A
PROJECT REPORT
ON

INTERACTIVE HCI SYSTEM TO TURN ANY SURFACE TO TOUCHSCREEN USING OPENCV

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING

In

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

By

KUSHAL CHAUDHARI	**B120433100**
HUZEFA LOKHANDWALA	**B120433103**
SHABIB KHAN	**B120433088**
SHIVANI SADARE	**B120433148**

Under the Guidance of

Prof. S.V.Thakur

Sinhgad Institutes

Submitted to

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

STES'S SINHGAD ACADEMY OF ENGINEERING, PUNE-411048

2017-2018

CERTIFICATE

This is to certify that the project report entitled

INTERACTIVE HCI SYSTEM TO TURN ANY SURFACE TO TOUCHSCREEN USING OPENCV

Submitted By

KUSHAL CHAUDHARI	**B120433100**
HUZEFA LOKHANDWALA	**B120433103**
SHABIB KHAN	**B120433088**
SHIVANI SADARE	**B120433148**

Is a bonafide work carried out by them under the supervision by **Prof. S. V. Thakur** and it is approved for the partial fulfilment of the requirement of **Savitribai Phule Pune University** for the Seminar in the Final Year of Electronics and Telecommunication Engineering.

This project report has not been earlier submitted to any other institute or University for the award of any degree or diploma.

Prof. S. V. Thakur	**Dr. M. M. Sardeshmukh**	**Dr. K. P. Patil**
Guide	**H.O.D**	**Principal, SAOE, Pune**

Place : Pune		**Prof. S. S. Shah**
Date :	**External Examiner**	**Project co-ordinator**

ACKNOWLEDGEMENT

We are thankful to **Prof. M.M. Sardeshmukh**, Head of E&TC Engineering Department of Sinhgad Academy Of Engineering, Pune for his kind co-operation and continuous motivation which helped us a lot during the course of his project.
Also we take this opportunity to express our sincere gratitude to our project guide **Prof. S.S. Shah** for their valuable and timely guidance. Without their sustained interest, unlimited patience and sound counsel, this work would have not been possible. The suggestions and ideas given by them have contributed significantly in the success of this project.
Last but not the least; we are also thankful to our parent and friends who have helped us directly or indirectly for the successful completion of this project.

Project Group Members:

KUSHAL CHAUDHARI	**B120433100**
HUZEFA LOKHANDWALA	**B120433103**
SHABIB KHAN	**B120433088**
SHIVANI SADARE	**B120433148**

Abstract

The interactive projector presents a new approach for controlling mouse movements using real-time camera. This technology can fill the gap that currently exists between human and a computer. The projector projects the computer's desktop onto a surface where the user controls the computer using interactive devices. The current technologies involve infrared detection, a resistive touch, an electromagnetic pen or associated software. The proposed model can be implemented using OpenCV library to speed-up image process and obtain high real-time performance. This low-cost interactive projector can be used in a variety of settings, including classrooms, corporate board rooms and other work environments. Furthermore, this study demonstrates the potential application of good scientific and practical value.

Table of Contents

List of Figure

CHAPTER - 1
INTRODUCTION

1.1 PRELUDE :

As computer technology continues to develop, people have smaller and smaller electronic devices and want to use them ubiquitously. There is a need for new interfaces designed specifically for reducing the effort of the end user. But there is still a disconnect between a user and a computer. Companies all over the globe are attempting to bridge this gap by focusing their R&D on new innovative HCI technologies. Human-Computer Interaction (HCI) researches the design and use of computer technology, focused on the interfaces between people and computers. Researchers in the field of HCI both observe the ways in which humans interact with computers and design technologies that let humans interact with computers in novel ways. Touch screens are a universal way to interact with computers and nowadays it is used globally in many applications. However, touch screens that are applied to projector systems have cost and other hardware limitations. By applying computer vision technology we can overcome these limitations to facilitate a cost efficient interactive system. Hence the main aim of our project is to bridge the disconnect between human and a computer. This project proposes the use of LED tracking and image processing along with perspective transformation which is cost effective and can be added to any projector to make it interactive.

1.2 MOTIVATION

With the growing development of technology, touch screens are becoming more popular in both commercial and personal purposes. Nowadays, Mobile phones, tablets, game machines and control panels cannot be imagined without the touch screens. In current scenario it is not possible for traditional projectors to provide an interactive link between digital and physical world. In a country like India it is not possible to integrate every classroom with a interactive projector system based on capacitive or resistive touchscreen technologies that increases hardware requirements and cost. Therefore the motivation is to apply a touch interface which is cost effective to existing traditional projector systems.

1.3 PROBLEM STATEMENT

To design a cost-efficient system which will provide an advanced interface between the user and projector using computer vision algorithms and an LED as a pointing device.

1.4 OBJECTIVES OF PROPOSED WORK

1. To design an LED pen with a momentary switch to act as a clicking mechanism.
2. Perspective transformation to triangulate location of the LED pointer on projected surface .
3. LED pointer detection and blob creation.
4. To generate contours to track the co-ordinates of LED pointer.
5. Mouse events using python GUI modules.
6. Integrating and testing all modules.

CHAPTER - 2
LITERATURE SURVEY

2.1 LITERATURE SURVEY

Sr. No	Paper Name	Author Name	Date of Publication	Description
1	An Approach to Convert Conventional Screen into Touch Screen Using Image Processing.	Md. Hanif Ali Sohag, Mst. Mahmuda Khatun, and Mohiuddin A.	2015	• Basic system working. • Block diagram basics. • Colorspace conversion • Brightness contrast control
2	A low cost Interactive Whiteboard system Using the Kinect.	Shuai Zhang, Wenrui He, Qiao Yu, and Xiaojuan Zheng.	2013	• Image Processing • Preliminary Processing • Characteristic Filtering • Perspective Transform
3	A Method for Controlling Mouse Movement using a Real-Time Camera.	Hojoon Park.	2011	• Calibration • Mouse functionality
4	Hacking the Nintendo Wii Remote	Johnny Chung Lee.	2010	• Infrared Camera Tracker • IR pen design
5	An Improved Interactive Whiteboard System: A New Design and an Ergonomic Stylus	Koray Yucel, Nevzat Orhan, Gizem Misirli, Gözde Bal, Yasar Guneri Sahin.	2008	• Basic system working. • Block diagram basics. • NWRC Working
6	Hand Mouse: Real Time Motion Detection System Based on Analysis of Finger Blobs	Ibrahim Furkan Ince, Manuel Socarras	2010	• Basic mouse api working. • Mouse api configurations and functions.
7	The Interactive Whiteboards (IWB) - Uses, Benefits and Challenges	Thierry Karsenti, Tae-Cheon Yang.	2008	• IWB benefits • Ambient light challenges and solutions.

2.2 LITERATURE REVIEW

2.2.1 *An Approach to Convert Conventional Screen into Touch Screen Using Image Processing*

The proposed system in this paper works by capturing the image of a laptop screen instead of an projector screen and utilises HSV color space as primary. The reason for choosing HSV is that the image captured by the webcam is a RGB colorspace. This RGB colorspace image takes longer time to process. As the stylus is using only bright red light as a touch point so there is no need of other color spaces. In order to extract the red light, the captured image is converted to HSV color model which defines a color space in terms of color type, color intensity and color brightness. In our proposed system, we are differentiating the bright red LED by adjusting the HSV values of the image. It provides only the area of the image which contains bright red color ignoring the other component of color.

This paper also uses alpha (α) and beta (β) parameters to adjust brightness and contrast of the looping images frames. Our proposed system uses these adjusted parameters to efficiently differentiate the touchpoint from the background image. For adjusting the contrast and brightness of the capture image the formula used is:

$$g(x) = \alpha f(x) + \beta$$

The parameters $\alpha > 0$ and β are the gain and bias parameters for controlling the contrast and brightness respectively. Considering *f(x)* as the source image pixels and *g(x)* as the output image pixels, the formula can also be written as:

$$g(i,j) = \alpha f(i,j) + \beta$$

where i and j indicates that the pixel is located in the *i-th* row and *j-th* column.

2.2.2 *Hacking the Nintendo Wii Remote*

This paper pioneered concept of the interactive low cost whiteboard systems. The Nintendo Wii remote, or Wiimote, is a handheld device resembling a television remote, but in addition to buttons, it contains a 3-axis accelerometer, a high-resolution high speed IR camera, a speaker, a vibration motor, and wireless Bluetooth connectivity. This technology makes the Wii remote one of the most sophisticated PC-compatible input devices available today.

In the tip of each Wii remote is an IR camera sensor manufactured by PixArt Imaging. The camera chip features an integrated multi-object tracking (MOT) engine, which provides high-resolution, high-speed tracking of up to four simultaneous IR light sources. The camera sensor’s exact specifications are unpublished, but it appears to provide location data with a resolution of 1,024 × 768 pixels, more than 4 bits of dot size or light intensity, a 100 Hz refresh rate, and a 45 degree horizontal field of view. The integrated hardware object tracking minimizes the data transmitted over the wireless connection.

By constraining the movement of IR emitters to a planar display surface, you can interact with the projected image as if it were an interactive whiteboard system. To discover the correspondence between the camera and projector coordinates, you use a four-point calibration process typical for any touch-screen system. From the four registered points, you can compute a homography, a warping matrix for mapping any new point visible to the camera to the correct pixel location in the projected image.

This approach also works with any flat display surface, such as an LCD or plasma television. However, displays that have a thick glass surface can cause unwanted reflections that result in erratic tracking behavior. Our paper also uses the concept based on this homographic transformation also known as perspective wrapping.

2.2.3 Low cost Interactive Whiteboard Using the Kinect

The four main methodologies used in this paper are are :

A. IMAGE PROCESSING:
Kinect obtains a large quantity of infrared data. The raw IR input data contains too much noise to be used for processing. A filtering algorithm is applied to smooth the image processing.

B. PRELIMINARY PROCESSING:
Preliminary positioning is based on two assumptions: one, the luminous spot intensity of the infrared LED is greater than the the IR emitted initiatively by Kinect but not too great to result in a black hole on Kinect depth image. The second assumption is that, the 3D depth of the IR LED is approximate to the background.

C. CHARACTERISTIC FILTERING:
Denoising techniques is based on projection plane. The projection plane should be as close to white in order to get an attractive rendering result. The luminous spot of the LED is then painted black. Next, the central point of the black mass is detected and its position is fixed.

D. PERSPECTIVE TRANSFORM:
The Kinect has 2 cameras, an IR camera and a collar camera. They are at different positions of the Kinect so the have have different coordinates. The correspondence between the two cameras is fixed.

2.2.4 A Method for Controlling Mouse Movement using a Real-Time Camera

In this methodology, hand gesture recognition is used.

A. CALIBRATION:
To recognise hands gesture, the input image is resized in order to map camera coordinates to screen coordinates. This is called as calibration. cvResize() function in OpenCV is used which maps a source image to a destination image as smoothly as possible. Using this function mapping of each input pixel to screen pointer is done.

B. MOUSE FUNCTIONALITY:

The touch area by the stylus can easily be differentiated and tracked due to the fact that the LED light intensity is brighter than the background screen. The contours of the binary image are found using the contour finding algorithm of OpenCV.

Around the touch point where the stylus was touched there might be some small noise areas. To eliminate the noise, only the largest contour area is considered as the touch point. The center co-ordinate is then calculated using moments calculation formula. The co-ordinate is mapped to screen co-ordinates to get the exact location of the touch co-ordinate using the following formula:

$$x = \left(\frac{Px \times X}{Ix}\right)$$

$$y = \left(\frac{Py \times Y}{Iy}\right)$$

where (x,y) are screen coordinates, , are the touch point coordinates on the processed image, Ix, Iy are the image resolution and X,Y are the screen resolution. Then the corresponding co-ordinate is passed to windows default mouse events to simulate mouse actions.

2.2.5 An Improved Interactive Whiteboard System: A New Design and an Ergonomic Stylus

An Interactive whiteboards (IWBs) is a special system that consists of a computer, a projector, a large display and a stylus. Sensors make the surface of the Whiteboard touch sensitive and then they send the gathered data to the computer using using Bluetooth connectivity. Next the computer processes the data sent by sensors, and reflects the changes to the whiteboard by means of projector. The sensor used is Nintendo Wii Remote Control(NWRC) which was introduced by Johnny C. Lee. He showed that NWRC can be used as an IR camera.

The system proposed in this paper consists of dual NWRC for IWB system. Secondary NWRCs are connected to the computer and installed on certain positions that each NWRC should detect the entire image on the surface. When the user touches on the surface, IR-enabled Stylus emits IR light and sends the positioning information to the computer. Primary NWRC is used as Master data. If the computer does not receive any positioning data from primary NWRC, secondary NWRC data automatically is used and hence secondary NWRC acts as master until the primary becomes available. But these systems need to use the NWRC which may not be readily available to the user. In particular, the use of natural interfaces through Interactive Whiteboards (IWs) in education environments can ease the presentation and comprehension of complex concepts, allow collaborative work between teachers and students and improve pedagogical practices. Although there are wide ranges of commercial IW solutions, they are generally expensive and difficult to afford and implement by a large number of education institutions. To solution to make this affordable is to make use of various computer vision algorithms for the required human computer interface.

2.3 MARKET ANALYSIS

2.3.1 Analysis of presence of interactive whiteboards (IWB) in classrooms of various countries

All around the world, more and more classrooms feature an interactive whiteboard (IWB). It refers to an electronic whiteboard that combines touch (pen-and-finger) control of the screen with computerized input from a variety of devices operated by teachers or students. It has become practically standard in the education systems of certain American states and countries such as Australia, Québec and especially Great Britain, where they are present in 100% of elementary schools. Although the success of this technology in the western part of the globe, this technology has not yet reached the Asian countries like India, Korea, China etc. IWBs began to be introduced into Québec's education system only in the last five years. The usual justification for this massive invasion put forward by both governments and businesses is that IWBs can improve school and academic outcomes for learners by improving teaching practices, by diversifying teaching resources (e.g., graphics, videos, audio), and by introducing more interactive teaching and learning activities. This research was published by Thierry Karsenti in 2016.

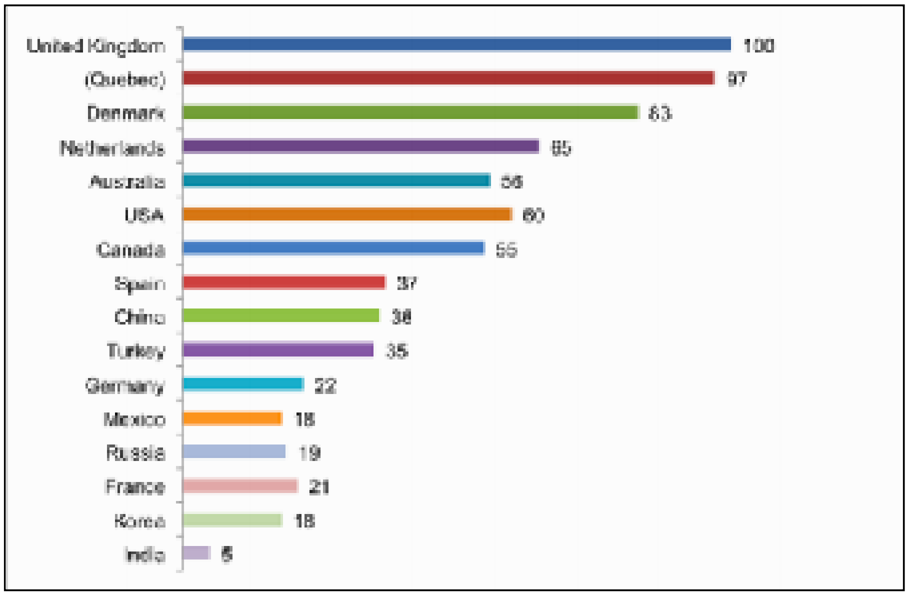


Fig 2.3.1. Analysis of presence of interactive whiteboards

The main reason for the lack of implementation of IWBs in developing countries like India is the high cost. The government and institutional heads would have to make huge investments to apply this technology in every classroom. If the cost is reduced we may see a change in the current scenario and many more developing countries can implement this technology.

Our system aims to provide a platform which has high interactivity on the same level as that of existing IWBs in a cheaper cost efficient way. This can be done by reducing the hardware. The current IWBs have a lot of hardware which constitute to a higher cost factor.

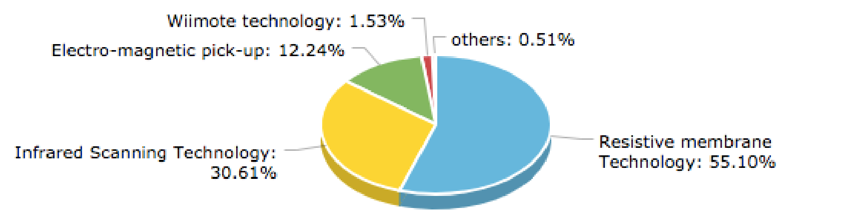
2.3.2 Analysis of types of IWB technologies

The surface of an interactive whiteboard is critical to its functionality. It is a distinguishing factor between the different technologies used in the boards. The interactive whiteboard captures the user inputs and detects where the user is touching the board, this information is then used as input to the computer running the interactive whiteboard software. The 3 main different technologies used for this purpose are:

1. Resistive Membrane - The board surface incorporates a soft flexible vinyl or polyester-based plastic front surface and a rigid backboard. The two layers of resistive material with a small gap between them create a touch-sensitive membrane, which is used to detect where a student or teacher touches the board. Applying pressure to the front surface (by using a pen or a finger) registers a contact point that is used as input to the interactive whiteboard software. Whiteboards based on resistive technology do not require special pens to write on the board, as ones finger can also be used.

2. Electro-Magnetic pick-up - These whiteboards are similar to traditional whiteboards in that they are quite rigid to the touch. The pens used with them emit a small magnetic field, which the board detects on pen impact or movement, and this information is then used as input to the computer running the interactive whiteboard software.

3. Infrared scanning - By attaching infrared scanning devices to an existing ordinary whiteboard or flat surface the board is transformed from an ordinary whiteboard or surface to act as an interactive whiteboard. These scanning devices are light and portable and can be used with different types and sizes of ordinary whiteboards. Tracking of colour and patterns is based upon using special encoded pens, each of which has a uniquely encoded reflective collar that the board uses to identify its colour and position. Infrared touch uses light emitting diodes and sensors that are embedded in a bezel around the display and emit and detect rows and columns of infrared light across the face of the display. This creates an invisible grid of infrared beams and on the opposite side of the display from the emitters, photodetectors identify touch when the plane of the grid is broken by a finger touch.



● Resistive membrane Technology	54	● Infrared Scanning Technology	30
● Electro-magnetic pick-up	12	● Wiimote technology	1.5
● others	0.5		

Fig 2.3.2 Different technologies used in market.

Boards range in size from 35 inches (diagonal) to approx 80 inches (diagonal) and normally cost in the region of Rs.1,17,000 - Rs.3,12,000, including controlling software. Some boards can be fitted to a moveable stand, costing in the region of Rs.20,000 enabling access in different locations, or can be wall-mounted. Generally the kit to do so is included in the price; however, installation and configuration of the board and projector could add another Rs.40,000 to the overall price. If the school does not already have a digital projector, this can make the quite expensive as suitable projector prices begin at approx. Rs43,000 plus up to Rs.25,000 for a replacement bulb. Some suppliers provide integrated, wall-mounted projectors, bringing the total cost to approx Rs.400,000 (including installation and training). Given the price variations it is essential to seek best value by obtaining quotations from at least three providers. Note that these indicative prices do not include the price of the computer itself. Infrared Interactive whiteboards are the least expensive option. Such systems can be purchased for under Rs.78,000 with the school providing its own standard board and projector.

CHAPTER - 3
BLOCK DIAGRAM AND SYSTEM MODEL

3.1 BLOCK DIAGRAM AND DESCRIPTION

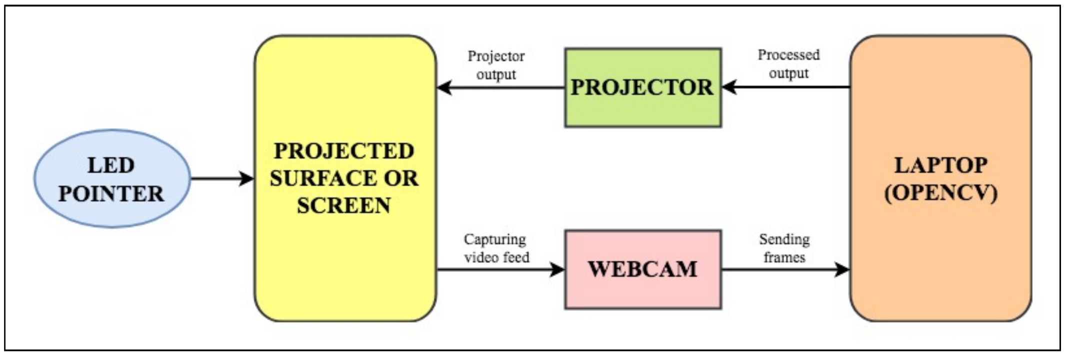


Fig. 3.1 System block diagram

1. LED Pen: To provide a LED pointer equivalent of a mouse click
2. Projected screen
3. Projector
4. Webcam: For capturing and sending video frames.
5. Computer: For processing received frames for further filtering and calibration using OpenCV.

The proposed model requires an LED pen to simulate the mouse events. The LED pen consists of a button switch which switches the LED on and off. One click of the LED button is equivalent to one left click. Hence drag and drop functions can also be performed by keeping the LED button pressed. The Webcam will detect the movement of the LED pointer and the output will be given to the computer for processing. Various filtering algorithms are used to detect the LED blob. Hence the real time location of the mouse pointer is obtained. These locations are mapped and given to mouse apis which will click on that exact location. The output will be projected with the help of the projector on the desired screen or surface.

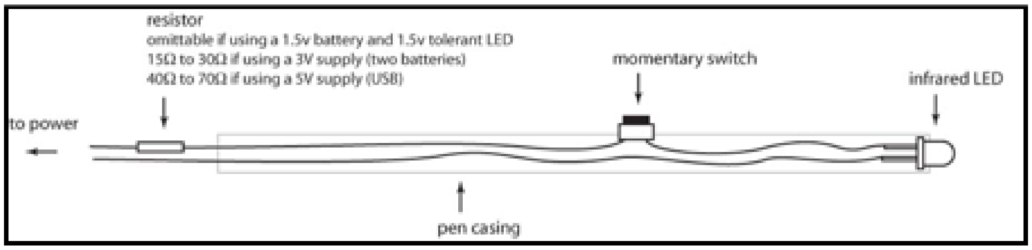
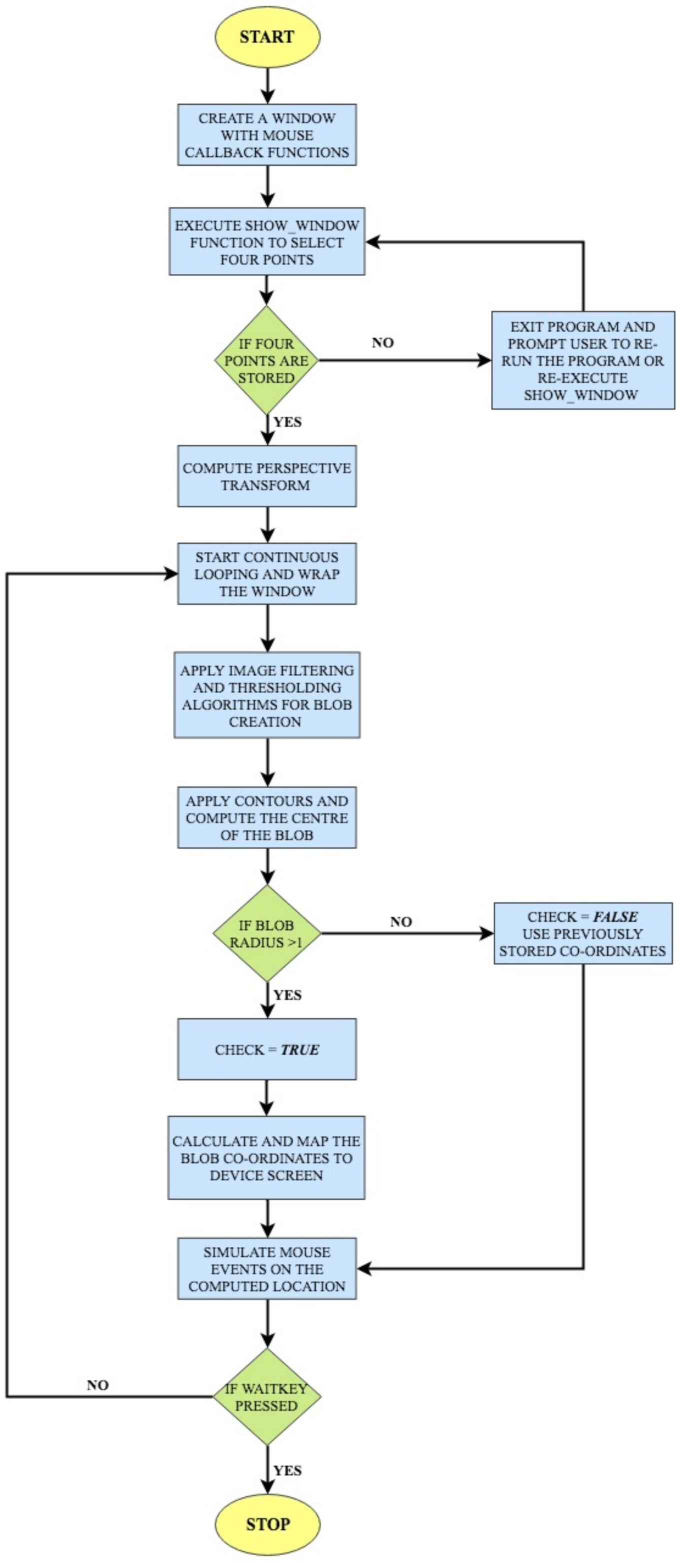

resistor
omittable if using a 1.5v battery and 1.5v tolerant LED
15Ω to 30Ω if using a 3V supply (two batteries)
40Ω to 70Ω if using a 5V supply (USB)
momentary switch
infrared LED
to power
pen casing

Fig 3.2 Construction of LED pen.

The led pen can be constructed using a simple circuit with momentary switch as shown in figure 3.2. The standard webcam processes the frames at a speed of 30 fps. The performance of the proposed model can be improved by increasing the fps of the webcam. This can be done by using a buffer for preprocessing frames or using a high fps webcam or camera.

3.2 FLOWCHART AND SYSTEM MODEL


START
CREATE A WINDOW WITH MOUSE CALLBACK FUNCTIONS
EXECUTE SHOW_WINDOW FUNCTION TO SELECT FOUR POINTS
IF FOUR POINTS ARE STORED
NO
EXIT PROGRAM AND PROMPT USER TO RE-RUN THE PROGRAM OR RE-EXECUTE SHOW_WINDOW
YES
COMPUTE PERSPECTIVE TRANSFORM
START CONTINUOUS LOOPING AND WRAP THE WINDOW
APPLY IMAGE FILTERING AND THRESHOLDING ALGORITHMS FOR BLOB CREATION
APPLY CONTOURS AND COMPUTE THE CENTRE OF THE BLOB
IF BLOB RADIUS >1
NO
CHECK = FALSE USE PREVIOUSLY STORED CO-ORDINATES
YES
CHECK = TRUE
CALCULATE AND MAP THE BLOB CO-ORDINATES TO DEVICE SCREEN
SIMULATE MOUSE EVENTS ON THE COMPUTED LOCATION
IF WAITKEY PRESSED
NO
YES
STOP

3.3 SYSTEM ARRANGEMENT

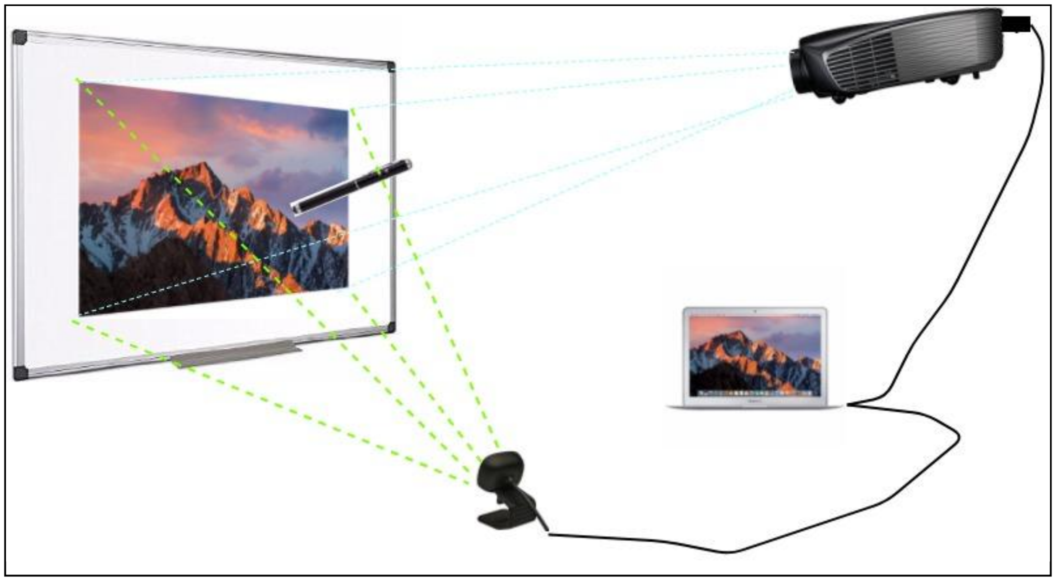


Fig. 3.3 System Arrangement

Fig. 3.3 shows the typical system arrangement of this project. The projector is mounted on top which projects on the surface. This projector remains fixed. The user will use the LED pen on the projected surface. The Laptop or Computer is connected to the projector via a cable. The webcam is also fixed and is placed at angle with respect to the screen such that the entire screen is visible by the webcam. The most suitable angle is 45 degree. This webcam is connected to the laptop for sending the captured video frames. Nowadays many wireless webcams are also used. For this system quality and resolution of the webcam does not matter, hence even a low cost webcam can be used. Another alternative is the internal laptop webcam. The computer will consist of the software which will convert the LED clicks into mouse clicks.

CHAPTER - 4
IMAGE PROCESSING

4.1 INTRODUCTION TO OPENCV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. OpenCV still retains a less comprehensive though extensive older C-interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.

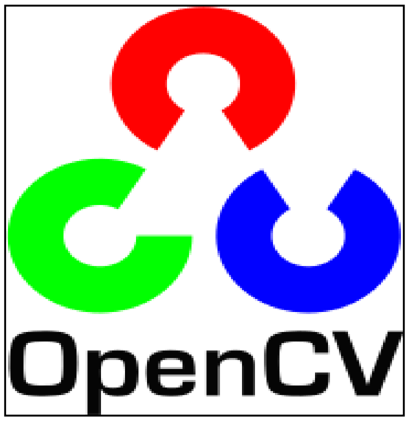


Fig. 4.1 OpenCV logo

The following modules are available:

A. core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

B. imgproc - an image processing module that includes linear and non-linear image filtering, geometric image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

C. video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

D. calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

E. features2d - salient feature detectors, descriptors, and descriptor matchers.

F. objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

G. highgui - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.

H. gpu - GPU-accelerated algorithms from different OpenCV modules.

Some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others. OpenCV runs on the following desktop operating systems: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

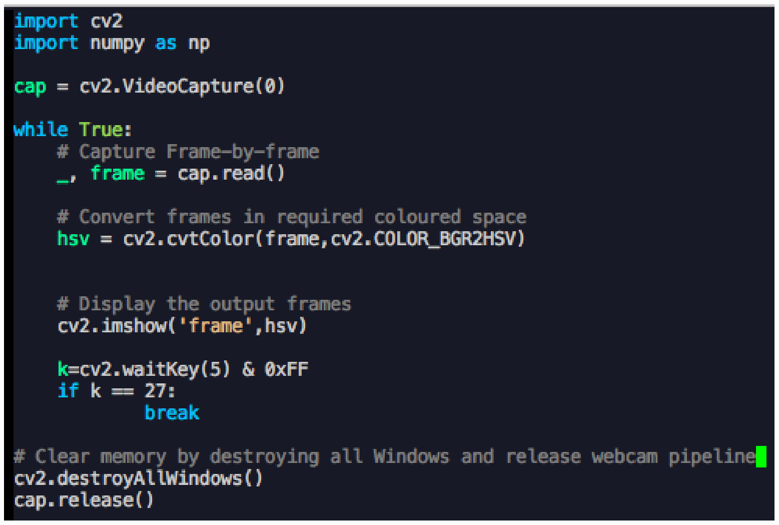
4.2 OPENCV APPLICATIONS

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Motion tracking
- Augmented reality

4.3 GETTING STARTED WITH VIDEO PROCESSING

Often, we have to capture live stream with camera. OpenCV provides a very simple interface to this. Fig 4.3 shows an example of capturing a video and converting it into grayscale using OpenCV algorithms.



```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    # Capture Frame-by-frame
    _, frame = cap.read()

    # Convert frames in required coloured space
    hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)


    # Display the output frames
    cv2.imshow('frame',hsv)

    k=cv2.waitKey(5) & 0xFF
    if k == 27:
            break

# Clear memory by destroying all Windows and release webcam pipeline
cv2.destroyAllWindows()
cap.release()
```

Fig. 4.3 Video processing example

To capture a video, you need to create a 'VideoCapture' object. Hence the video is continuously captured and stored in a variable called 'cap'. Its argument can be either the device index or the name of a video file. Device index is just the number to specify the active camera. Normally one camera will be connected. So the device index will be 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame.

‘cap.read()’ returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value. This is stored in a variable called ‘frame’.‘cv2.cvtColor’ is a colour conversion algorithm with accepts the frame and converts it into gray.

For displaying the converted frame we use the ‘imshow’ function which will create a seperate window and output the result.‘waitkey’ is used to exit out of the while loop and terminate the program if a specific key is pressed. In the above example the waitkey is the ‘q’ button. Therefore once waitkey is pressed, it will exit out of the while loop and all the capture will be released and all the created windows by imshow will be destroyed.

CHAPTER - 5
SOFTWARE ALGORITHMS

5.1 INTRODUCTION

The entire software code can be divided in into various algorithm or functions. The different algorithms have been tested separately and in the end all of them have been combined to give us one working system. The order in which these algorithms are processed remains fixed and cannot be changed. They are the fundamental pillars of the project. The main 5 algorithms are :

1. Perspective Transformation
2. Image Filtering
3. Contouring
4. Coordinate Mapping
5. Mouse Click

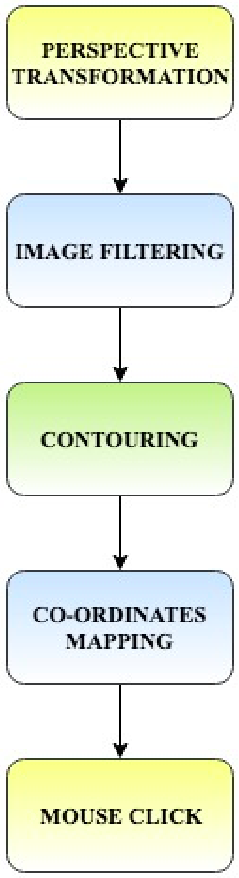


Fig. 5.1 Software Algorithms

The input will be the captured video frames from the webcam. This can be done using VideoCapture function of OpenCV. Then this captured frames undergo these 5 algorithms in the same sequence. The detailed description of each algorithm is given in the next section.

5.2 PERSPECTIVE TRANSFORMATION

The selected area of the captured image is not suitable for processing as the webcam is not placed straight to the screen. So, the touch point coordinates is different from the real world coordinates for the perspective distortion of the image. But for the proper implementation of the system, it needs to establish a relationship between the pixels in the webcam and the pixels in the touch point. So perspective transform algorithm is applied to the selected area to overcome the problem. Projective geometry transformation relates the coordinates of a point in a scene to the coordinates of its projection onto an image plane.

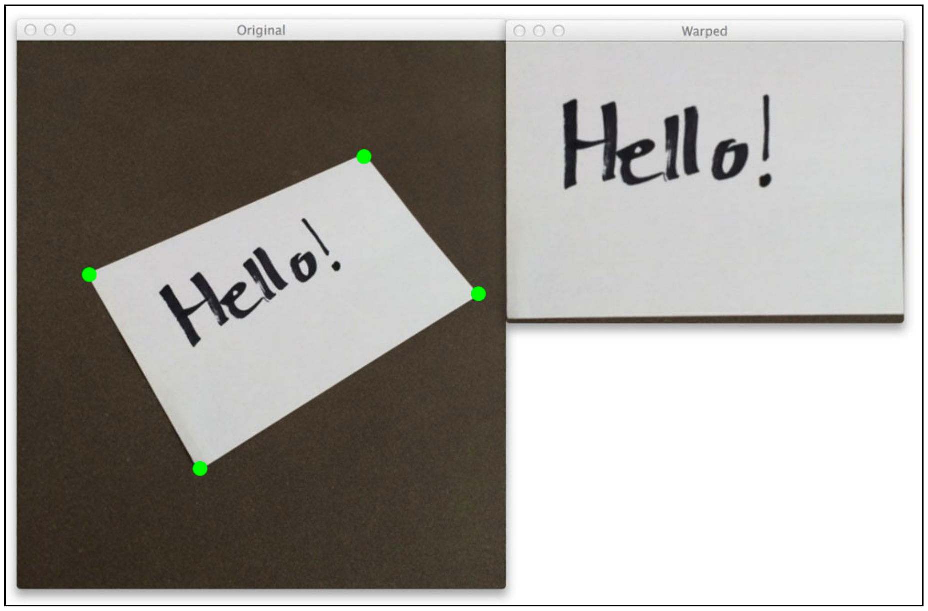


Fig. 5.2 (a) Perspective Transform demonstration

Fig. 5.2(a) shows the perspective transform performed on a blank sheet of paper. The camera will be placed at an angle to view the screen, hence we will select the main desired four points. The perspective transformation algorithm will warp those four points and discard the rest of the screen. They does not change the image content but deforms the pixel grid and maps this deformed grid to the destination image.

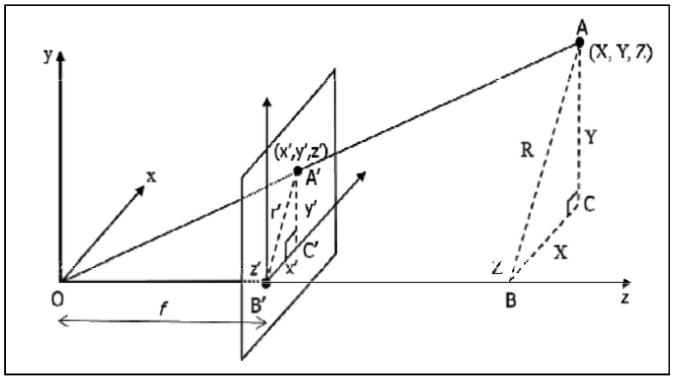


Fig. 5.2 (b) Mathematical analysis of perspective transformation

From OA′B′ and OAB triangles:

$$\frac{f}{z} = \frac{r'}{R}$$

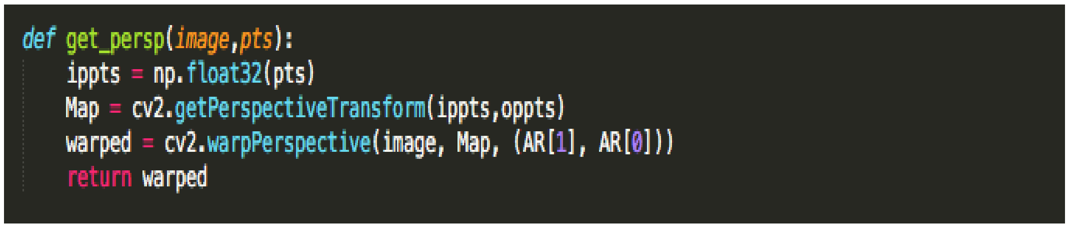
From A′B′C′ and ABC triangles:

$$\frac{x'}{X} = \frac{y'}{Y} = \frac{r'}{R}$$

So, the perspective projection equations:

$$x' = \frac{Xf}{Z}\ ;\ y' = \frac{Yf}{Z}\ ;\ z' = f$$

Within the camera coordinate system the perspective projection of a scene point onto the image plane is described by the above equations. After performing the algorithm, a linear relation between the real screen and the virtual screen is established.



```
def get_persp(image,pts):
    ippts = np.float32(pts)
    Map = cv2.getPerspectiveTransform(ippts,oppts)
    warped = cv2.warpPerspective(image, Map, (AR[1], AR[0]))
    return warped
```

Fig 5.2 (c) Perspective Transform functions

Fig 5.2 (c) is the the snippet of the Perspective Transformation algorithm. The input 4 points selected by the user are stored in 'ippts'. These input points are given to the 'cv2.getPerspectiveTransform' function. This function calculates a perspective transform from four pairs of the corresponding points. This corresponding points are nothing but the co-ordinates of the screen measured with the help of its aspect ratio. The two parameters given to this function are coordinates of quadrangle vertices in the source image and coordinates of the corresponding quadrangle vertices in the destination image. The function calculates the 3×3 matrix of a perspective transform so that:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

The 'cv2.warpPerspective' will apply the perspective transform to the input image or video. The parameters for this function are the input image, the perspective transform calculated by the 'cv2.getPerspectiveTransform' function and the desired aspect ratio.Thus the perspective transformation maps the 3D world co-ordinates to the required 2D co-ordinate system.

5.3 IMAGE FILTERING

Image filtering is necessary for detecting the LED pointer. The image frames captured by the camera should be filtered in such a way that the LED pointer on the surface will appear as a blob. For obtaining a blob various filtering functions are applied.
These functions are :

1. Warping
2. Blurring
3. HSV conversion
4. Masking
5. Otsu Thresholding

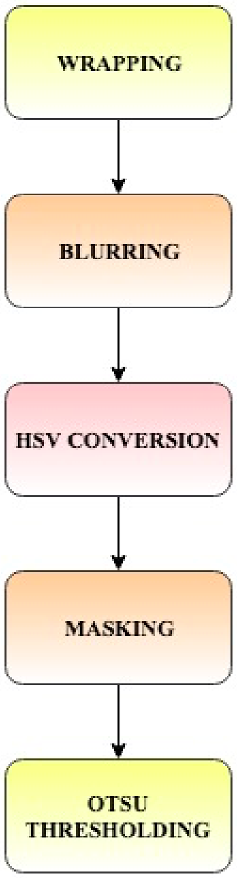


Fig. 5.3 (a) Filtering sequence

The captured image by the webcam is a RGB color image which takes longer time to process. As the stylus is using only bright red light as a touch point so there is no need of other color spaces. In order to extract the red light, the captured image is converted to HSV color model which defines a color space in terms of color type, color intensity and color brightness. In our proposed system, we are differentiating the bright red LED by adjusting the HSV values of the image. It provides only the area of the image which contains bright red color ignoring the other component of color.

The brightness and contrast of the image is adjusted so that the touch point can easily be differentiated from the background image. For adjusting the contrast and brightness of the capture image the following formula is used:

$$g(x) = \alpha f(x) + \beta$$

The parameters α > 0 and βare the gain and bias parameters for controlling the contrast and brightness respectively. Considering f(x) as the source image pixels and g(x)as the output image pixels, the formula can also be written as:

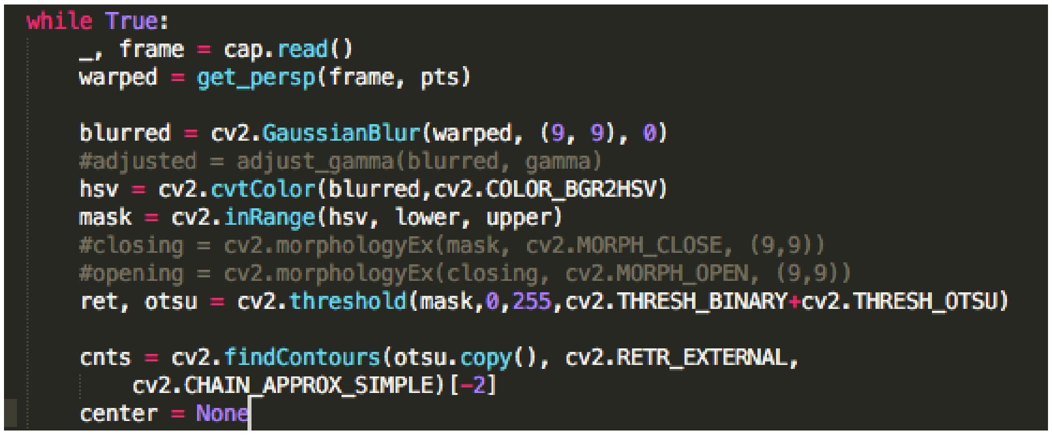
$$g(i,j) = \alpha f(i,j) + \beta$$

where i and j indicates that the pixel is located in the i-th row and j-th column.

The adjusted image is converted to a HSV colorspace vis cv2.cvtColor function. The HSV colorspace is used for its accuracy in various lighting conditions. Masking function will convert the HSV image into binary 1's and 0's, hence the output of mask will be a resultant of white and black. The values close to 255 are rounded off to 255 and the values closer to 0 are rounded off to 0. After proper image adjustments, it is required to reduce the surrounding noises. The use of the Gaussian kernel for smoothing has become extremely popular in image processing application. So, the adjusted image is further smoothed for a better accuracy using a 3×3 Gaussian filter. The Gaussian function (2D) can be expressed as the product of two one- dimensional functions in each coordinate axis.

$$g(x,y) = \frac{1}{2\Pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. After that adaptive threshold is applied to the binary image which creates a blob of the LED to act as pointer location and this blob provides a clear touch area for further processing.

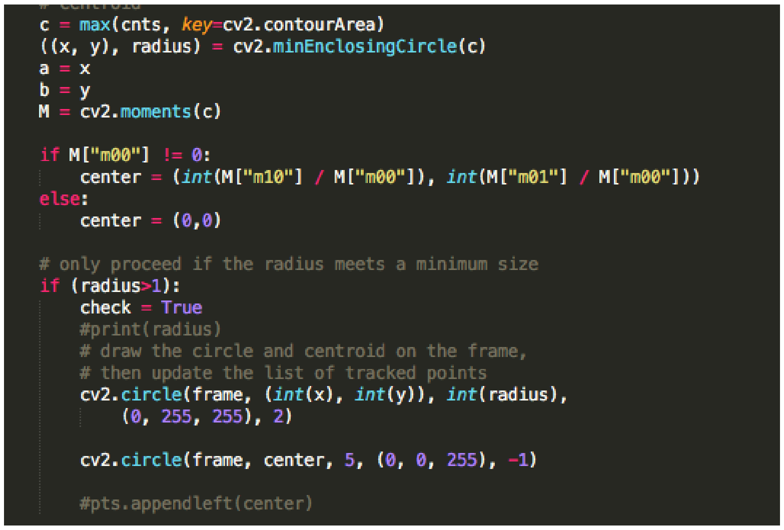


```
while True:
    _, frame = cap.read()
    warped = get_persp(frame, pts)

    blurred = cv2.GaussianBlur(warped, (9, 9), 0)
    #adjusted = adjust_gamma(blurred, gamma)
    hsv = cv2.cvtColor(blurred,cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower, upper)
    #closing = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, (9,9))
    #opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, (9,9))
    ret, otsu = cv2.threshold(mask,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    cnts = cv2.findContours(otsu.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)[-2]
    center = None
```

Fig. 5.3 (b) Image Filtering Algorithm

5.4 CONTOURING

The touch area by the LED pointer can easily be differentiated and tracked due to the fact that the LED light intensity is brighter than the background screen. The contours of the binary image are found using the contour finding algorithm of OpenCV. Around the touch point where the stylus was touched there might be some small noise areas. To eliminate the noise, only the largest contour area is considered as the touch point. The center coordinate is then calculated using moments calculation formula.



```
c = max(cnts, key=cv2.contourArea)
((x, y), radius) = cv2.minEnclosingCircle(c)
a = x
b = y
M = cv2.moments(c)

if M["m00"] != 0:
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
else:
    center = (0,0)

# only proceed if the radius meets a minimum size
if (radius>1):
    check = True
    #print(radius)
    # draw the circle and centroid on the frame,
    # then update the list of tracked points
    cv2.circle(frame, (int(x), int(y)), int(radius),
        (0, 255, 255), 2)

    cv2.circle(frame, center, 5, (0, 0, 255), -1)

    #pts.appendleft(center)
```

Fig. 5.4 (a) Contouring algorithm

Image moments help to calculate some features like center of mass of the object, area of the object etc. From this moments useful data can be extracted like area, centroid etc. Hence using these moments the center of the LED blob is calculated and the radius of blob is measured. After this the radius is checked. Normally some white noise having radius less than 1 might be present. To eliminate this we can give a condition to accept blob only if radius is greater than 1. When this condition is satisfied a circle is drawn around the contour and centre co-ordinates are computed and stored in x and y variable.

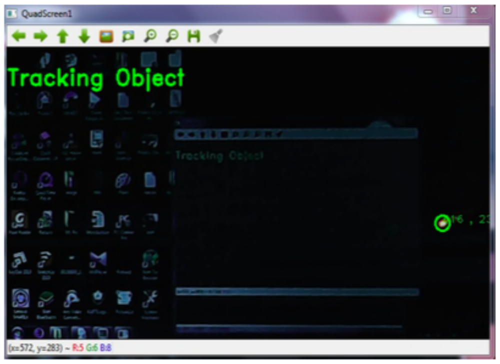


Fig. 5.4(b) LED blob tracking

5.5 COORDINATE MAPPING

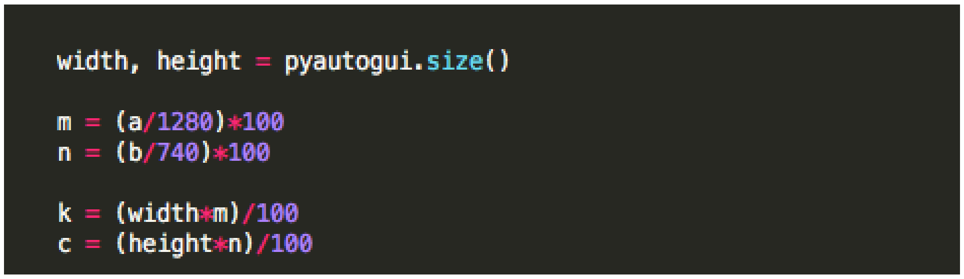
Coordinate mapping is essential as the projected screen size and the laptop or computer screen size is not the same. Hence a click generated at a point on the projected screen has to match the point on the equivalent desktop.

The coordinate is mapped to screen coordinates to get the exact location of the touch coordinate using the following formula:

$$x = \frac{(P_x \times X)}{I_x}$$

$$y = \frac{(P_y \times Y)}{I_y}$$

where (x,y) are screen coordinates, , are the touch point coordinates on the processed image, are the image resolution and X,Y are the screen resolution. Then the corresponding coordinate is passed to windows default mouse events to simulate mouse actions.



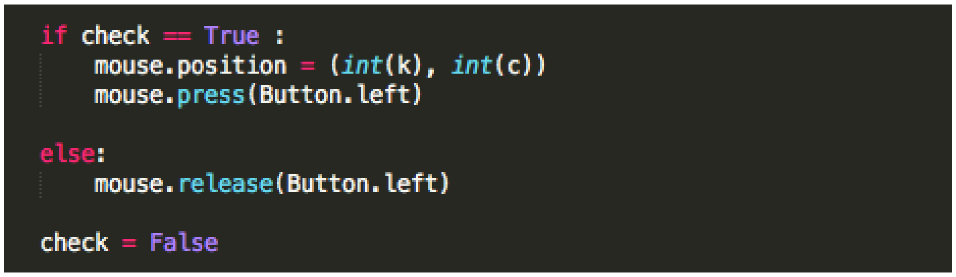
```
width, height = pyautogui.size()

m = (a/1280)*100
n = (b/740)*100

k = (width*m)/100
c = (height*n)/100
```

Fig. 5.5 Mapping algorithm

Fig. 5.5 shows the implementation of Mapping algorithm. The width and height of the window created is calculated using 'pyautogui.size()'. (a,b) are the contour centroid coordinated. (k,c) are the final mapped coordinates obtained. These coordinates can be given to the mouse api which will perform a mouse click event. This is an essential part of the program. If this algorithm is not implemented then the LED click generated by the user on the projected surface will not be the equivalent of the desktop screen.

5.6 MOUSE CLICK

Various mouse clicking apis are available like pyautogui, win32api, pynput. The one used in this algorithm is pynput which is compatible with Windows, MacOS, Linux etc.



```
if check == True :
    mouse.position = (int(k), int(c))
    mouse.press(Button.left)

else:
    mouse.release(Button.left)

check = False
```

Fig. 5.6 Mouse Clicking algorithm

Earlier a ‘check’ variable was initialized which was made true incase the blob radius was greater than 1. This ‘check’ variable is verified in order to implement a mouse click. If ‘check’ is equal to True the ‘mouse.position’ will move the mouse cursor to the calculated mapped coordinates i.e. (k,c). Then ‘mouse.press’ function will perform a left click. These two functions will be implemented only if the LED button is pressed.

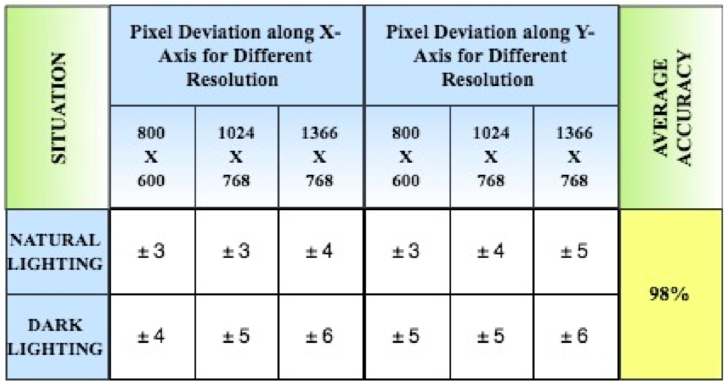
As soon as the click button is released, the radius will become less than 0 as no blob will be detected on the projected surface. Hence now ‘check’ variable will be equal to False and the ‘if’ condition shown in Fig 5.6 will not be satisfied. Therefore the ‘else’ condition will be executed which will be the ‘mouse.release’ function. At the end the ‘check’ value is again reset to false.

CHAPTER - 6
RESULT

6.1 RESULTS

The whole system is implemented by using only a webcam and a LED light. The system works with various ambient light conditions. We implemented the system in daylight and at dark light. But it gives higher accuracy in both conditions.Experiment is made in different ambient light conditions. The light effects are reduced by controlling the contrast and brightness of the captured image. As compared to the IR touch screen system which is affected by the sunlight, this system is more accurate and can simulate mouse events very efficiently.

The image processing is done on a laptop having Intel Core i5 processor 2.60 GHz, 4GB memory running on Windows 7 64 bit Operating system. It can also be supported on various systems like OS-X and other linux or unix systems. In our experiment, the display screen of resolution 1024×768 is converted to touch screen with the captured image of resolution 640×480. The HSV values and threshold values are adjusted properly for working with all environment light condition.



SITUATION	Pixel Deviation along X-Axis for Different Resolution			Pixel Deviation along Y-Axis for Different Resolution			AVERAGE ACCURACY
	800 X 600	1024 X 768	1366 X 768	800 X 600	1024 X 768	1366 X 768	
NATURAL LIGHTING	±3	±3	±4	±3	±4	±5	98%
DARK LIGHTING	±4	±5	±6	±5	±5	±6	

Table - 6.1 Mouse Clicking algorithm

We found the average pixel deviation of ±4 pixels along x-axis and deviation of ±5 pixels along the y-axis which can be considered as zero deviation with respect to the entire display screen. The overall accuracy for the implemented system is 98%.

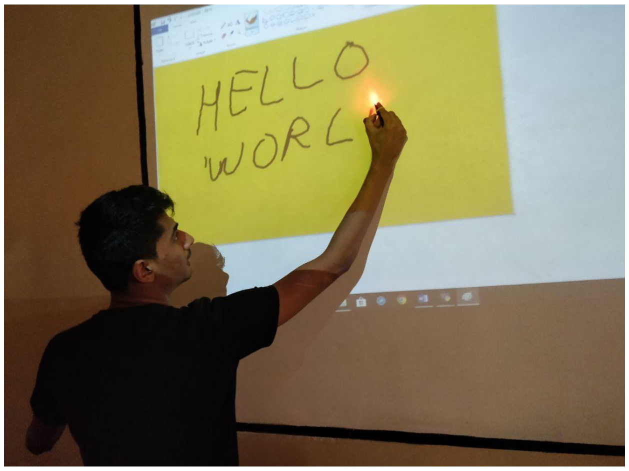


Fig.6.1 Implementation and final result

CHAPTER - 7
ADVANTAGES, DISADVANTAGES AND APPLICATIONS

7.1 ADVANTAGES

1 - Increase engagement in collaborative productions

Instead of spending 30 minutes on one-way presentations being shared across a PowerPoint presentation this system allows users to engage with the information being discussed. Files can be easily shared, accessed, edited, and saved on the spot. Meeting leaders can emphasize things in real-time—making changes to whatever topic is at hand as feedback is received from coworkers.

2 - Adding the element of touch

Touch interface adds a new perspective of interaction. Users can communicate with their audiences in an interactive way which helps for better understanding. Using this touch interface the user will be able to control the large projected screens on the spot. The teachers can stress on particular parts by underlining, circling and annotating with digital ink of different colors to improve students understanding of the lessons.

3 - Effectively annotate documents

Unlike document sharing, users can make persistent and effective changes to documents during the session. Annotation tools can allow for 3D modeling, estimating, hyperlinking, video links, and other applications that can improve communication and make documents stronger. Writing is clear and concise, and harder to misconstrue.

4 - Low-cost system

The total cost of the system is very less because the project is completely software based. The hardware required is a webcam, laptop, projector which every user already have. The IWB in the market cost around Rs. 1,00,000 to Rs. 3,00,000 hence compared to this our system is very cost efficient.

5 - Easily Integratable

The system software can work on various desktop operating system (Windows, MacOS, Linux). Hence it is platform independent. Hence by simply running the program on a pc any projected surface can be converted into a touch interface. Maintenance is also negligible.

6 - Replacing Traditional Blackboards

The traditional blackboards are less interactive and make use of chalk and duster. Many people are allergic to chalk dust which causes respiratory problems. Our system will eliminate the use of chalk and duster hence saving a lot of time and effort of the teachers. It will provide a clean environment and enhance the overall development of the classroom.

7 - Low maintenance

The system is low maintenance as the hardware requirement is low.

7.2 DISADVANTAGES

1 - LED pointer should be in the line of sight of the camera

If the LED pointer is not visible by the camera then no blob will be detected and hence no click will be generated. No object should be blocking the camera from viewing the projected surface. To eliminate this problem the camera can be mounted on top of the projector which will avoid the ground interferences.

2 - Camera has to be fixed

The camera cannot be moved as the screen is calibrated for that position. A little deflection of the camera will distort the accuracy of the click. This can be solved by using better calibrating algorithms rather than the 4 point selection perspective transformation.

7.3 APPLICATIONS

Wherever a normal projector is used our system can be integrated with to make it into a touch interface. The different applications are:

1 - Educational Purposes

This project can be used in schools and colleges for teaching purposes. The advantage of it being cost efficient will make it affordable for many developing countries to use this technology. Hence it can be installed in every classroom

2 - Seminar Halls and Corporate Boardrooms

Usually long presentations tend to be boring if they are not interactive. This system can be used in seminar halls or boardrooms for presenting to an audience which will increase the level of interactiveness by adding annotations and 3D softwares.

3 - Sport Training Facilities

Various sports like Football, Cricket, Hockey etc require team and game planning. In order for better understanding, the coaching staff can use this technology to show different field placements, tactics, analysis etc.

CHAPTER - 8
CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

In this project we have designed and implemented a vision- based touch screen computer interaction system using only a webcam and a LED pointer. The proposed system provides a natural, cost-effective and comfortable human computer interaction interface without use of any complex hardware devices. We created an LED pointer with which the user would register clicks. The webcam view was calibrated using perspective transform and the LED blob was tracked using OpenCV filtering, contouring and mapping functions. Each button click was converted into a mouse click using the pynput mouse api. At the end, all the modules were combined and integrated to produce an interactive touch interface for the user. Compared to the other touch screen interface techniques this system does not lack efficiency in ambient light condition. This makes it usable at any time of the day and in any given environment. It requires less time to setup and is very easy to use. This system can be used for interactive learning, business and industrial purposes because of its high accuracy and real time response.

8.2 FUTURE SCOPE

We are still developing the system and in our future work, we are planning to improve the feasibility of the system by including multi-touch gesture facility for zoom in, zoom out, scrolling up, scrolling down, and handwriting recognition facility which will make it more fascinating. Apart from bringing convenience and flexibility in education, use of interactive projector increases teacher-student engagement and boosts the performance in the long run. Interactive projectors are considered as superior teaching tool. It's even important with the shift in teaching/board rooms to more student/employees involvement. We can create a stand-alone module having its own Operating System and hence will be portable. Nowadays many mobile phones have projector technology, this system can be integrated to make it more interactive.

CHAPTER - 9
REFERENCES

REFERENCES

[1] An approach to convert Conventional Screen into touch screen using Image Processing by Md. Hanif Ali Sohag, Mst. Mahmuda Khatun, and Mohiuddin Ahmad (2015)

[2] Low-Cost Interactive Whiteboard Using the Kinect – By Shuai Zhang, Wenrui He, Qiao Yu, and Xiaojuan Zheng. (2011)

[3] A method for controlling mouse movement using a real time camera- by Hajoon Park. (2011)

[4] An improved interactive whiteboard system: A new design and an Egronomic Stylus - by Koray Yucel, Nevzat Orhan, Gizem Misirli, Gözde Bal, Yasar Guneri Sahin. (2010)

[5] Hacking the Nintendo Wii Remote - by Johnny Chung Lee. (2008)

[6] Hand Mouse: Real Time Motion Detection System Based on Analysis of Finger Blobs - by Ibrahim Furkan Ince, Manuel Socarras-Garzon, Tae-Cheon Yang. (2010)

[7] The Interactive Whiteboards (IWB) - Uses, Benefits and Challenges by Thierry Karsenti