

SIMPLE GESTURE RECOGNITION USING OPENCV

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

PASUPULETI EESHA VENKAT (37110549)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

**(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PASUPULETI EESHA VENKAT (37110549)** who carried out the project entitled "**SIMPLE GESTURE RECOGNITION USING OPENCV**" under our supervision from **November 2020 to April 2021**.

Internal Guide

Dr. S.PRAYLA SHRY M.E., Ph.D.,

Head of the Department

Submitted for Viva-Voce Examination held on:

Internal Examiner

External Examiner

DECLARATION

I **PASUPULETI EESHA VENKAT (37110549)**, hereby declare that the Project Report entitled "**SIMPLE GESTURE RECOGNITION USING OPENCV**" done by me under the guidance of **Dr. S.PRAYLA SHRY M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D.**, Dean, School Of Computing, **Dr. S. VIGNESHWARI M.E., Ph.D.**, and **Dr. L. LAKSHMANAN M.E., Ph.D.**, Heads Of the Department, Department of Computer Science and Engineering for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S.PRAYLA SHRY M.E., Ph.D.**,

for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express our thanks to all Teaching and Non-teaching staff of the **Department of Computer Science & Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The implementation of opencv in python for a gesture recognition system for the functionalities of the mouse. Object recognition techniques are used for extraction of information from the capture. We do the write operation, drag and drop operations along with other mouse actions. This will be helpful for the teaching procedure involving projectors. This will overcome the restrictions a corporeal device has would enrich the experience of interaction to the users. The technique of establishing a process of interaction between human and computer is evolving since the invention of computer technology. The mouse is an excellent invention in HCI (Human-Computer Interaction) technology. Though wireless or Bluetooth mouse technology is invented still, that technology is not completely device free. A Bluetooth mouse has the requirement of battery power and connecting dongle. Presence of extra devices in a mouse increases the difficulty to use it. The proposed mouse system is beyond this limitation. This paper proposes a virtual mouse system based on HCI using computer vision and hand gestures. Gestures captured with a built-in camera or webcam and processed with colour segmentation & detection technique. The user will be allowed to control some of the computer cursor functions with their hands which bear coloured caps on fingertips. Primarily, a user can perform left clicks, right clicks, and double clicks, scrolling up or down using their hand in different gestures. This system captures frames using a webcam or built-in cam and processes the frames to make them track-able and after that recognizes different gestures made by users and perform the mouse function. So the proposed mouse system eliminates device dependency in order to use a mouse. Therefore it can be proved beneficial in order to develop HCI technology. The approach we use will provide us with a better way of teaching through projectors and also we can do simple mouse operations without using the mouse (virtually). This helps in presentations and seminars very well.

TABLES OF CONTENTS

SNo	CHAPTER NAME	PAGENO
	ABSTRACT	v
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
1	INTRODUCTION	1
	1.1 Gesture recognition and pen computing	2
	1.2 Gesture only interfaces	2
	1.3 Tracking Technologies	3
	1.4 Natural gesture only interfaces	3
	1.5 Gesture Based Interaction	4
	1.6 Gesture Types	6
2	LITERATURE SURVEY	7
	2.1 Literature Survey	7
	2.2 Existing System	10
	2.3 Proposed System	11
	2.4 Algorithm	12
3	METHODOLOGY	14
	3.1 Methodology	14
	3.2 Requirements	14
	3.3 Getting Started	15
	3.4 Integration of Opencv with Dev c++	15
	3.5 Configuring included files	17
	3.6 Configuring Static library files	18
	3.7 Configuring Dynamic library files	18
	3.8 Numpy introduction	19
	3.8.1 Operations using numpy	19
	3.8.2 Numpy- A Replacement for Matlab	19
	3.8.3 Numpy- Environment	19

SNo	CHAPTER NAME	PAGENO
	3.9 Numpy-ndarray object	21
	3.10 Numpy-Datatypes	23
	3.11 WxPython- introduction	26
	3.12 Implementation	26
4	RESULTS AND DISCUSSION	29
	4.1 Virtual Mouse	29
	4.2 Removes the requirement of having a physical Mouse	29
	4.3 Operations	29
5	CONCLUSION AND FUTURE WORK	30
	5.1 CONCLUSION	30
	5.2 FUTURE WORK	30
	REFERENCES	31
	APPENDIX	32
	SCREENSHOTS	37

LIST OF TABLES

SNo	TABLE NAME	PAGENO
2.1	Literature Survey	8
3.1	Numpy objects	23
3.2	Numpy datatypes	24

LIST OF FIGURES

FIGURE No	FIGURE NAME	PAGE No
3.1	Tools option	16
3.2	Compiler options	16
3.3	Configuring included files	17
3.4	Configuring Static library	18
3.5	Configuring Dynamic library	18
3.6	Ndarray object	22
3.7	Phases	27
3.8	Calibration and recognition phases	28
5.1	Desktop view of clicking operation	37
5.2	View from webcam	37
5.3	Click operation view from webcam	38

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 the face and hand gesture recognition. Many approaches have been made using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of posture, gait, proxemics, and human behaviour is also the subject of gesture recognition techniques.

Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse.

Gesture recognition enables humans to interface 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 potentially make conventional input devices such as mouse, keyboards and even touch-screens redundant.

Gesture recognition can be conducted with techniques from computer vision and image processing. Interface with computers using gestures of the human body, typically hand movements. In gesture recognition technology, a camera reads the movements of the human body and communicates the data to a computer that uses the gestures as input to control devices or applications. For example, a person clapping his hands together in front of a camera can produce the sound of cymbals being crashed together when the gesture is fed through a computer.

One way gesture recognition is being used is to help the physically impaired to interact with computers, such as interpreting sign language. The technology also has the potential to change the way users interact with computers by eliminating input devices such as joysticks, mice and keyboards and allowing the unencumbered body to give signals to the computer through gestures such as finger pointing.

Unlike haptic interfaces, gesture recognition does not require the user to wear any special equipment or attach any devices to the body. The gestures of the body are read by a camera instead of sensors attached to a device such as a data glove. In addition to hand and body movement, gesture recognition technology also can be used to read facial and speech expressions (i.e., lip reading), and eye movements.

The literature includes ongoing work in the computer vision field on capturing gestures or more general human pose and movements by cameras connected to a computer.

GESTURE RECOGNITION AND PEN COMPUTING

In some literature, the term gesture recognition has been used to refer more narrowly to non-text-input handwriting symbols, such as inking on a graphics tablet, multi-touch gestures, and mouse gesture recognition. This is computer interaction through the drawing of symbols with a pointing device cursor (see discussion at Pen computing).

GESTURE ONLY INTERFACES

The gestural equivalent of direct manipulation interfaces is those which use gesture alone. These can range from interfaces that recognize a few symbolic gestures to those that implement fully fledged sign language interpretation. Similarly interfaces may recognize static hand poses, or dynamic hand motion, or a combination of both. In all cases each gesture has an unambiguous semantic meaning associated with it that can be used in the interface. In this section we will first briefly review the technology used to capture gesture

input, then describe examples from symbolic and sign language recognition. Finally we summarize the lessons learned from these interfaces and provide some recommendations for designing gesture only applications.

TRACKING TECHNOLOGIES

Gesture-only interfaces with syntax of many gestures typically require precise hand pose tracking. A common technique is to instrument the hand with a glove which is equipped with a number of sensors which provide information about hand position, orientation, and flex of the fingers. The first commercially available hand tracker, the Data glove, is described in Zimmerman, Lanier, Blanchard, Bryson and Harvill (1987), and illustrated in the video by Zacharey, G. (1987). This uses thin fiber optic cables running down the back of each hand, each with a small crack in it. Light is shone down the cable so when the fingers are bent light leaks out through the cracks. Measuring light loss gives an accurate reading of hand pose. The Dataglove could measure each joint bend to an accuracy of 5 to 10 degrees (Wise et. al. 1990), but not the sideways movement of the fingers (finger abduction). However, the CyberGlove developed by Kramer (Kramer 89) uses strain gauges placed between the fingers to measure abduction as well as more accurate bend sensing (Figure XX). Since the development of the Dataglove and Cyberglove many other gloves based input devices have appeared as described by Sturman and Zeltzer (1994).

NATURAL GESTURE ONLY INTERFACES

At the simplest level, effective gesture interfaces can be developed which respond to natural gestures, especially dynamic hand motion. An early example is the Theramin, an electronic musical instrument from the 1920's. This responds to hand position using two proximity sensors, one vertical, the other horizontal. Proximity to the vertical sensor controls the music pitch, to the horizontal one, loudness. What is amazing is that music can be made with orthogonal control of the two prime dimensions, using a control system that provides no fixed reference points, such as frets or mechanical feedback. The hands work in extremely subtle ways to articulate steps in what is actually a continuous control space. The Theramin is successful

because there is a direct mapping of hand motion to continuous feedback, enabling the user to quickly build a mental model of how to use the device.

GESTURE BASED INTERACTION

The CyberGlove captures the position and movement of the fingers and wrist. It has up to 22 sensors, including three bend sensors (including the distal joints) on each finger, four abduction sensors, plus sensors measuring thumb crossover, palm arch, wrist flexion and wrist abduction. (Photo: Virtual Technologies, Inc.)

Once hand pose data has been captured by the gloves, gestures can be recognized using a number of different techniques. Neural network approaches or statistical template matching is commonly used to identify static hand poses, often achieving accuracy rates of better than 95% (Väänänen and Böhm 1993). Time dependent neural networks may also be used for dynamic gesture recognition [REF], although a more common approach is to use Hidden Markov Models. With this technique Kobayashi is able to achieve an accuracy of XX% (Kobayashi et. al. 1997), similar results have been reported by XXXX and XXXX. Hidden Markov Models may also be used to interactively segment out glove input into individual gestures for recognition and perform online learning of new gestures (Lee 1996). In these cases gestures are typically recognized using pre-trained templates; however gloves can also be used to identify natural or untrained gestures. Wexelblat uses a top down and bottom up approach to recognize natural gestural features such as finger curvature and hand orientation, and temporal integration to produce frames describing complete gestures (Wexelblat 1995). These frames can then be passed to higher level functions for further interpretation.

Although instrumented gloves provide very accurate results they are expensive and encumbering. Computer vision techniques can also be used for gesture recognition overcoming some of these limitations. A good review of vision based gesture recognition is provided by Palovic et. al. (1995). In general, vision based systems are more natural to use than glove interfaces, and are capable of excellent hand and body tracking,

but do not provide the same accuracy in pose determination. However for many applications this may not be important. Sturman and Zeltzer point out the following limitations for image based visual tracking of the hands (Sturman and Zeltzer 1994):

The resolution of video cameras is too low to both resolve the fingers easily and cover the field of view encompassed by broad hand motions.

The 30- or 60- frame-per-second conventional video technology is insufficient to capture rapid hand motion.

Fingers are difficult to track as they occlude each other and are occluded by the hand.

There are two different approaches to vision based gesture recognition; model based techniques which try to create a three-dimensional model of the users hand and use this for recognition, and image based techniques which calculate recognition features directly from the hand image. Rehg and Kanade (1994) describe a vision-based approach that uses stereo camera to create a cylindrical model of the hand. They use finger tips and joint links as features to align the cylindrical components of the model. Etoh, Tomono and Kishino (1991) report similar work, while Lee and Kunii use kinematic constraints to improve the model matching and recognize 16 gestures with XX% accuracy (1993). Image based methods typically segment flesh tones from the background images to find hands and then try and extract features such as fingertips, hand edges, or gross hand geometry for use in gesture recognition. Using only a coarse description of hand shape and a hidden markov model, Starner and Pentland are able to recognize 42 American Sign Language gestures with 99% accuracy (1995). In contrast, Martin and Crowley calculate the principle components of gestural images and use these to search the gesture space to match the target gestures (1997). to generate recommendations. To help business user's select trusted online auction sellers, a recommender system was designed in which trading relationships are used to calculate the level of recommendations.

GESTURE TYPES

In computer interfaces, two types of gestures are distinguished:

1. Offline gestures: Those gestures that are processed after the user interaction with the object. An example is the gesture to activate a menu.
2. Online gestures: Direct manipulation gestures. They are used to scale or rotate a tangible object.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

1. G. R. S. Murthy & R. S. Jadon proposed the importance of gesture recognition lies in building efficient human–machine interaction. Its applications range from sign language recognition through medical rehabilitation to virtual reality. Given the amount of literature on the problem of gesture recognition and the promising recognition rates reported, one would be led to believe that the problem is nearly solved. Sadly their is not so. A main problem hampering most approaches is that they rely on several underlying assumptions that may be suitable in a controlled lab setting but do not generalize to arbitrary settings. Several common assumptions include: assuming high contrast stationary backgrounds and ambient lighting conditions.

2. Erdem, E. Yardimci, Y. Atalay, V. Cetin proposed system is used to control the mouse cursor and implement its function using a real-time camera. We implemented mouse movement, selection of the icons and its functions like right, left, double click and scrolling. This system is based on image comparison and motion detection technology to do mouse pointer movements and selection of icon. From the results, we can expect that if the algorithms can work in all environments then our system will work more efficiently. This system could be useful in presentations and to reduce work space.

3. Dorin Comaniciu , Visvanathan Ramesh , Peter Meer A new approach toward target representation and localization, the central component in visual tracking of nonrigid objects, is proposed. The feature histogram-based target representations are regularized by spatial masking with an isotropic kernel. The masking induces spatially-smooth similarity functions suitable for gradient-based optimization, hence, the target localization problem can be formulated using the basin of attraction of the local maxima. We employ a metric derived from the Bhattacharyya coefficient as similarity measure, and use the mean shift procedure to perform the optimization.

4. Meenakshi panwar and pawan singh mehra Proposed a shape based approach for hand gesture recognition with several steps including smudges elimination orientation detection, thumb detection, finger counts etc Visually Impaired people can make use of hand gestures for writing text on electronic document like MS Office, notepad etc. The strength of their approach includes its simplicity, ease of implementation, and it does not required any significant amount of training or post processing, it provide us with the higher recognition rate with minimum computation time.

5. pei xu In their project, they develop a real-time gesture-based HCI system who recognizes gestures only using one monocular camera and extend the system to the HRI case. The developed system relies on a CNN classifier to learn features and to recognize gestures. They employ a series of steps to process the image and to segment the hand region before feeding it to the CNN classifier in order to improve the performance of the CNN classifier. 3,200 gesture images are collected to test the CNN classifier and demonstrate that the CNN classifier combined with our image processing steps can recognize gestures with high accuracy in real time.

6. Zhang, L., Lin, F., Zang, B In this paper, we presented a novel approach for Content Based Image Retrieval by combining the color and texture features called Wavelet-Based Color Histogram Image Retrieval (WBCHIR). Similarity between the images is ascertained by means of a distance function. The experimental result shows that the proposed method outperforms the other retrieval methods in terms of Average Precision. Moreover, the computational steps are effectively reduced with the use of Wavelet transformation. As a result, there is a substational increase in the retrieval speed. The whole indexing time for the 1000 image database takes 5-6 minutes.

7. Dorin Comaniciu , Visvanathan Ramesh , Peter Meer In this paper, we presented a novel approach for Content Based Image Retrieval by combining the color and texture features called Wavelet-Based Color Histogram Image Retrieval (WBCHIR). Similarity between the images is ascertained by means of a distance function. The experimental result shows that the proposed method outperforms the other retrieval methods in terms of Average Precision. Moreover, the computational steps are

effectively reduced with the use of Wavelet transformation. As a result, there is a substantial increase in the retrieval speed. The whole indexing time for the 1000 image database takes 5-6 minutes.

8. M.A.Moni proposed Technique of gesture recognition using HMM has been reviewed and analyzed. Most of the time using colored images as better result can be achieved. Approaches is that they rely on several underlying assumptions that do not generalize to arbitrary settings. With the help of sign language which these patients use as a means of communication can acquire proper speech. Recall measures the ability of the system to retrieve all the models that are relevant, while precision measures the ability of the system to retrieve only the models that are relevant

9. Dinesh Kumar Vishwakarm and Varun Grover proposed Method based on real-time hand tracking and gesture recognition. The system's results have shown that their efficient interface can be used by the users with ease and accuracy in all lighting conditions. More applications can't be controlled using hand gestures using their method. Real-time gesture-based HCI system, The developed system relies on a CNN classifier to learn features and to recognize gestures. The major drawback is the system is working for invalid Gestures too.

10. X. Wan and C. C. Kuo, "Color distribution analysis and quantization for image retrieval", In SPIE Storage and Retrieval for Image and Video Databases The performance of retrieval of the system can be measured in terms of its recall and precision. Recall measures the ability of the system to retrieve all the models that are relevant, while precision measures the ability of the system to retrieve only the models that are relevant. it provide us with the higher recognition rate with minimum computation time. The major drawback is the system is working for invalid Gestures too. With the help of sign language which these patients use as a means of communication can acquire proper speech The whole indexing time for the 1000 image database takes 5-6 minutes.

EXISTING SYSTEM

The a virtual monitor is defined as a “virtual space where the mouse pointer can be controlled by the hands between the Kinect and the user.” The advantage that results from using the virtual monitor is a reference point that can control the mouse pointer. In addition, the coordinates in the virtual space can express the coordinates on the real monitor accurately, making the movement of the mouse pointer natural. In addition, it is possible to increase the accuracy of the mouse functions and make them more intuitive, because the user can use the virtual monitor as a touch-screen.

A user's physical features are used to create a virtual monitor, because every user's physical features are different, including their height and arm length. Therefore, the scope to deal with the interface for each person individually is limited and different. Thus, the virtual monitor must be created with the appropriate size and location for each user by extracting the user's physical features, to make it user-friendly. The proposed hand-mouse interface extracts a user's physical features using Kinect in real time. The RGB image that is obtained using the Kinect SDK can reconstruct a skeleton image. The coordinates and depth of each joint are obtained using this skeleton image. information that can be obtained using the Kinect SDK.

Drawbacks of Existing System:

- Plans recommended are not dynamic
- Not working for long running process
- Not well defined in the info of categories

PROPOSED SYSTEM

Use of mouse is the most not unusual manner of interacting with a monitor. but for laptops we've got a sensitive contact pad which may be broken effortlessly. these kind of require upkeep and time to time substitute. wi-fi mouse is a primary advancement

but it calls for precise hardware. it could be used from a specific radius and calls for a floor to function the mouse.

The technique use will provide with a higher way of coaching via projectors and also we will do easy mouse operations with out the use of the mouse (without a doubt). This helps in shows and seminars thoroughly. The implementation of opencv in python for a gesture recognition machine for the functionalities of the mouse. item popularity strategies are used for extraction of data from the capture. Do the write operation, drag and drop operations at the side of different mouse actions. This could be helpful for the teaching method involving projectors. this may triumph over the restrictions a corporeal device has could enhance the revel in of interaction to the customers. The runtime operations are managed by the webcam of the connected laptop or desktop. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Only use a single camera we pass it as '0'. Can add additional camera to the system and pass it as 1,2 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture. We could also apply colour detection techniques to any image by doing simple modifications in the code. The proposed system controls the functions of mouse pointer by detecting red, and perform the mouse functions such as left click, dragging, cursor movement, and the file transfer between two systems in a same network. This method detect the red colour objects for the mouse control. The user uses the red colour objects on their finger tip for the better performance. When the number of contours are two, then it perform the simple mouse movement action. Otherwise, when the number of contour is one then it perform the left click. This system also supports the simple file transfer between two or more systems in the same network connection. The left side of the computer's screen act as a communication channel between the systems. ie, the file which is to be copy should drag and drop on the left side of the computer's screen. Then the dropped file will be copied to the destination or the receiver system.

Algorithm

The Algorithm used in the proposed system includes HSV algorithm. In colour image processing, there are various models one of which is the hue, saturation, value (HSV) model. Using this model, an object with a certain colour can be detected and to reduce the influence of light intensity from the outside. Part of image processing is to use colour based processing. Digital image processing is a discipline that studies image processing techniques. The image referred in this research is a static image from vision sensors (webcam). Mathematically, the image is a continuous function of light intensity on two-dimensional field. In order to be processed by a computer, an image should be presented numerically with discrete values. A digital image can be represented by a two-dimensional matrix $f(x, y)$ consisting of M columns and N rows. In colour image processing, there are various models one of which is the hue, saturation, value (HSV) model. Using this model, an object with a certain colour can be detected and to reduce the influence of light intensity from the outside. Tests were performed using six kinds of colours, ie brown, yellow, green, blue, black and white. This Virtual Mouse Hand Recognition application uses a simple colour cap on the finger without the additional requirement of the hardware for the controlling of the cursor using simple gestures and hand control. This is done using vision based hand gesture recognition with inputs from a webcam. The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program. We capture the live feed stream, frame by frame. Then we process each captured frame which is in RGB(default) colour space to HSV colour space. There are more than 150 colour-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV.

The operation is performed by clicking the object and dragging it. It is similar to the open gesture, but the difference is we only have one object here so we only need to calculate the center of it. And that will be placed on the location where we will position our mouse pointer. Instead of mouse release operation we will be performing a mouse press operation. In order to implement the dragging we introduce a variable 'pinchflag'.

It will be set to 1 if it was clicked earlier. So after clicking whenever we find the open gesture we check if the pinchflag is set to 1. If it is set to one then Drag operation is performed otherwise the mouse move operation is performed. The major extension to this work can be done to make system able to work at much complex background and compatible with different light conditions. It can be made as an effective user interface and which can include all mouse functionalities. And also, it would be ideal to research into advanced mathematical materials for image processing and investigate on different hardware solutions that would result in more accurate hand detections. Virtual Mouse that will soon to be introduced to replace the physical computer mouse to promote convenience while still able to accurately interact and control the computer system. To do that, the software requires to be fast enough to capture and process every image, in order to successfully track the user's gesture. Therefore, this project will develop a software application with the aid of the latest software coding technique and the open-source computer vision library also known as the OpenCV. Not only did this project show the different gesture operations that could be done by the users but it also demonstrated the potential in simplifying user interactions with personal computers and hardware systems.

CHAPTER 3

METHODOLOGY

METHODOLOGY:

Installing required libraries

Writing gesture recognition code

Gesture Recognition steps:

Step1: Defining coordinates of display and resolution of camera.

Step 2: Determining it's a closed gesture and bounding the objects.

Step 3: Drawing bounds for the object and lines in between.

Step 4: Defining the point of action for the mouse controls.

Step 5: Conversion of camera image to display resolution.

Step 6: Defining actions of the Moments.

REQUIREMENTS:

Programming Language – Python 2.7 or above

Packages used:

1. Pynput
2. Pyobjc

3. Wx

4. Numpy

GETTING STARTED

OpenCV can be downloaded from the following link:

<http://sourceforge.net/projects/opencvlibrary/>

Choose any of the several available versions. Prefer OpenCV2.1

Next, you would need a compiler like DevC++, CodeBlocks, VisualC++. These can be downloaded from the following links:

- DevC++ : <https://sourceforge.net/projects/orwelldevcpp/>
- CodeBlocks : <http://www.codeblocks.org/downloads/26/>
- VisualC++ : <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express>

Install one of the above compilers and next you would need to link the library with the installed compiler.

INTEGRATION OF OPENCV WITH DEVCC++

First of all, Indicate the header files you want to add. for that, select ***Tools->Compiler Options.***



Fig.3.1 Tools option

Then click on the plus sign(Under Compiler set to configure)to add a new compiler named here,OpenCV.

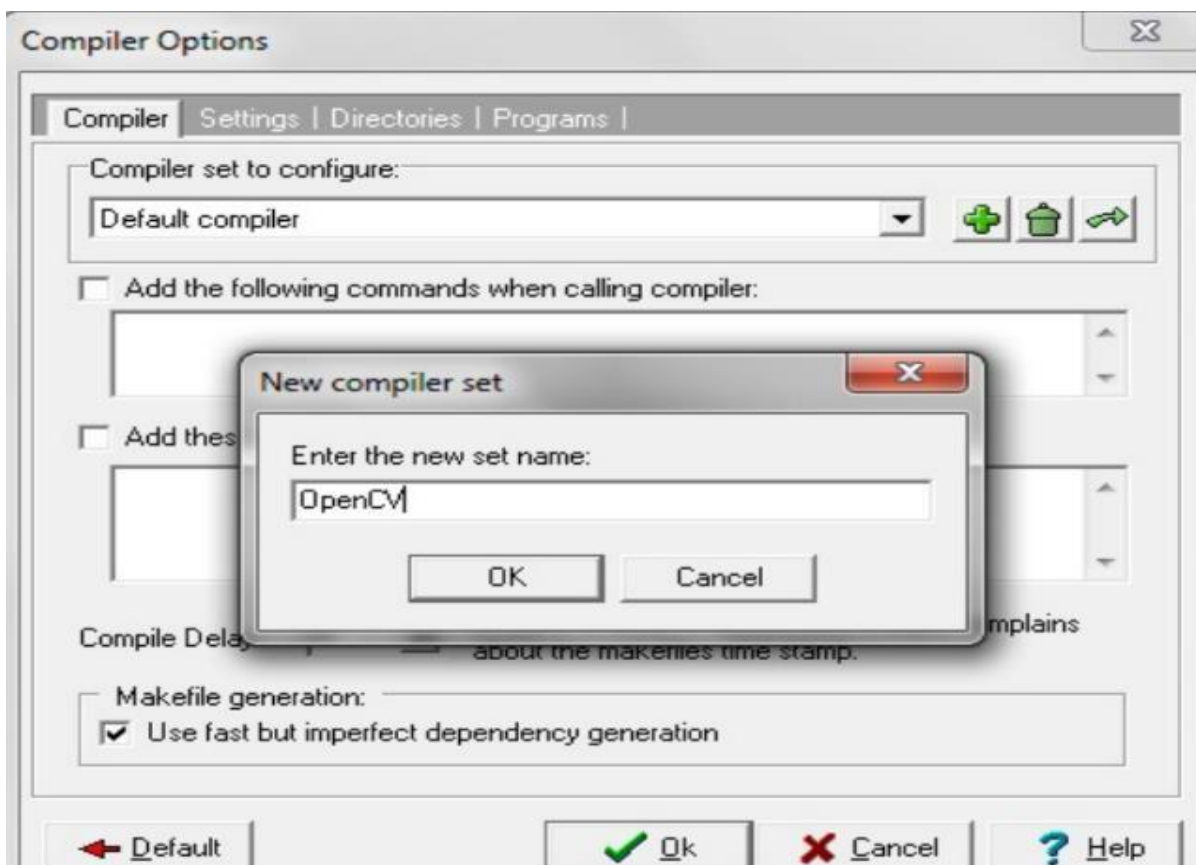


Fig.3.2 Compiler options

To finish on this section Add the following commands.. write-L"C:\OpenCV2.1\lib" -
lxcv210 -lcv210 -lcvaux210 -lhighgui210 -lml210

[note that the text in inverted commas is basically the location of the lib(libraries) folder of OpenCV2.1. So, it won't work well if u've not installed OpenCV in the default folde.]

CONFIGURING INCLUDED FILES

Next, click on Directories and then on C Includes to add all the headers, located in some C:\OpenCV2.1 subdirectories. You only need to add C:\OpenCV2.1\include\opencv in the include tab to get things to work.

If you want to code in C++ then do the same for C++ includes

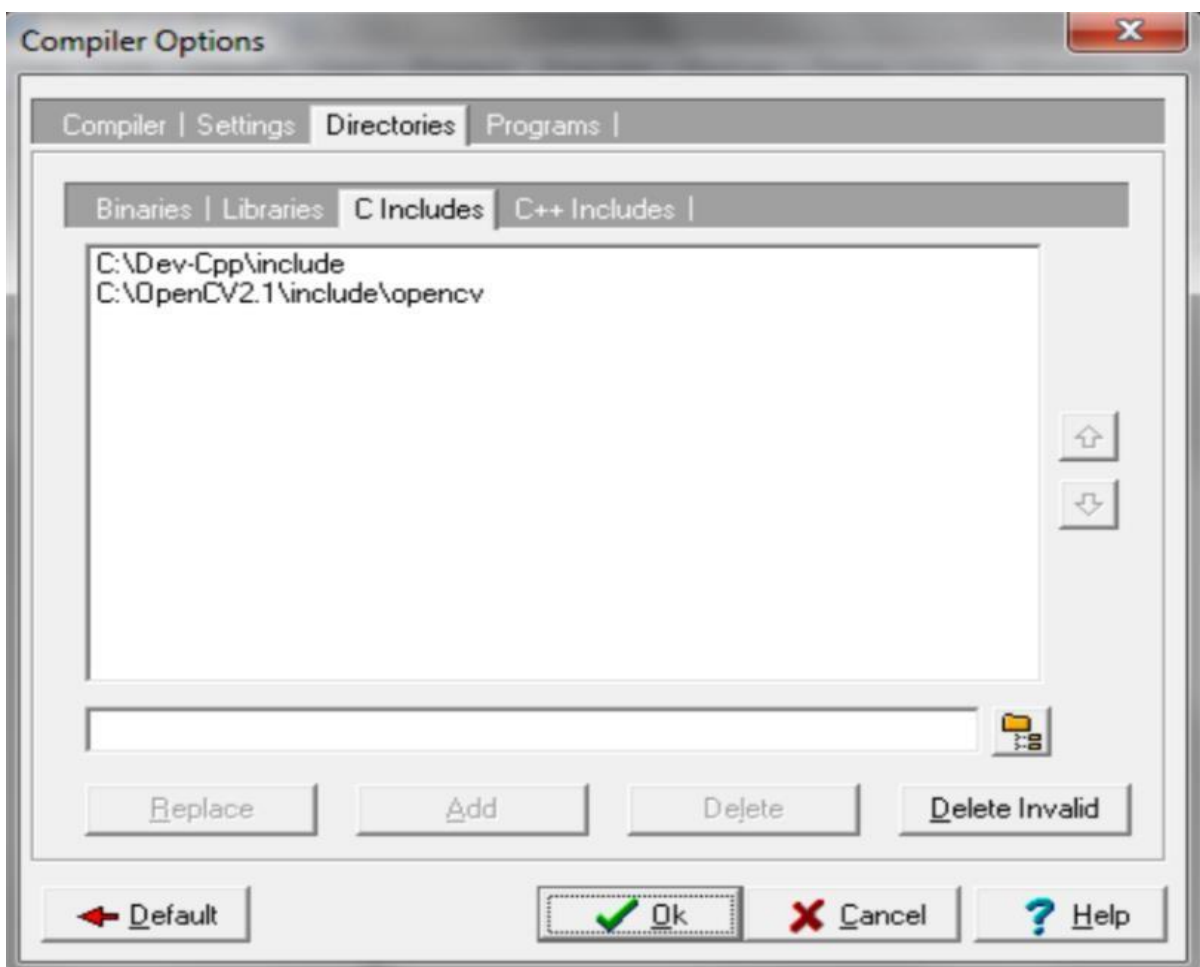


Fig.3.3 Configuring included files

CONFIGURING STATIC LIBRARY FILES : In the libraries section under the same heading directories you will need to add C:\OpenCV2.1\lib . (if already present, ignore this step.)

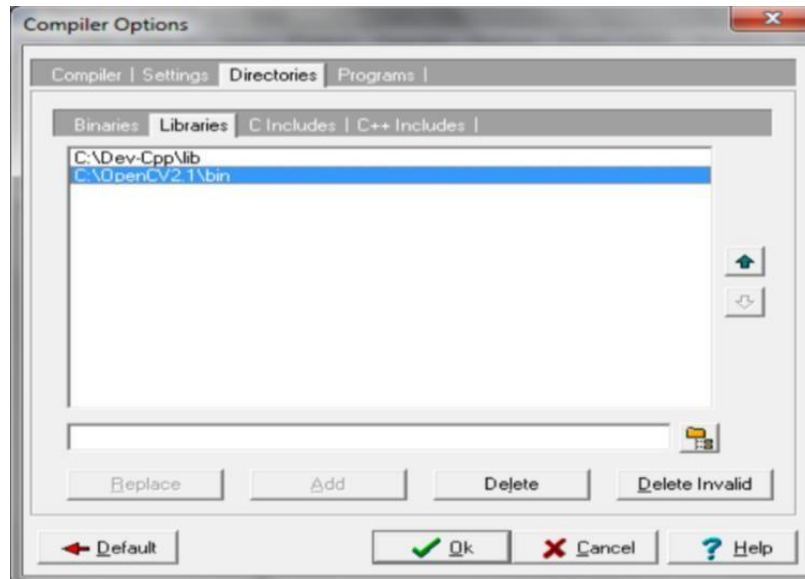


Fig.3.4 Configuring static library

CONFIGURING DYNAMIC LIBRARY FILES And to finish, add the bin directory where the dlls are: i.e. add C:\OpenCV2.1\bin to binaries subdivision.

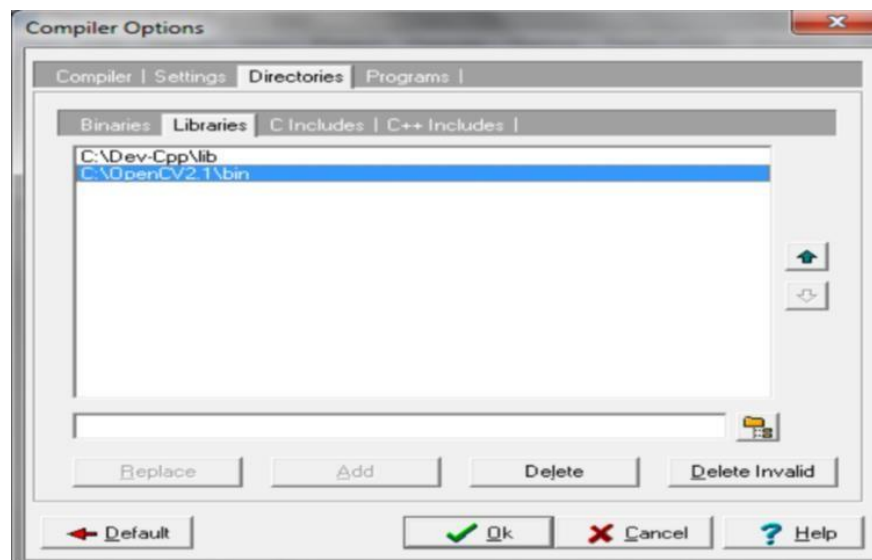


Fig.3.5 configuring dynamic library

Congratulations...!! With this, You are done with configuring OpenCV with DevCPP, you can try running a sample code.

NUMPY INTRODUCTION

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations:

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like SciPy (Scientific Python) and Matplotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.

Numpy – Environment

Standard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

```
pip install numpy
```

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

Windows

Anaconda (from <https://www.continuum.io>) is a free Python distribution for SciPy stack. It is also available for Linux and Mac.

Canopy (<https://www.enthought.com/products/canopy/>) is available as free as well as commercial distribution with full SciPy stack for Windows, Linux and Mac.

Python (x,y): It is a free Python distribution with SciPy stack and Spyder IDE for Windows OS. (Downloadable from <http://python-xy.github.io/>)

Linux

Package managers of respective Linux distributions are used to install one or more packages in SciPy stack.

For Ubuntu

```
sudo apt-get install python-numpy python-scipy python-matplotlibpythonipython-notebook python-pandas python-sympy python-nose
```

For Fedora

```
sudo yum install numpyscipy python-matplotlibpython python-pandas sympy python-nose atlas-devel
```

Building from Source

Core Python (2.6.x, 2.7.x and 3.2.x onwards) must be installed with distutils and zlib module should be enabled.

GNU gcc (4.2 and above) C compiler must be available. To install NumPy, run the following command.

```
Python setup.py install
```

To test whether NumPy module is properly installed, try to import it from Python prompt.

```
import numpy
```

If it is not installed, the following error message will be displayed.

Traceback (most recent call last):

```
File "<pyshell#0>", line 1, in <module>
```

```
import numpy
```

ImportError: No module named 'numpy'

Alternatively, NumPy package is imported using the following syntax:

```
import numpy as np
```

NUMPY – NDARRAY OBJECT

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type:

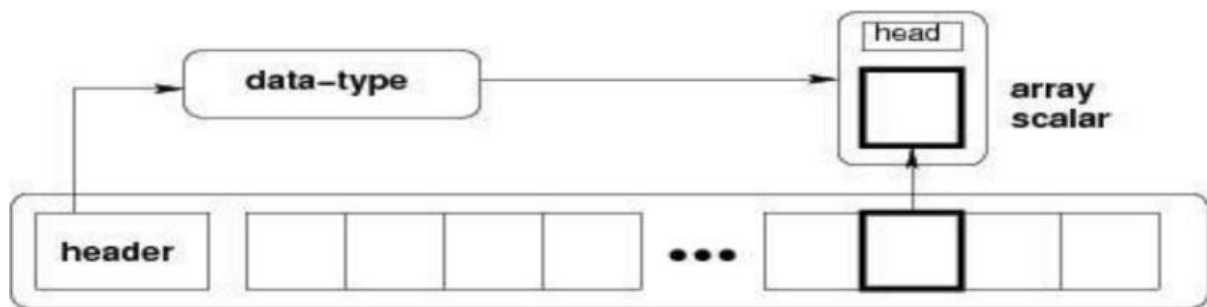


Fig.3.6 ndarray object

An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows:

```
numpy.array
```

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

```
numpy.array(object, dtype=None, copy=True, order=None, subok=False, ndmin=0)
```

Object	Any object exposing the array interface method returns an array, or any (nested) sequence
Dtype	Desired data type of array, optional
Copy	Optional. By default (true), the object is copied

Order	C (row major) or F (column major) or A (any) (default)
Subok	By default, returned array forced to be a base class array. If true, sub-Classes passed through
Ndimin	Specifies minimum dimensions of resultant array

Table.3.1 Numpy objects

NUMPY – DATA TYPES

NumPy supports a much greater variety of numerical types than Python does. The following table shows different scalar data types defined in NumPy

DataTypes	Description
bool_	Boolean (True or False) stored as a byte
Int_	Default integer type (same as C long; normally either int64 or int32)
Intc	Identical to C int (normally int32 or int64)
Intp	Integer used for indexing (same as C ssize_t; normally either int32 or int64)
Int8	Byte (-128 to 127)
Int16	Integer (-32768 to 32767)
Int32	Integer (-2147483648 to 2147483647)
Int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)

UInt16	Unsigned integer (0 to 65535)
UInt32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

Table.3.2 Numpy Datatypes

NumPy numerical types are instances of dtype (data-type) objects, each having unique characteristics. The dtypes are available as np.bool_, np.float32, etc.

Data Type Objects (dtype)

A data type object describes interpretation of fixed block of memory corresponding to an array, depending on the following aspects:

- Type of data (integer, float or Python object)
- Size of data
- Byte order (little-endian or big-endian)

- In case of structured type, the names of fields, data type of each field and part of the
- memory block taken by each field
- If data type is a subarray, its shape and data type

The byte order is decided by prefixing '<' or '>' to data type. '<' means that encoding is little-endian (least significant is stored in smallest address). '>' means that encoding is big-endian (most significant byte is stored in smallest address).

A dtype object is constructed using the following syntax:

```
numpy.dtype(object, align, copy)
```

The parameters are:

- Object: To be converted to data type object
- Align: If true, adds padding to the field to make it similar to C-struct
- Copy: Makes a new copy of dtype object. If false, the result is reference to built-in data type object

Each built-in data type has a character code that uniquely identifies it.

- 'b': boolean
- 'i': (signed) integer
- 'u': unsigned integer
- 'f': floating-point
- 'c': complex-floating point
- 'm': timedelta
- 'M': datetime
- 'O': (Python) objects
- 'S', 'a': (byte-)string
- 'U': Unicode
- 'V': raw data (void)

WXPYTHON – INTRODUCTION

wxPython is a Python wrapper for wxWidgets (which is written in C++), a popular cross-platform GUI toolkit. Developed by Robin Dunn along with Harri Pasanen, wxPython is implemented as a Python extension module.

Just like wxWidgets, wxPython is also a free software. It can be downloaded from the official website <http://wxpython.org>. Binaries and source code for many operating system platforms are available for download on this site.

Principal modules in wxPython API include a core module. It consists of wxObject class, which is the base for all classes in the API. Control module contains all the widgets used in GUI application development. For example, wx.Button, wx.StaticText (analogous to a label), wx.TextCtrl (editable text control), etc.

wxPython API has GDI (Graphics Device Interface) module. It is a set of classes used for drawing on widgets. Classes like font, color, brush, etc. are a part of it. All the container window classes are defined in Windows module.

Official website of wxPython also hosts Project Phoenix – a new implementation of wxPython for Python 3.*. It focuses on improving speed, maintainability, and extensibility. The project began in 2012 and is still in beta stage.

IMPLEMENTATION

The implementation of opencv in python for a gesture recognition system for the functionalities of the mouse. Object recognition techniques are used for extraction of information from the capture. We do the write operation, drag and drop operations along with other mouse actions. This will be helpful for the teaching procedure involving projectors. This will overcome the restrictions a corporeal device has would enrich the experience of interaction to the users. Use of mouse is the most common way of interacting with a monitor. But for laptops we have a sensitive touch pad which can be damaged easily. All these require maintenance and time to time replacement.

Virtual Mouse that will soon to be introduced to replace the physical computer mouse to promote convenience while still able to accurately interact and control the computer system. To do that, the software requires to be fast enough to capture and process every image, in order to successfully track the user's gesture. Therefore, this project will develop a software application with the aid of the latest software coding technique and the open-source computer vision library also known as the OpenCV. The scope of the project is as below:

- Real time application.
- User friendly application.
- Removes the requirement of having a physical mouse.

During the process of colour recognition, it contains 2 major phases which are the calibration phase and recognition phase. The purpose of the calibration phase is to allow the system to recognize the Hue Saturation Values of the colours chosen by the users, where it will store the values and settings into text documents, which will be used later on during the recognition phase. While on the recognition phase, the system will start to capture frames and search for colour input with based on the values that are recorded during the calibration phase. The phases of the virtual mouse is as shown in figure below.

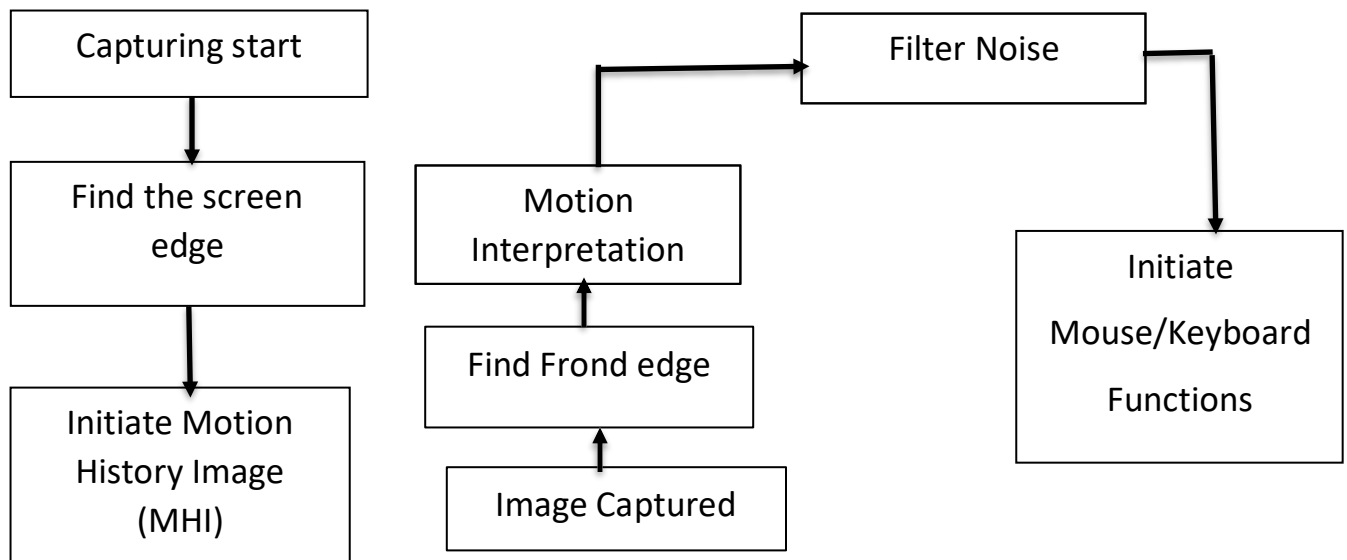


Fig.3.7 Phases

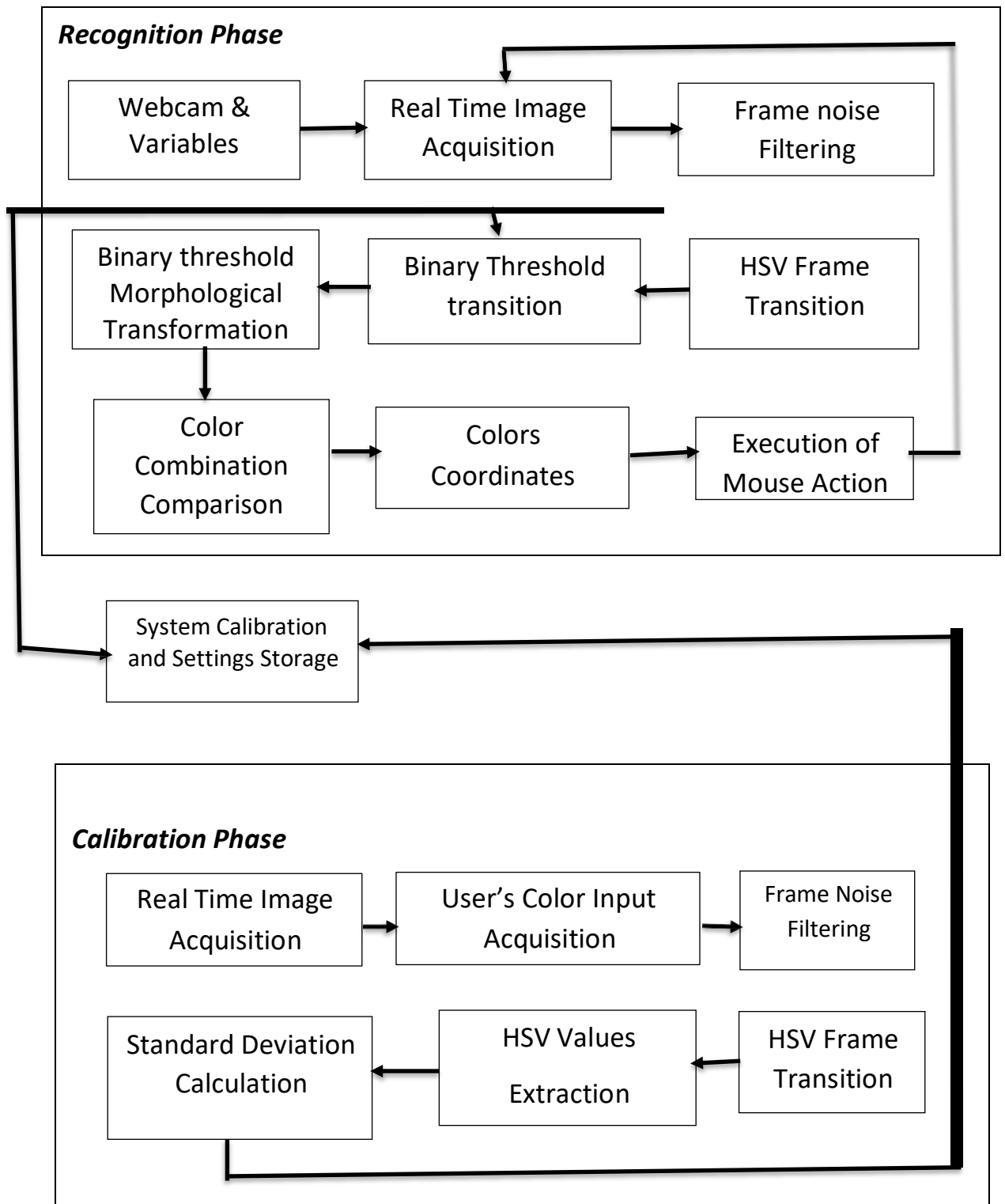


Fig.3.8 Calibration and Recognition Phases

CHAPTER 4

RESULTS AND DISCUSSION

RESULTS:

- Virtual Mouse
- Removes the requirement of having a physical mouse.
- Drag, Drop and click Operations

VIRTUAL MOUSE

A virtual mouse using hand gesture recognition is a system that allows users to give mouse inputs to computer without using an actual mouse hardware. To the extreme, it can also be called as hardware because it uses a camera for tracking hands.

REMOVES THE REQUIREMENT OF HAVING A PHYSICAL MOUSE

Virtual mouse uses camera device to make image processing of the hand gestures to make mouse operations possible without using the physical device as mouse. These steps help in removing the requirement of having the physical mouse.

OPERATIONS

- Click operation is performed by using virtual mouse to make selection etc.
- Drag and drop operations are performed by making the click operation to extend the process by moving the gesture after selection.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION

The approach we use will provide us with a better way of teaching through projectors and also we can do simple mouse operations without using the mouse (virtually). This helps in presentations and seminars very well. first we are doing a mouse release to ensure the mouse left button is not pressed. Then we are converting the detected coordinate from camera resolution to the actual screen resolution. After that we set the location as the mouse.position. but to move the mouse it will take time for the curser so we have to wait till the curser reaches that point. So we started a loop and we are not doing anything there we are just waiting will the current mouse location is same as assigned mouse location. That is for the open gesture.

FUTURE WORK

- Features such as enlarging and shrinking the windows, closing windows, etc. by using the palm and multiple fingers are the future scope of this project.
- In the future, we plan to add more features such as enlarging and shrinking windows, closing windows, etc. by using the palm and multiple fingers.
- Can also open the browser or any drives (C: /D:/E: etc) with the help of hand gestures instead of moving the cursor.

REFERENCES:

- [1]. G. R. S. Murthy & R. S. Jadon "A Review Of Vision Based Hand Gestures Recognition", International Journal of Information Technology and Knowledge Management July-December 2015, Volume 2, No. 2, pp. 405-410
- [2]. Erdem, E. Yardimci, Y. Atalay, V. Cetin, "Computer vision based mouse", Acoustics, Speech, and Signal Processing, 2015. Proceedings. (ICASS). IEEE International
- [3]. R. I. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", 2nd ed. Cambridge University Press, ISBN: 0521540518, 004, 2016
- [4]. Dorin Comaniciu , Visvanathan Ramesh , Peter Meer, "KernelBased Object Tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence, v.25 n.5, p.564-575, May 2016
- [5]. Zhang, L., Lin, F., Zang, B," A CBIR method based on color-spatial feature", In: IEEE Region 10 International Conference TENCON, pp. 166–169 (1999) 18G, 2017
- [6]. Meenakshi Panwar and Pawan Singh Mehra , "Hand Gesture Recognition for Human Computer Interaction", in Proceedings of IEEE International Conference on Image Information Processing (ICIIP 2011), Wanknaghat, India, November 2017
- [7]. Pei Xu, "A Real-Time Hand Gesture Recognition And Human-Computer Interaction System," Multimedia Tools Appl., vol. 76, no. 16, pp. 17 595–17 612, 2017.
- [8]. X. Wan and C. C. Kuo, "Color distribution analysis and quantization for image retrieval", In SPIE Storage and Retrieval for Image and Video Databases IV, Vol. SPIE 2670, pp- 9–16. 1996. 46, 2018
- [9]. M. W. Ying and Z. HongJiang, "Benchmarking of image feature for content-based retrieval", IEEE. Pp-253-257, 1998. 47, 2018
- [10]. Z. Zhenhua, L. Wenhui and L. Bo, "An Improving Technique of Color Histogram in Segmentationbased Image Retrieval", 2009 Fifth International Conference on Information Assurance and Security, IEEE, pp-381-384, 2009, 2019.

APPENDIX

Sample Code

```
import cv2

import numpy as np

from pynput.mouse import Button ,Controller

import wx

mouse=Controller()


app=wx.App(False)
(sx,sy)=wx.GetDisplaySize()
(camx,camy)=(420,240)
cap=cv2.VideoCapture(0)
cap.set(3,camx)
cap.set(4,camy)


#range for HSV (green color)
lower_g=np.array([33,70,30])
upper_g=np.array([102,255,255])


#Kerenel
kernelOpen=np.ones((5,5))
kernelClose=np.ones((20,20))


mLocOld=np.array([0,0])
mouseLoc=np.array([0,0])


DampingFactor=2 #Damping factor must be greater than 1


isPressed=0
```

```
openx,openy,openw,openh=(0,0,0,0)
```

```
while True:
```

```
    ret,img=cap.read()
```

```
    img=cv2.resize(img,(540,320))
```

```
    imgHSV=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
```

```
    mask=cv2.inRange(imgHSV,lower_g,upper_g)
```

```
    #using morphology to erase noise as maximum as possible
```

```
    new_mask=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
```

```
    another_mask=cv2.morphologyEx(new_mask,cv2.MORPH_CLOSE,kernelClose)
```

```
    final_mask=another_mask
```

```
    im2,conts,h=cv2.findContours(final_mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
```

```
    # Once 2 objects are detected the center of there distance will be the reference on  
controlling the mouse
```

```
    if(len(conts)==2):
```

```
        #if the button is pressed we need to release it first
```

```
        if(isPressed==1):
```

```
            isPressed=0
```

```
            mouse.release(Button.left)
```

```
        #drawing the rectagle around both objects
```

```
        x1,y1,w1,h1=cv2.boundingRect(conts[0])
```

```
        x2,y2,w2,h2=cv2.boundingRect(conts[1])
```

```
        cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
```

```

cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)

#the line between the center of the previous rectangles
cx1=int(x1+w1/2)
cy1=int(y1+h1/2)
cx2=int(x2+w2/2)
cy2=int(y2+h2/2)
cv2.line(img,(cx1,cy1),(cx2,cy2),(255,0,0),2)

#the center of that line (reference point)
clx=int((cx1+cx2)/2)
cly=int((cy1+cy2)/2)
cv2.circle(img,(clx,cly),2,(0,0,255),2)

#adding the damping factor so that the movement of the mouse is smoother
mouseLoc=mLocOld+((clx,cly)-mLocOld)/DampingFactor
mouse.position=(sx-int((mouseLoc[0]*sx)/camx),int((mouseLoc[1]*sy)/camy))
while mouse.position!=(sx-
int((mouseLoc[0]*sx)/camx),int((mouseLoc[1]*sy)/camy)):
    pass

#setting the old location to the current mouse location
mLocOld=mouseLoc

#these variables were added so that we get the outer rectangle that combines
both objects
openx,openy,openw,openh=cv2.boundingRect(np.array([[[[x1,y1],[x1+w1,y1+h1],[
x2,y2],[x2+w2,y2+h2]]]]))

#when there's only when object detected it will act as a left click mouse
elif(len(conts)==1):

```

```

x,y,w,h=cv2.boundingRect(conds[0])

# we check first and we allow the press fct if it's not pressed yet
#we did that to avoid the continues pressing
if(isPressed==0):

    if(abs((w*h-openw*openh)*100/(w*h))<30): #the difference between th
combined rectangle for both object and the
        isPressed=1                #the outer rectangle is not more than 30%
        mouse.press(Button.left)
        openx,openy,openw,openh=(0,0,0,0)

#this else was added so that if there's only one object detected it will not act as a
mouse
else:
    #getting rectangle coordinates and drawing it
    x,y,w,h=cv2.boundingRect(conds[0])
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

    #getting the center of the circle that will be inside the outer rectangle
    cx=int(x+w/2)
    cy=int(y+h/2)
    cv2.circle(img,(cx,cy),int((w+h)/4),(0,0,255),2)#drawing that circle

    mouseLoc=mLocOld+((cx,cy)-mLocOld)/DampingFactor
    mouse.position=(sx-int((mouseLoc[0]*sx)/camx),int((mouseLoc[1]*sy)/camy))
    while mouse.position!=(sx-
int((mouseLoc[0]*sx)/camx),int((mouseLoc[1]*sy)/camy)):
        pass
    mLocOld=mouseLoc

```

```
        #showing the results
        cv2.imshow("Virtual mouse",img)

        #waiting for 'W' to be pressed to quit
        if cv2.waitKey(1) & 0xFF==ord('w'):
            break

    cap.release()
    cv2.destroyAllWindows()

}
```

SCREENSHOTS



Fig.5.1 Desktop view of clicking operation

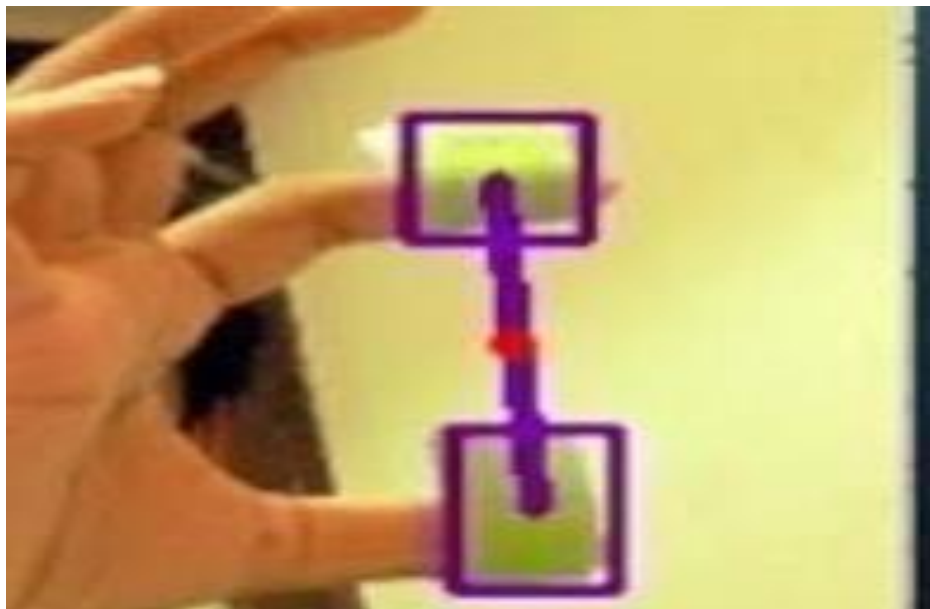


Fig.5.2 View from webcam



Fig.5.3 Click operation view from webcam