**KARNATAK LAW SOCIETY’S**

**GOGTE INSTITUTE OF TECHNOLOGY**

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

**DEPARTMENT OF**

**COMPUTER SCIENCE ENGINEERING**



A Course Activity Report for

#### Python Programming

Course code :**18CS661**

COURSE ACTIVITY TOPIC:

### CURSOR MOVEMENT ON OBJECT MOTION

**SUBMITTED BY:**

|  |  |  |
| --- | --- | --- |
| **SL NO.** | **NAME** | **USN** |
| 1 | CHIDAMBAR KULKARNI | 2GI20EC037 |
| 2 | LAXMAN KALYANI | 2GI20EC060 |
| 3 | VIDYADHEESH N | 2GI20EC161 |
| 4 | KIRAN BURLI | 2GI20EE013 |

## Guided by

#### Prof. Prasad Pujer

**KARNATAK LAW SOCIETY’S**

**GOGTE INSTITUTE OF TECHNOLOGY**

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

**DEPARTMENT OF**

**COMPUTER SCIENCE ENGINEERING**



# CERTIFICATE

Certified that the course activity entitled **“ CURSOR MOVEMENT ON OBJECT MOTION”** In partial fulfilment for the award of Bachelor of Engineering in KLS GOGTE

INSTITUTE OF TECHNOLOGY of the Visvesvaraya Technological University, Belagavi during the year 2022-2023.It is certified that all corrections/suggestions indicated have been

incorporated in the report. The activity report has been approved as it satisfies the academic requirements in respect of course activity prescribed for the said Degree

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **USN** | **MARKS** | **SIGNATURE** |
| CHIDAMBAR K | 2GI20EC037 |  |  |
| LAXMAN K | 2GI20EC060 |  |  |
| VIDYADHEESH N | 2GI20EC161 |  |  |
| KIRAN BURLI | 2GI20EE013 |  |  |

SIGNATURE OF THE STAFF: DATE:

### CONTENTS:

1. ABSTRACT
2. INTRODUCTION
3. METHODOLOGY
4. PROJECT SPECIFICATION
5. SOURCE CODE
6. EXPLANATION
7. OUTPUT
8. CONCLUSION
9. REFERENCE

### ABSTRACT

This project is a mouse simulation system which performs all the functions performed by your mouse corresponding to your hand movements and gestures. Simply speaking, a camera captures your video and depending on your hand gestures, you can move the cursor and perform left click, right click, drag, select and scroll up and down. The predefined gestures make use of only three fingers marked by different colors.

You are watching a movie with your friends on a laptop and one of the guys gets a call. You must get off your place to pause the movie. You are giving a presentation on a projector and need to switch between applications. You must move across the whole stage to the podium to use your mouse. How better would it be if you could control your mouse from wherever you were? Well, we have a solution!

The project is essentially a program which applies image processing, retrieves necessary data and implements it to the mouse interface of the computer according to predefined notions. The code is written on Python. It uses of the cross-platform image processing module OpenCV and implements the mouse actions using Python specific library PyAutoGUI. Thus, in addition to a webcam (which almost all laptops are already loaded with) a computer needs to be pre- equipped with the following packages for the project to run such as:

* 1. Python 2.7 interpreter
  2. OpenCV
  3. Numpy
  4. PyAutoGUI

Video captures by the webcam is processed and only the three colored finger tips are extracted. Their centers are calculated using method of moments and depending upon their relative positions it is decided that what action is to be performed.

## Introduction

The project “Mouse control using Hand Gestures” is developed aiming to better the process of human-computer interaction. It aims to provide the user a better understanding of the system and to let them use alternate ways of interacting with the computer for a task.

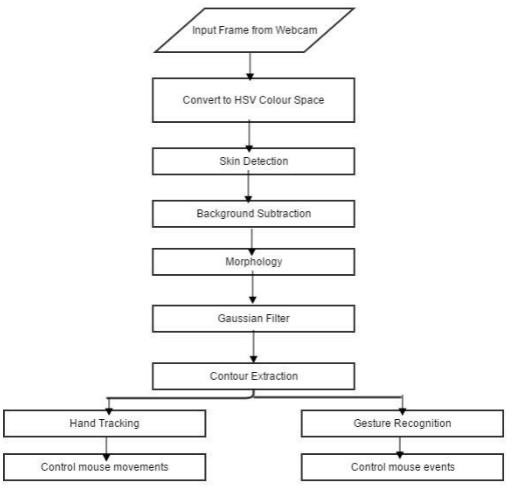
The task here is to control the mouse even from a distance just by using hand gestures. It uses a program in python and various libraries such as PyAutoGUI, Numpy and image processing module OpenCV to read a video feed which identifies the users’ fingers represented by three different colors and track their movements. It retrieves necessary data and implements it to the mouse interface of the computer according to predefined notions.

The project can be useful for various professional and non-professional presentations. It can also be used at home by users for recreational purposes like while watching movies or playing games.

## Methodology

#### Framework Architecture

The algorithm for the entire system is shown in Figure below.



In order to reduce the effects of illumination, the image can be converted to chrominance colour space which is less sensitive to illumination changes. The HSV colour space was chosen since it was found by to be the best colour space for skin detection. The next step would be to use a method that would differentiate selected colour pixels from non-colour pixels in the image (colour detection). Background subtraction was then performed to remove the face and other skin colour objects in the background. Morphology Opening operation (erosion followed by dilation) was then applied to efficiently remove noise. A Gaussian filter was applied to smooth the image and give better edge detection. Edge detection was then performed to get the hand contour in the frame. Using the hand contour, the tip of the index finger was found and used for hand tracking and controlling the mouse movements. The contour of the hand was also used for gesture recognition. The system can be broken down in four main components, which are:

* + 1. Colour detection
    2. Colour Contour Extraction
    3. Hand Tracking
    4. Gesture Recognition
    5. Cursor Control

# PROJECT SPECIFICATIONS

Software Specifications:

* + - 1. 64-bit Operating System: Windows 8 or Higher

2.OpenCV 2.4.9 needs to be installed prior to running.

3.Windows Administrator permissions are needed for some parts of the program to function properly.

Hardware Specifications:

1. A Webcam

Environment Specifications:

1. A clear white background There should be no other objects (especially red, blue,y)

## Source Code

import cv2

import numpy as np import pyautogui

# Set the region of interest (ROI) for hand detection top, right, bottom, left = 10, 350, 225, 590

# Set the resolution for cursor movement screen\_width, screen\_height = pyautogui.size() mov\_resolution = (screen\_width, screen\_height)

# Initialize the previous centroid position prev\_centroid = None

# Set the sensitivity factor for cursor movement sensitivity = 2.5

# Start the video capture cap = cv2.VideoCapture(0)

while True:

# Read the video frame ret, frame = cap.read()

# Flip the frame horizontally frame = cv2.flip(frame, 1)

# Extract the region of interest (ROI) for hand detection roi = frame[top:bottom, right:left]

# Convert the ROI to grayscale

gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

# Apply Gaussian blur to reduce noise gray = cv2.GaussianBlur(gray, (7, 7), 0)

# Perform thresholding to segment the hand region

\_, thresh = cv2.threshold(gray, 100, 255, cv2.THRESH\_BINARY)

# Find contours in the thresholded image

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Find the contour with the maximum area (hand) if len(contours) > 0:

hand\_contour = max(contours, key=cv2.contourArea)

# Calculate the centroid of the hand contour M = cv2.moments(hand\_contour)

if M["m00"] != 0:

cx = int(M["m10"] / M["m00"])

cy = int(M["m01"] / M["m00"]) centroid = (cx, cy)

# Move the cursor based on the centroid motion if prev\_centroid is not None:

dx = int((cx - prev\_centroid[0]) \* sensitivity) dy = int((cy - prev\_centroid[1]) \* sensitivity) pyautogui.moveRel(dx, dy)

prev\_centroid = centroid

# Draw the hand contour and centroid on the frame cv2.drawContours(roi, [hand\_contour], 0, (0, 255, 0), 2)

cv2.circle(roi, centroid, 5, (0, 0, 255), -1)

# Display the frame cv2.imshow("Hand Motion", frame)

# Check for keypress to exit

if cv2.waitKey(1) == ord("q"): break

# Release the video capture and close all windows cap.release()

cv2.destroyAllWindows()

## Explanation

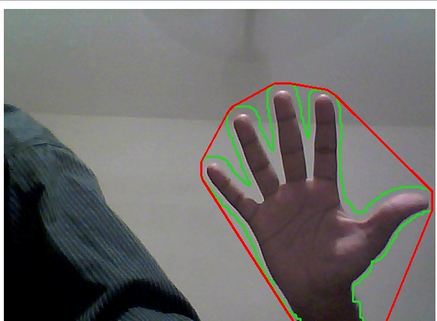
1. Import the necessary libraries: `cv2` for computer vision operations, `numpy` for numerical computations, and `pyautogui` for controlling the cursor.
2. Set the region of interest (ROI) coordinates to define the area where hand detection will be performed. Adjust these values according to the specific setup and camera placement.
3. Get the screen resolution using `pyautogui.size()` to determine the range of cursor movement.
4. Initialize the `prev\_centroid` variable to store the previous centroid position of the hand. This will be used to calculate the motion vector for cursor movement.
5. Set the sensitivity factor (`sensitivity`) for cursor movement. This factor determines the speed at which the cursor moves based on the hand motion.
6. Start the video capture by creating a `VideoCapture` object.
7. Enter the main loop to continuously read frames from the video capture.
8. Flip the frame horizontally using `cv2.flip()` to match the movements of the hand.
9. Extract the ROI from the frame using the defined coordinates.
10. Convert the ROI to grayscale for hand detection.
11. Apply Gaussian blur to reduce noise in the grayscale image.
12. Perform thresholding to segment the hand region from the background.
13. Find contours in the thresholded image using `cv2.findContours()`.
14. Find the contour with the maximum area, which represents the hand.
15. Calculate the centroid of the hand contour using `cv2.moments()`.
16. If the centroid is valid (non-zero), calculate the motion vector based on the difference between the current and previous centroid positions, multiplied by the sensitivity factor. Use `pyautogui.moveRel()` to move the cursor relative to its current position.
17. Update the previous centroid position.
18. Draw the hand contour and centroid on the ROI for visualization.
19. Display the frame in a window named "Hand Motion" using `cv2.imshow()`.

20.Check for a keypress in every iteration. If the 'q' key is pressed, break the loop.

21.Release the video capture and close all windows.

This code continuously captures frames from the webcam, detects the hand within the defined ROI, calculates the centroid, and moves the cursor accordingly based on the hand's motion. The sensitivity factor determines the speed of cursor movement. The code also draws the hand contour and centroid on the frame for visualization purposes.

**OUTPUT**

****

# CONCLUSION

The vision based cursor control using hand gesture system was developed in Python language, using the OpenCV library. The system could control the movement of a Cursor by tracking the users’ hand. Cursor functions were performed by using different hand gestures. The system has the potential of being a viable replacement for the computer mouse, however due to the constraints encountered; it cannot completely replace the computer mouse. The major constraint of the system is that it must be operated in a well-lit room. This is the main reason why the system cannot completely replace the computer mouse, since it is very common for computers to be used in outdoor environments with poor lighting condition. The accuracy of the hand gesture recognition could have been improved, if the Template Matching hand gesture recognition method was used with a machine learning classifier. This would have taken a lot longer to implement, but the accuracy of the gesture recognition could have been improved. It was very difficult to control the cursor for precise cursor movements, since the cursor was very unstable. The stability of the cursor control could have been improved if a Kalman filter was incorporated in the design. The Kalman filter also requires a considerable amount of time to implement and due to time constraints, it was not implemented. All the operations which were intended to be performed using various gestures were completed with satisfactory results

**REFERENCE**

|  |
| --- |
| 1. Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj,” Mouse  Control using a Web Camera based on Color Detection”,IJCTT,vol.9, Mar 2014. |
| 2. Angel, Neethu.P.S,”Real Time Static & Dynamic Hand Gesture Recognition”, International Journal of Scientific & Engineering  Research Volume 4, Issue3, March-2013. |