Vision. In Proc. 3rd UK VR-Sig Conference, DeMontfort University, Leicester, UK, July 1996.

[22]. Crowley, J., Brard, F. and Coutaz, J. (1995). Finger tracking as an input device for augmented reality. In Proc. Workshop Automatic Face and Gesture Recognition, pp. 195-200.

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

[24]. Kurata, T., Kato, T., Kourogi, M., Keechul, J., and Endo, K. (2002). A functionally-distributed hand tracking method for wearable visual interfaces and its applications. In IAPR Workshop on Machine Vision Applications, pp. 84-89.

[25]. Kolsch, M. and Turk, M. (2005). Hand tracking with Flocks of Features. Computer Vision and Pattern Recognition, CVPR, vol. 2, 20-25.

[26]. Lee, J. and Kunii, T. (1995). Model-based analysis of hand posture. IEEE Comput. Graph. Appl., vol 15. no. 5, pp. 77-86.

THE END