AI VIRTUAL MOUSE USING HAND GESTURES

A MINI PROJECT REPORT

Submitted by

M H MOGDOOM KHAN SAHIB	110120104022
A MOHAMED BASITH ALI	110120104024
A S MOHAMED MUBEEN	110120104028
K M D MOHAMED RIYSATH	110120104038

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING

ANNA UNIVERSITY:: CHENNAI 600 025

MAY-2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that the project report "**AI VIRTUAL MOUSE USING HAND GESTURES** is a bonafide work of **M H MOGDOOM KHAN SAHIB (110120104022), A MOHAMED BASITH ALI (110120104024), A.S. MOHAMED MUBEEN (110120104028) AND K M D MOHAMED RIYASATH (110120104038)** who carried out the project work under my supervision

SIGNATURE	**SIGNATURE**
MRS. G. SULTHANA BEGAM	**MR. S.N MOHAMED IRFAN**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science an Engineering	**Department of Computer Science an Engineering**
Aalim Muhammad Salegh College of engineering	**Aalim Muhammad Salegh College of engineering**
Chennai 600055	**Chennai 600055**

Submitted for the University practical examination held on.....at Aalim Muhammed Salegh College of Engineering.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost we would like to thank the God of Almighty who is our refuge and strength. We would like to express our heartfelt thanks to our **Beloved Parents** who sacrifice their presence for our better future. We are very much indebted to thank our college Founder Alhaj**. Dr. S.M.SHAIKNURUDDIN**, Chairperson JANABA ALHAJIYANI**M.S. HABIBUNNISA**, Aalim Muhammed Salegh Group of Educational Institutions, Honorable Secretary&Correspondent JANAB ALHAJI **S.SEGU JAMALUDEEN,** Aalim Muhammed Salegh Group of Educational Institutions for providing necessary facilities all through the course. We take this opportunity to put forth our deep sense of gratitude to our beloved Principal **Prof. Dr. S. SATHISH** for granting permission to undertake the project.

We also express our gratitude to **Mrs. G. SULTHANA BEGAM,** Head of the Department, Computer Science And Engineering, Aalim Muhammed Salegh College of Engineering, Chennai, for his involvement and constant support, valuable guidance to effectively complete our project in our college We would like to express our thanks to our project guide **Mr.S.NMOHAMED IRFAN,** Assistant Professor, Computer Science And Engineering, Aalim Muhammed Salegh College of Engineering who persuaded us to take on this project and never ceased to lend his encouragement and support. Finally we take immense pleasure to thank our family members, faculties and friends for their constant support and encouragement in doing our project.

ABSTRACT

This system enables users to perform mouse actions through hand movements, eliminating the need for traditional input devices.The system begins with the detection of hand gestures, leveraging hand landmark estimation algorithms to accurately locate and track the user's hand in real-time. Through a combination of feature extraction, gesture recognition, and mapping, the system associates recognized hand gestures with specific mouse actions, such as cursor movement, clicking, and scrolling.The integration and interaction aspects ensure seamless integration of the system into the user's computing environment, enabling effortless and intuitive control over the virtual mouse. The system can be integrated with various hardware devices, such as cameras or sensors, and can accommodate different operating systems and software applications.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

In computing terminology mouse is a pointing device that detects two dimensional movements in relation to the medium. This movement is translated into the movement of a pointer on a screen, allowing the user to manage the Graphical User Interface (GUI) on a computer. There are several other types of mouses that have already existed in modern technology, namely the mechanical mouse, which uses a hard rubber ball that rolls around when the mouse is moved to identify movement. Years later, the optical mouse was introduced, which replaced the hard rubber ball with an LED sensor that detects table top movement and then delivers the data to the computer to be processed. lately the laser mouse was invented to enhance the precision of movement with the shortest hand movement, overcoming the challenges of the optical mouse, which has difficulty tracking high-gloss surfaces. However, no matter how accurate it is, there are still physical and technical drawbacks within the mouse itself. A computer mouse, for example, is an edible hardware device because it requires replacement in the long run, either because the mouse buttons have deteriorated, resulting in inappropriate clicks, or because the entire mouse is no longer encountered by the computer.

Despite its shortcomings, computer technology continues to advance, and the relevance of human-computer interactions grows as well. Since the debut of a mobile device with touch screen technology, the world has begun to demand that the same technology be implemented on other technical devices, including desktop systems. Even while touch screen technology for desktop computers is currently available, the cost can be expensive Therefore, a virtual human computer interaction device that uses a webcam or other picture capturing devices to replace the actual mouse or keyboard could be an alternative to the touch screen.

2.SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

Computer Desktop or Laptop The computer desktop or a laptop will be utilized to run the visual software in order to display what webcam had captured. A notebook which is a small, lightweight and inexpensive laptop computer is proposed to increase mobility.

System will be using

- **Processor**: Core2Duo
- **Main Memory**: 4GB RAM
- **Hard Disk**: 320GB
- **Display**: 14" Inch
- **Camera:**

  Camera is utilized for image processing; the camera will continuously be taking image in order for the program to process the image and find pixel position.

2.2 SOFTWARE REQUIREMENTS

- Python (version 3.7 or above) – Programming language
- OpenCV (Version 4.5. 3.56) – For image processing
- NumPy (version 1.19.2) -- For frame collection in an array
- Mediapipe— For Hand Tracking
- PyautoGui -- For controlling the mouse movement and click

3.SYSTEM ANALYSIS

3.1 EXSISTING SYSTEM

- The existing system consists of the generic mouse and trackpad system of monitor controlling and the non-availability of a hand gesture system. The remote accessing of monitor screen using the hand gesture is unavailable.

- The existing virtual mouse control system consists of the simple mouse

- operations using the hand recognition system, where we could perform the basic mouse operation like mouse pointer control, left click, right click, drag etc. The further use of the hand recognition is not been made use of.

3.2 PROPOSED SYSTEM

- Using our project, we could make use of the laptop or web-cam and by recognizing the hand gesture we could control mouse and perform basic operations like mouse pointer controlling, select and deselect using left click, and a quick access feature for file transfer between the systems connected via network LAN cable.
- The project done is a “Zero Cost” hand recognition system for laptops, which uses simple algorithms to determine the hand, hand movements and by assigning an action for each movement. But we have mainly concentrated on the mouse pointing and clicking actions along with an action for the file transfer between connected systems by hand action and the movements

4.SYSTEM DESCRIPTION

4.1 HAND DETECTION

Hand detection is a fundamental component of the AI virtual mouse using hand gestures system. It is responsible for identifying and localizing the presence of a hand in the video stream or image input. By accurately detecting the hand, the system can proceed with subsequent steps such as hand landmark estimation and gesture recognition.

Skin Color-based Detection: This approach relies on the distinctive skin color of hands to identify hand regions. It involves applying color thresholds to the image or video frames to isolate pixels that belong to the skin color range.

Background Subtraction: By capturing an initial frame without the hand present, subsequent frames can be compared to identify the difference caused by the presence of a hand. This method can help extract hand regions from the background effectively.

Once the hand regions are detected, employ techniques such as object tracking algorithms to track the hand across frames, ensuring smooth and continuous hand detection.

Implementing hand detection typically involves using computer vision libraries and frameworks such as OpenCV, MediaPipe, or TensorFlow. These libraries provide pre-built functions, models, and APIs for hand detection, making it easier to develop the AI virtual mouse system.

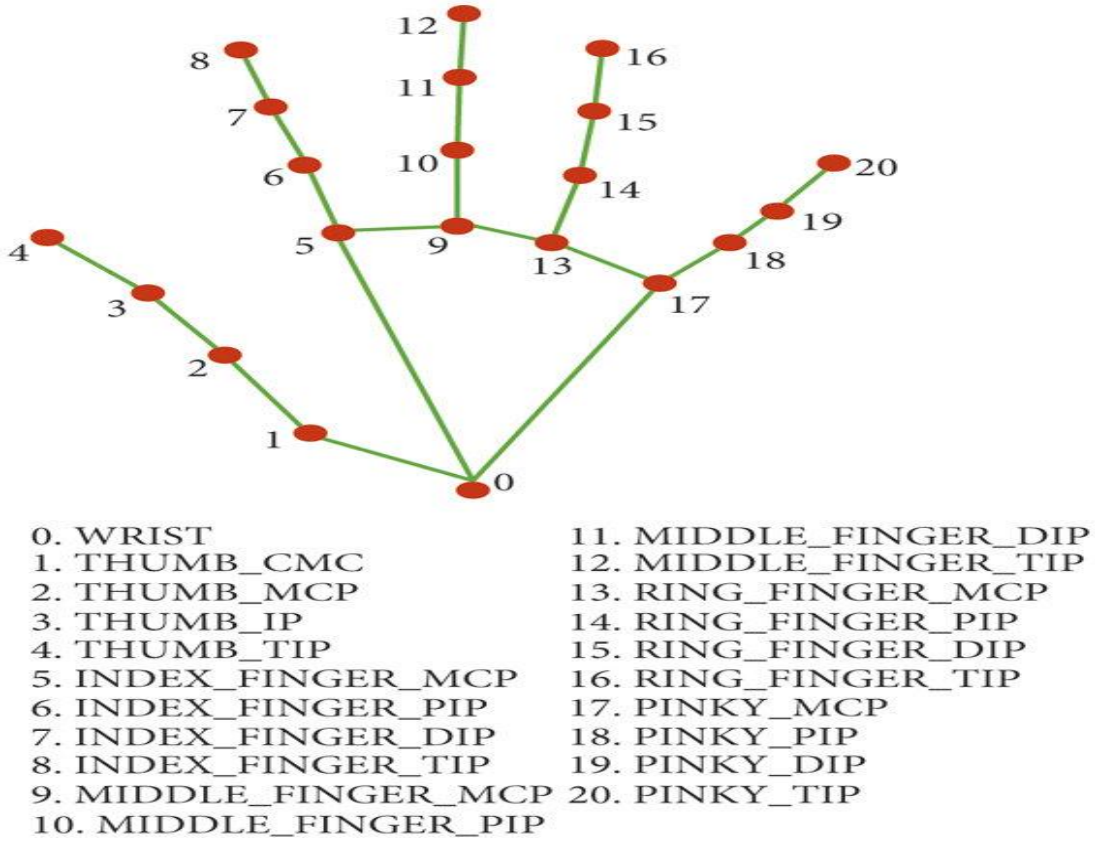


Fig 4.1 Hand Detection

4.2 HAND LANDMARK ESTIMATION

Hand landmark estimation is a crucial step in the AI virtual mouse system using hand gestures. It involves estimating the positions of essential landmarks or keypoints on the hand, such as fingertips, palm, and the base of the hand. Accurate landmark estimation enables the system to capture the hand's configuration and movements, facilitating gesture recognition and precise control of the virtual mouse.

Optical Flow: Optical flow algorithms can track the movement of hand keypoints across consecutive frames, estimating their positions in each frame

based on the flow of pixel intensities. This technique helps maintain the continuity of hand landmark estimation.

Use the chosen hand landmark estimation technique to predict or estimate the positions of hand landmarks. When utilizing optical flow, track the movement of keypoints across frames, estimating their positions based on the flow vectors.

It uses techniques like smoothing or filtering to refine the estimated positions of hand landmarks, reducing noise and improving accuracy.

Hand landmark estimation is a critical component that bridges hand detection and gesture recognition in the AI virtual mouse system. It enables precise tracking of hand movements and the extraction of meaningful features for gesture recognition, ultimately facilitating intuitive and accurate control of the virtual mouse

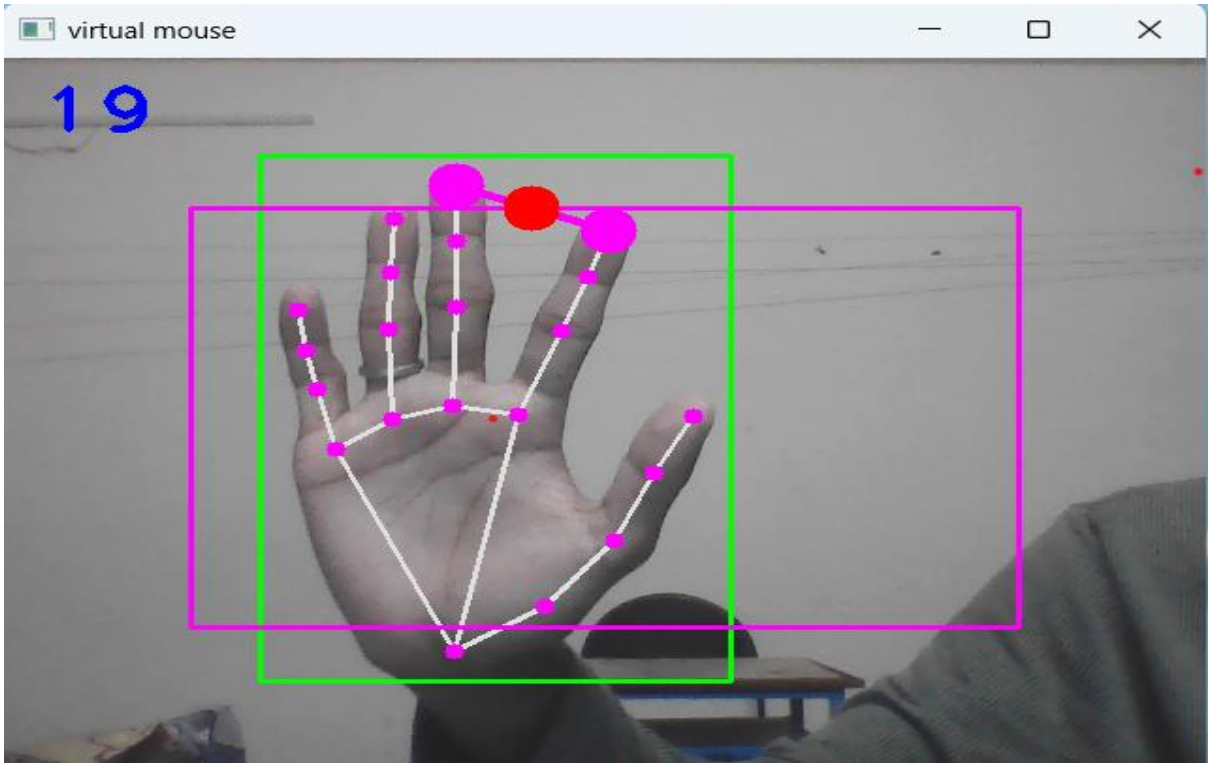


Fig 4.2 Hand Landmark

4.3 GESTURE RECOGNITION

Gesture recognition is a key component of the AI virtual mouse system using hand gestures. It involves identifying and classifying specific hand gestures performed by the user, allowing the system to map them to corresponding mouse actions

During runtime, feed the hand landmarks obtained from the hand landmark estimation into the trained gesture classification model.The model will predict the gesture performed based on the input features, enabling real-time recognition of hand gestures.Define a mapping between the recognized gestures and corresponding mouse actions, allowing the system to control the virtual mouse accordingly.

4.3.1 Gesture Refinement and Optimization:

Continuously evaluate and refine the gesture recognition model based on user feedback and real-world usage scenarios.Consider techniques such as data augmentation, model fine-tuning, or ensemble methods to improve the model's performance and robustness.Optimize the model's inference speed to ensure efficient real-time gesture recognition.

Gesture recognition is a critical component that bridges hand landmarks and mouse control in the AI virtual mouse system. By accurately classifying hand gestures, the system can interpret user intentions and map them to corresponding mouse actions, offering a seamless and intuitive control experience

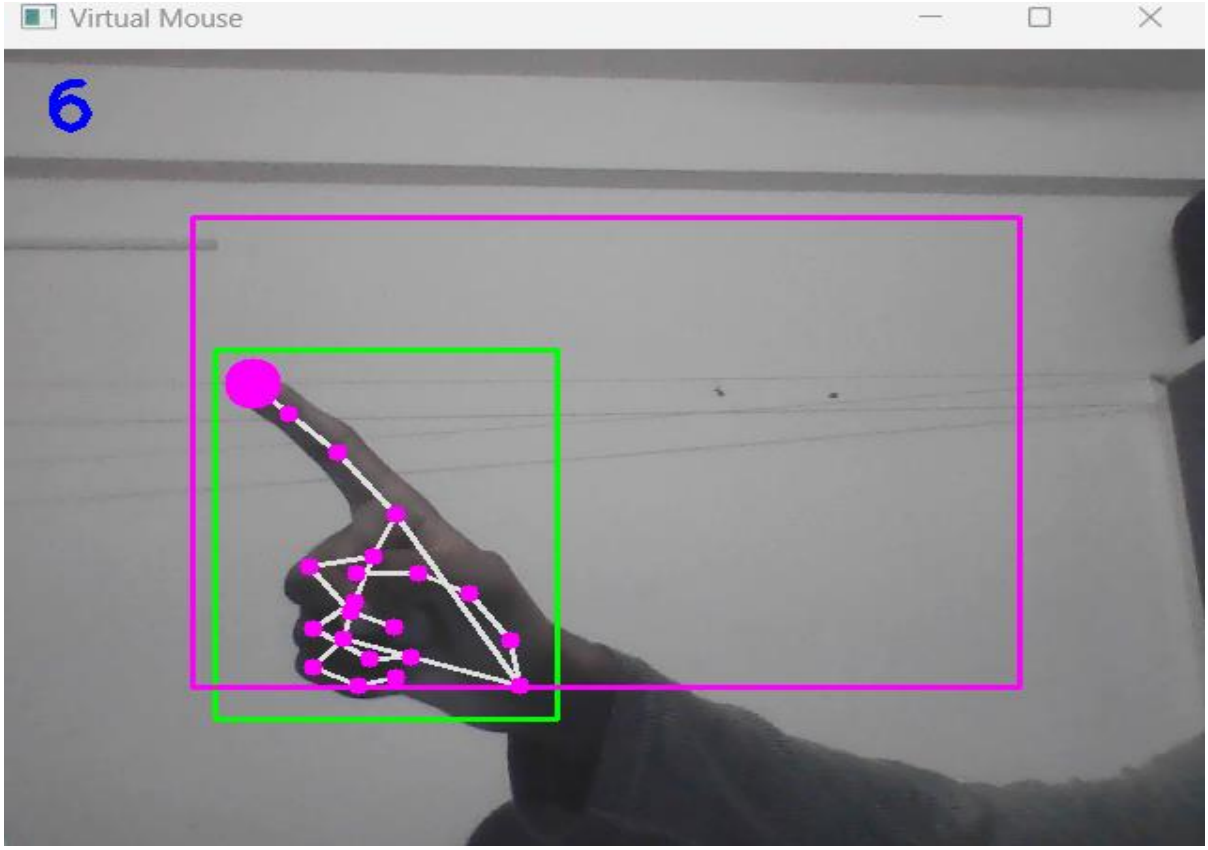


Fig 4.3 Gesture Recognition

4.4 GESTURE MAPPING

Gesture mapping is the process of associating recognized hand gestures with specific mouse actions in the AI virtual mouse system. Once a gesture is recognized through gesture recognition, it needs to be mapped to the appropriate mouse action to control the virtual mouse effectively.

Define a set of gestures that correspond to specific mouse actions, such as cursor movement, left-click, right-click, scrolling, etc. The mapping can be based on user-defined gestures or predefined gestures established by the system.

Establish a clear and intuitive mapping scheme that ensures consistency and ease of use. For example, a swiping motion can be mapped to cursor movement, a pinch gesture to zooming or scrolling, and a closed fist to a mouse click.Map the recognized gestures to their corresponding mouse actions based on the defined mapping scheme.

Define a set of rules or lookup tables that map each recognized gesture to the desired mouse action. For example, if the recognized gesture corresponds to cursor movement, calculate the displacement of the cursor based on the direction and magnitude of the hand movement.

Consider additional parameters such as hand velocity or duration of the gesture to adjust the mouse action accordingly, providing a more fine-grained control experience.

4.5 USER INTERFACE

When implementing an AI virtual mouse system using hand gestures, user interfaces play a crucial role in providing a visual representation of the virtual mouse and facilitating user interaction**.**

4.5.1 Cursor Visualization:

Display a virtual cursor on the computer screen that moves in response to the user's hand gestures. The cursor should accurately reflect the position of the user's hand to provide real-time feedback.

Use visual cues such as a distinct cursor shape or color to differentiate it from the regular system cursor, making it clear that the virtual mouse is being controlled through hand gestures.

Provide visual feedback to indicate the recognized gestures and corresponding mouse actions. This could include displaying icons, animations, or

text overlays representing the recognized gestures and the associated mouse actions.

4.5.2 Control Configuration:

Allow users to customize or configure the mapping between gestures and mouse actions based on their preferences or specific requirements. Provide an intuitive and accessible interface for users to define their own gesture-to-action mappings or select from pre-defined configurations.

Ensure that the user interface provides smooth, responsive, and real-time feedback to user gestures. The virtual cursor movement should closely follow the user's hand movements without noticeable lag.

4.6 INTEGRATION AND INTERACTION

Integration and interaction are crucial aspects of an AI virtual mouse system using hand gestures. It involves seamlessly integrating the system into the user's computing environment and enabling intuitive interaction between the user and the virtual mouse.

4.6.1 System & Hardware Integration:

Ensure that the AI virtual mouse system can be easily integrated with the user's existing computing setup, including desktop computers, laptops, or other devices. Provide compatibility with common operating systems (e.g., Windows, macOS, Linux) and support for popular software applications to ensure broad usability.

Enable the AI virtual mouse system to work with different types of cameras or sensors capable of capturing hand gestures. This may include webcams, depth cameras (e.g., Microsoft Kinect), or specialized motion sensing devices.Ensure the system can interface with the hardware effectively, retrieving input from the camera or sensor and processing it in real-time.

4.6.2 User Calibration and Setup:

Provide a calibration process that allows users to set up the system according to their individual hand size, lighting conditions, and other factors that may affect gesture recognition accuracy.

Guide users through the setup process with clear instructions, ensuring they can optimize the system's performance based on their specific environment.

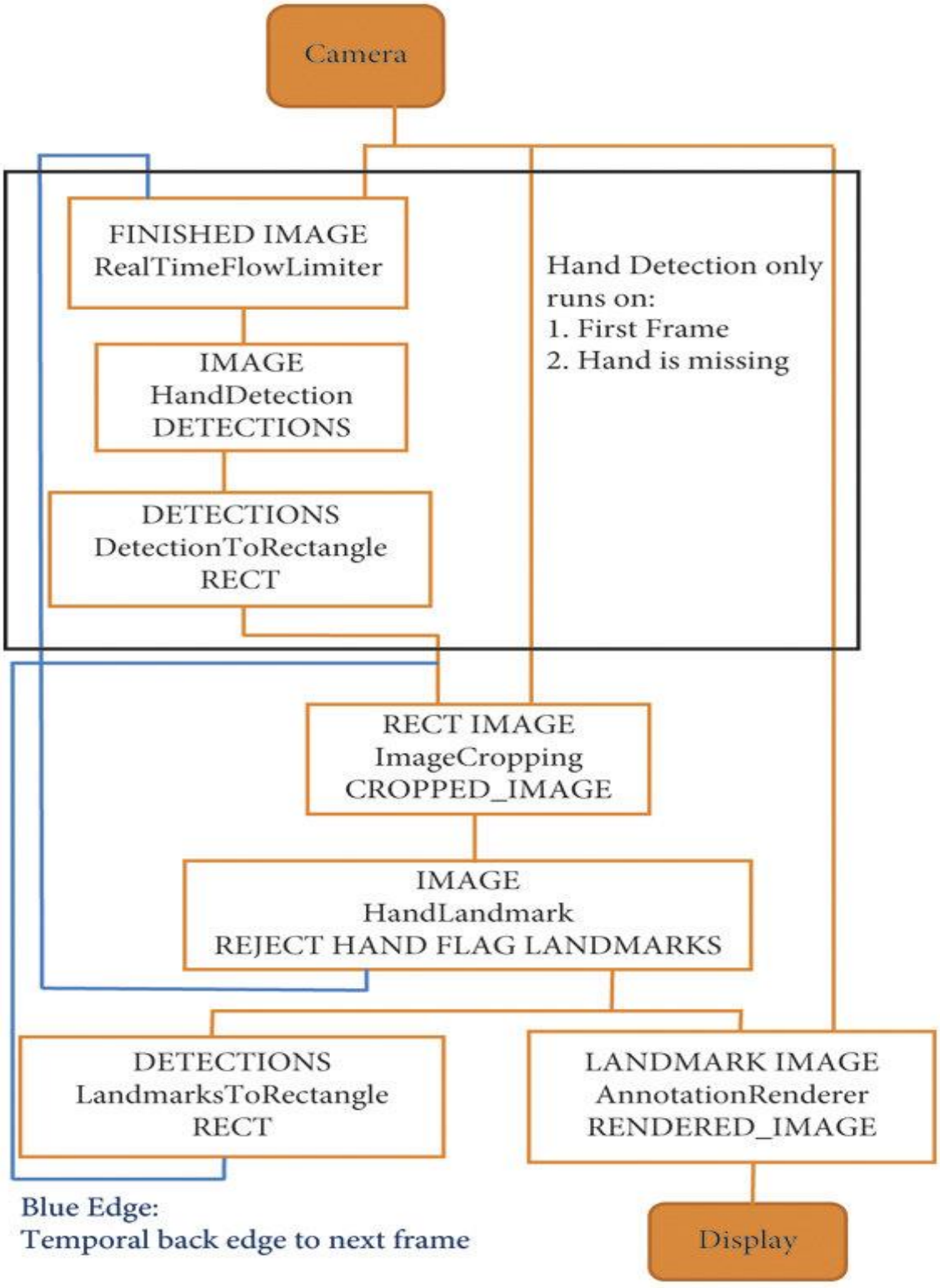


Fig 4.4 Working Model

PHASE 1: USING THE INDEX FINGER FOR CURSOR MOVEMENT

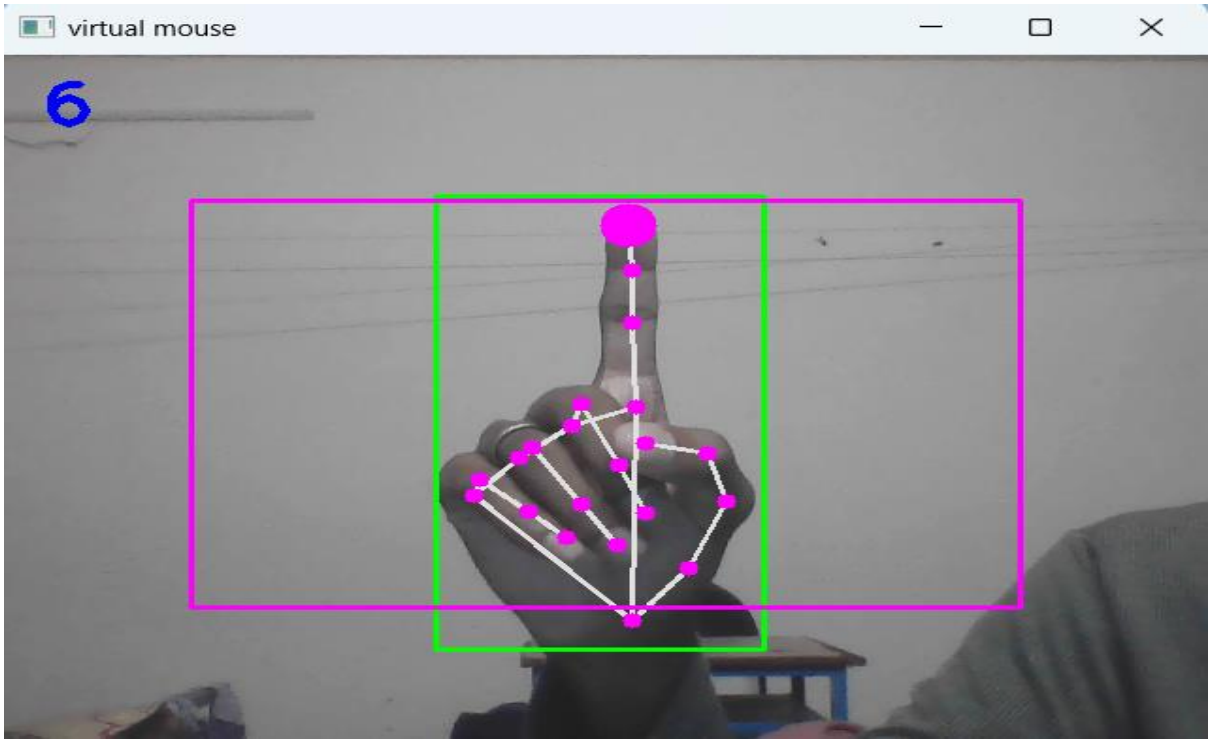


Fig 4.5 Cursor Movement

PHASE 2: JOINING THE INDEX AND MIDDLE FINGER FOR CLICK OPERATIONS

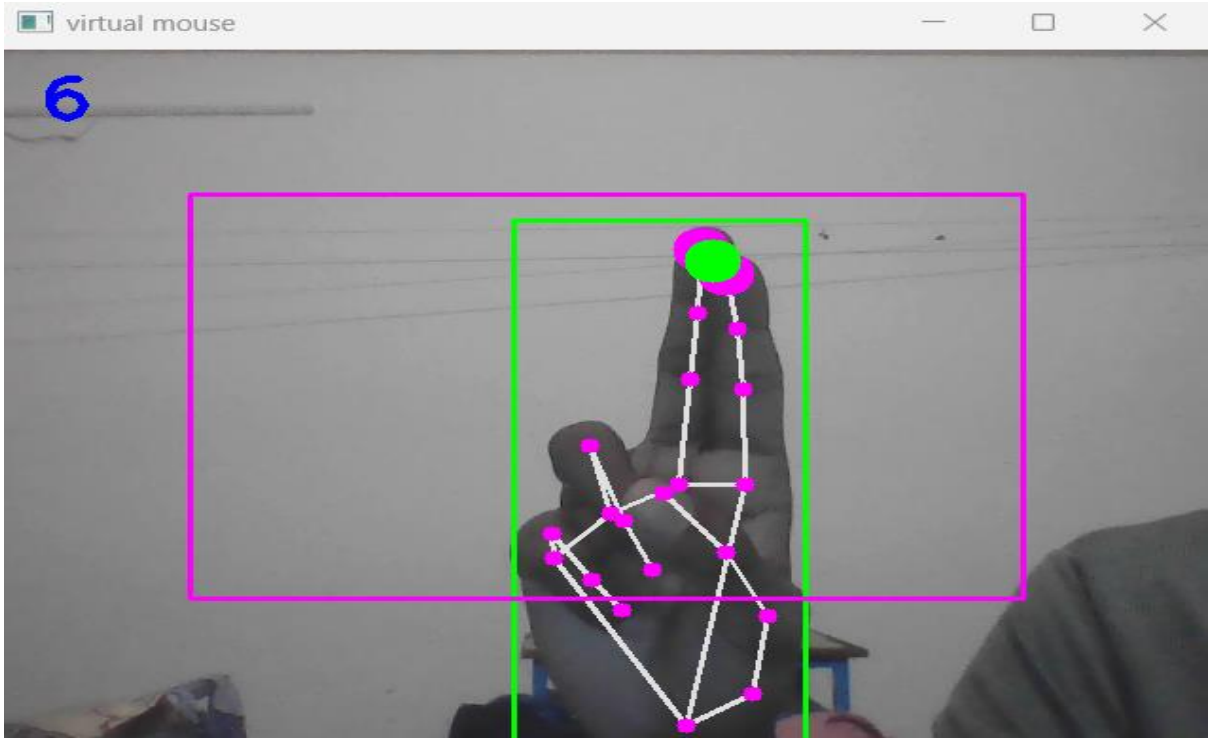


Fig 4.6 Click Operations

5. FUTURE ENHANCEMENTS

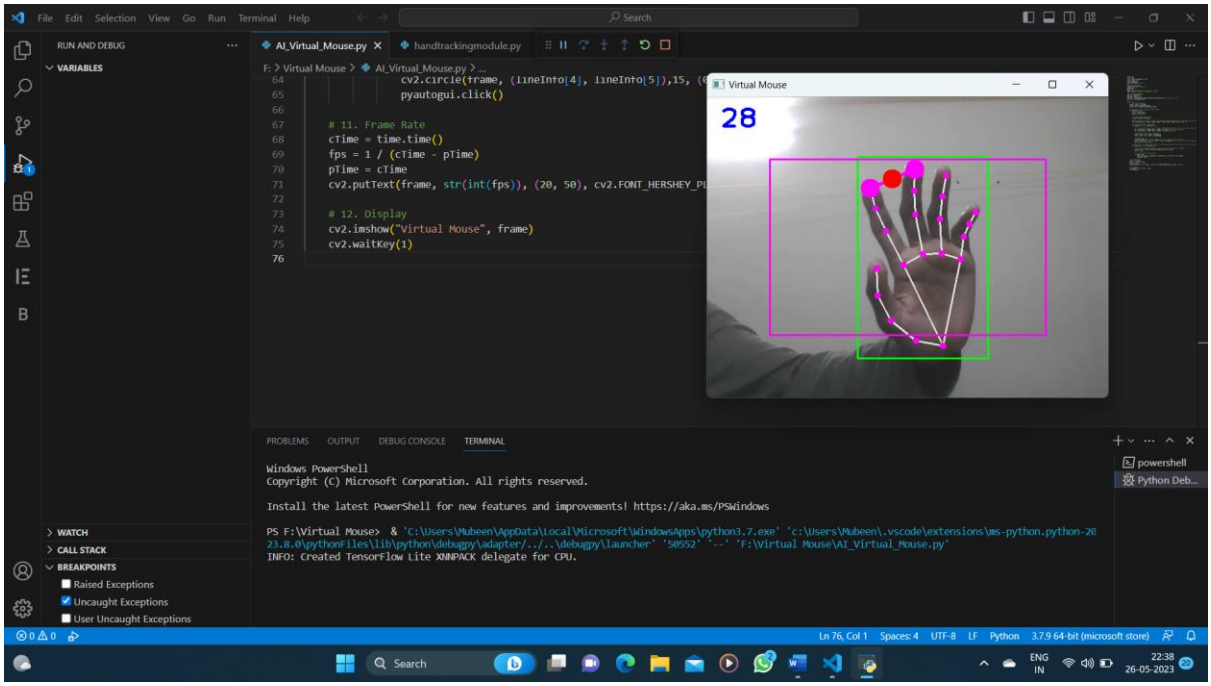
The future enhancements for an AI virtual mouse using hand gestures can involve advancements in technology, improved user experience, and expanded functionality Develop more advanced algorithms and machine learning models to improve the accuracy and robustness of gesture recognition. This can involve incorporating deep learning techniques, leveraging larger and diverse training datasets, and refining the feature extraction process. Explore real-time adaptation and personalization of gesture recognition models to adapt to individual user preferences and varying hand shapes and sizes. Expand the system to support a wider range of hand gestures and gestures combinations. This can enable users to perform complex gestures or sequences of gestures to trigger specific mouse actions or system commands, providing a more flexible and powerful interaction experience. These future enhancements aim to further improve the accuracy, flexibility, and usability of the AI virtual mouse system, providing users with a more seamless and intuitive way to interact with computers and digital devices using hand gestures.

6. CONCLUSION

In conclusion, an AI virtual mouse using hand gestures offers a novel and intuitive way for users to interact with computers and digital devices. By leveraging computer vision, machine learning, and gesture recognition technologies, this system enables users to control a virtual mouse using hand movements, providing a seamless and natural interaction experience.Through the detection of hand gestures, the system accurately identifies and tracks the user's hand movements in real-time. It maps these gestures to specific mouse actions, such as cursor movement, clicking, scrolling, and more. The integration of visual feedback, user interfaces, and calibration mechanisms enhances the user experience, allowing for precise and responsive control over the virtual mouse.Overall, an AI virtual mouse using hand gestures represents a significant step towards more natural and intuitive human-computer interaction, paving the way for innovative and immersive computing experiences in the future.

APPENDIX 1

OUTPUT:



APPENDIX 2

SAMPLE CODE:

AI_VIRTUAL_MOUSE.PY

```
import cv2
import time
import handtrackingmodule as htm
import numpy as np
import pyautogui
pyautogui.FAILSAFE = False

wCam, hCam = 640, 480
frameR = 100 #frame reduction
smoothening = 8
pTime = 0
plocX, plocY = 0, 0#previous locations of x and y
clocX, clocY = 0, 0#current locations of x and y

cap = cv2.VideoCapture(0)
cap.set(3, wCam)#width
cap.set(4, hCam)#height
detector = htm.handDetector(detectionCon=0.60,maxHands=1)#only one hand at a time
wScr, hScr = pyautogui.size()

while True:
    # 1. Find hand Landmarks
    success, frame = cap.read()
    frame = detector.findHands(frame)
    lmList, bbox = detector.findPosition(frame)

    # 2. Get the tip of the index and middle fingers
    if len(lmList) != 0:
        x1,y1 = lmList[8][1:]
        x2,y2 = lmList[12][1:]
        #print(x1, y1, x2, y2)

        # 3. Check which fingers are up
        fingers = detector.fingersUp()

        #in moving mouse it was easy to move mouse upwards but in downward direction it is
tough so we are setting region
        cv2.rectangle(frame, (frameR, frameR), (wCam - frameR, hCam - frameR),(255, 0,
255), 2)

        # 4. Only Index Finger : Moving Mode
        if fingers[1] == 1 and fingers[2] == 0:
```

```
        # 5. Convert Coordinates as our cv window is 640*480 but my screen is full HD so
have to convert it accordingly
        x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))#converting x coordinates
        y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))#converting y

        # 6. Smoothen Values avoid fluctuations
        clocX = plocX + (x3 - plocX) / smoothening
        clocY = plocY + (y3 - plocY) / smoothening

        # 7. Move Mouse
        pyautogui.moveTo(wScr - clocX, clocY)#wscr-clocx for avoiding mirror inversion
        cv2.circle(frame, (x1, y1), 15, (255, 0, 255), cv2.FILLED)#circle shows that we are in
moving mode
        plocX, plocY = clocX, clocY

    # 8. Both Index and middle fingers are up : Clicking Mode but only if both fingers are
near to each other
    if fingers[1] == 1 and fingers[2] == 1:

        # 9. Find distance between fingers so that we can make sure fingers are together
        length, frame, lineInfo = detector.findDistance(8, 12, frame)
        #print(length)

        # 10. Click mouse if distance short
        if length < 35:
            cv2.circle(frame, (lineInfo[4], lineInfo[5]),15, (0, 255, 0), cv2.FILLED)
            pyautogui.click()

    # 11. Frame Rate
    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
    cv2.putText(frame, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,(255, 0, 0), 3)

    # 12. Display
    cv2.imshow("Virtual Mouse", frame)
    cv2.waitKey(1)
```

HANDTRACKINGMODULE.PY

```
import cv2
import mediapipe as mp
import time
import math
import numpy as np

class handDetector():
    def __init__(self, mode=False, maxHands=1, modelComplexity=1, detectionCon=0.5,
trackCon=0.5):
        self.mode = mode
```

```
        self.maxHands = maxHands
        self.modelComplex = modelComplexity
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.modelComplex,
                                        self.detectionCon, self.trackCon)#)# for Hands for a particular
instance = self.jls_extract_def()
        self.mpDraw=mp.solutions.drawing_utils#object for Drawing
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self,img,draw=True):
        imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)#converting to RGB bcoz hand
recognition works only on RGB image
        self.results=self.hands.process(imgRGB)#processing the RGB image
        if self.results.multi_hand_landmarks:# gives x,y,z of every landmark or if no hand than
NONE
            for handLms in self.results.multi_hand_landmarks:#each hand landmarks in results
                if draw:

self.mpDraw.draw_landmarks(img,handLms,self.mpHands.HAND_CONNECTIONS)#joinin
g points on our hand

        return img

    def findPosition(self,img,handNo=0,draw=True):
        xList=[]
        yList=[]
        bbox=[]
        self.lmlist=[]
        if self.results.multi_hand_landmarks:# gives x,y,z of every landmark
            myHand=self.results.multi_hand_landmarks[handNo]#Gives result for particular
hand
            for id,lm in enumerate(myHand.landmark):#gives id and lm(x,y,z)
                h,w,c=img.shape#getting h,w for converting decimals x,y into pixels
                cx,cy=int(lm.x*w),int(lm.y*h)# pixels coordinates for landmarks
                # print(id, cx, cy)
                xList.append(cx)
                yList.append(cy)
                self.lmlist.append([id,cx,cy])
                if draw:
                    cv2.circle(img,(cx,cy),5,(255,0,255),cv2.FILLED)
            xmin,xmax=min(xList),max(xList)
            ymin,ymax=min(yList),max(yList)
            bbox=xmin,ymin,xmax,ymax

            if draw:
                cv2.rectangle(img,(bbox[0]-20,bbox[1]-20),(bbox[2]+20,bbox[3]+20),(0,255,0),2)
```

```
        return self.lmlist,bbox

    def fingersUp(self):#checking which finger is open
        fingers = []#storing final result
        # Thumb < sign only when  we use flip function to avoid mirror inversion else > sign
        if self.lmlist[self.tipIds[0]][1] > self.lmlist[self.tipIds[0] - 1][1]:#checking x position of 4
is in right to x position of 3
            fingers.append(1)
        else:
            fingers.append(0)

        # Fingers
        for id in range(1, 5):#checking tip point is below tippoint-2 (only in Y direction)
            if self.lmlist[self.tipIds[id]][2] < self.lmlist[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)

            # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15,t=3):# finding distance between two
points p1 & p2
        x1, y1 = self.lmlist[p1][1],self.lmlist[p1][2]#getting x,y of p1
        x2, y2 = self.lmlist[p2][1],self.lmlist[p2][2]#getting x,y of p2
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2#getting centre point

        if draw: #drawing line and circles on the points
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)

        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():

    PTime=0# previous time
    CTime=0# current time
    cap=cv2.VideoCapture(0)
    detector=handDetector()

    while True:
        success,img=cap.read()#T or F,frame
        img =detector.findHands(img)
        lmlist,bbox= detector.findPosition(img)
```

```
        if len(lmlist)!=0:
            print(lmlist[4])

        CTime=time.time()#current time
        fps=1/(CTime-PTime)#FPS
        PTime=CTime#previous time is replaced by current time


cv2.putText(img,str(int(fps)),(10,70),cv2.FONT_HERSHEY_COMPLEX,3,(255,0,255),3)#
showing Fps on screen


        cv2.imshow("Image",img)#showing img not imgRGB
        cv2.waitKey(1)


if __name__=="__main__":
    main()
```

REFERENCES

1) Banerjee, A., Ghosh, A., Bharadwaj, K., & Saikia, H. (2014). Mouse control using a web camera based on colour detection. arXiv preprint arXiv:1403.4722.
2) Chu-Feng, L. (2008). Portable Vision-Based HCI. [online] Available at: http://www.csie.ntu.edu.tw/~p93007/projects/vision/vision_hci_p93922 007.pdf [Accessed 25 Aug. 2015].
3) Park, H. (2008). A method for controlling mouse movement using a real-time camera. Brown University, Providence, RI, USA, Department of computer science.
4) Kumar N, M. (2011). Manual Testing: Agile software development. [online] Manojforqa.blogspot.com. Available at: http://manojforqa.blogspot.com/2011/09/agile-softwaredevelopment.html [Accessed 27 Aug. 2015].
5) Niyazi, K. (2012). Mouse Simulation Using Two Coloured Tapes. 36 IJIST, 2(2), pp.57- 63.
6) Sekeroglu, K. (2010). Virtual Mouse Using a Webcam. [online] Available at:http://www.ece.lsu.edu/ipl/SampleStudentProjects/ProjectKazim/Virt ual%20Mouse% 20Using%20a%20Webcam_Kazim_Sekeroglu.pdf [Accessed 29 Aug. 2015].
7) Tutorialspoint.com, (n.d.). SDLC - Agile Model. [online] Available at: http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm [Accessed 27 Aug. 2015].
8) Tabernae.com, (n.d.). Software Life Cycle | Web Development Outsourcing| IT Offshore Outsourcing. [online] Available at: http://www.tabernae.com/process.aspx [Accessed 28 Aug. 2015]