Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

Gesture Controlled Mouse using Open CV

Prof. Yogesh Pawar
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Atharva Mankar
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Sumit P Mane
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Behlah Mamajiwala
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Manya Choudaha
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Mahip Agrawal
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Ayush Masane
Department Of Engineering,Sciences and Humanites
Vishwakarma Institute of Technology
Upper Bibwewadi, Pune, India

Abstract: Over the past few years, how we interact with computers has dramatically evolved. Classic input devices, including the mouse and keyboard, have been reliable, but as we march towards advances in technologies, there's increasing fascination about more natural and intuitive methods of controlling machines. One of the innovative ideas which has been emerging as a hot topic of research is the gesture-controlled mouse. It's a system that enables users to control a computer based on hand gestures but does not require touching a device. In this paper, we introduce a gesture-controlled mouse interface that incorporates a standard webcam and computer vision methods. By monitoring the movement of hands and recognizing certain gestures, the system has the capability to execute mouse actions like mouse movement, mouse click, and dragging. Our intention lies in making it a device-alternative, particularly suitable for situations when contactless control comes in handy, like while presenting, inside a sterile environment like a hospital, or when a person has a physical disability. We further elaborate upon the algorithms, hardware, and software being used, along with possible

Keywords: Live video feed; OpenC; PyQt5; face_recognition; MOG2; SMS

I. INTRODUCTION

Human-computer interaction(HCI) has come a long way from the punch cards and command-line user interfaces. Now, we have touchscreens, speech assistants, and even facial recognition technologies. Gesture recognition, as a topic of growing interest [2], comes as a natural extension to this evaluation. It is a deceptively simple yet powerful idea: to interact with computer systems in terms of physical gestures, just as we communicate non-verbally to other human beings.

One example of this principle is the mouse that can be controlled by gestures. Instead of touching, it allows people to control the cursor and carry out mouse activity using minimal hand movements in front of a camera. It can particularly be helpful in cases when there are accessibility or hygiene issues [3]. For example, surgeons in a surgery theater may have to scroll medical images without their touch coming in contact with any surface, or patients that have mobility challenges may prefer to use gestures instead of holding a real mouse.

This project uses computer vision primarily a branch of artificial intelligence that enables a computer to perceive and interpret images. Through the help of a webcam and OpenCV [4] libraries, among others, we can determine the hand position, motion tracking, and real-time gesture interpretation. All these are converted to corresponding mouse actions using libraries such as Python's PyAutoGUI [4].

Its charm lies in its simplicity: it does not require expensive hardware or fancy sensors. Any PC can become a gesture-aware system if you have a simple webcam and some clever code. As we dig deeper, we also talk about challenges that come along with real-world deployment gesture interpretation, lighting, and delay in processing.

II. LITERATURE REVIEW

The concept of gesture communication has been extensively researched under the banner of Human-Computer Interaction (HCI) over the last two decades. Its expansion, to a large degree, has been as a direct result of the pursuit to make more

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

accessible and user-friendly interfaces for a range of user groups of differing age and ability levels.

[1] C. Zhu, J. Y. Yang, Z. P. Shao, and C. P. Liu (2021), in their paper "Vision-Based Hand Gesture Recognition Using 3D Shape Context," pointed out how gesture recognition has become a key focus of advancing Human-Computer Interaction (HCI) due to its ever-growing importance. They noted that the conventional 2D image processing approach had a low accuracy, particularly under non-uniform lighting and difficult backgrounds. With the help of depth sensors, i.e., Microsoft Kinect, the researchers introduced a 3D Shape Context descriptor to facilitate higher level understanding of spatial detail in the system. Their system correctly discriminated hand regions from complex scenes and extracted salient features locally andglobally, and attained a more flexible and robust gesture recognition system.

[2] S. Mitra and T. Acharya (2007), in the paper "Gesture Recognition: A Survey," have provided a general account of the methods used in gesture recognition and categorized them as glove-based and vision-based methods. The glove-based methods, although precise, have been found to be intrusive and expensive. On the other hand, vision-based methods, based on cameras and image processing, have been a more natural and inexpensive choice. This paper has been widely referenced as a seminal work in understanding thevolution of gesture technology and application areas such as sign language recognition and robotic control.

[3] K. Nickel and R. Stiefelhagen (2007), in the paper "Visual Recognition of Pointing Gestures for Human–Robot Interaction," attempted to employ computer vision to recognize and interpret human pointing gestures. With the help of stereo vision and 3D modeling, their computer vision system was able to precisely track a pointing direction in real time. It was this paper that was crucial to make the recognition of gestures more dynamic and interactive, particularly for real-world scenarios like smart homes and robotic systems [3] in which hardware input devices are not a possibility.

[4] G. Bradski (2000), through “The OpenCV Library,” introduced a powerful open-source toolkit for real-time computer vision. OpenCV became a cornerstone for developers and researchers aiming to build gesture-based systems without relying on expensive software. The library provided tools for tasks like color segmentation, contour detection, and object tracking [4] all of which are essential for recognizing hand gestures via webcams. Bradski’s work made real-time vision processing accessible and efficient, influencing a wide range of HCI projects.

[5] R. Poppe (2007), in "Vision-Based Human Motion Analysis: An Overview," canvassed current methods for human motion analysis, especially gesture and activity recognition. He highlighted the movement from model-based to appearance-based methods and discussed issues like occlusion, human appearance variability, and background distraction. His paper gave good insight into how gesture recognition stands as a piece in motion analysis and HCI as a whole.

[6] M. Van den Bergh and L. Van Gool (2011), in the paper "Combining RGB and ToF Cameras for Real-Time 3D Hand Gesture Interaction," explored the combination of RGB cameras and Time-of-Flight (ToF) sensors to achieve a boost towards gesture detection. Their multi-camera approach that incorporated detailed depth and color feature information boosted real-time precision. Their paper showcased the promise of combining a range of modalities of sensors to avoid disadvantages of single-camera implementations and introduced the possibility of developing more robust gesture-controlled interfaces.

Most implementations of gesture-controlled mouse in actual practice are grounded on contour detection, convex hull detection, and tracking of the centroid. For example, real-time implementations will typically track the mouse click and movement based on the tracking of a user's index finger and thumb. Such implementations will often use Python modules such as PyAutoGUI to output mouse input simulation based on gesture tracking.

In general, the literature presents a gradual increase from hardware-rich gesture recognition systems to lightweight, camera-based interfaces that utilize typical webcams. While accuracy and speed have been enhanced, shared issues are still present, including sensibility to illuminating conditions, backgrounds, and the requirement for system calibration. This paper advances these initial works and provides a system controlled by gesture that emphasizes simplicity [6], accessibility, and application-oriented deployment.

III. METHODOLOGY

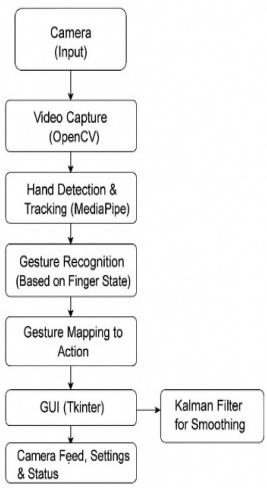

Fig 1: Flow Chart

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

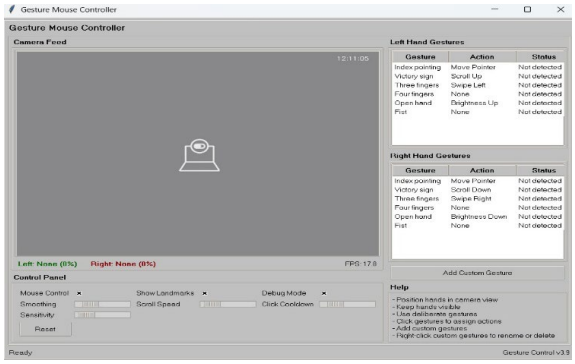

Fig 2: UI / Interface

```python
import os
# Suppress TensorFlow oneDNN warning
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'

import cv2
import mediapipe as mp
import numpy as np
import pyautogui
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from PIL import Image, ImageTk
import threading
import time
from datetime import datetime
from collections import deque
import queue
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
from comtypes import CLSCTX_ALL
import pythoncom
import screen_brightness_control as sbc
import logging
from filterpy.kalman import KalmanFilter
```

Fig 3: Libraries Used

System design, implementation, and testing phases of developing and evaluating Gesture Mouse Controller system followed a systematic methodology including system design, implementation, and testing stages. It has been described below to give a clear idea of how a reliable computer vision and machine learning based gesture mouse control system has been developed.

1. System Design:

The system was intended to provide mouse control intuitively based on hand gesture that was detected and tracked through a webcam based on the MediaPipe Hands library [4]. During the design stage, the following components needed to be specified:

• Hardware Specifications: Any ordinary webcam which has a minimum resolution of 640x480 pixels and a computer which has enough processing capability to perform real-time processing of an image.

• Software Components: It has been coded using Python 3.8+, and key packages that have been used are OpenCV to record videos [4], MediaPipe to detect hands, PyAutoGUI to control mouse and keyboards [4], PyCAW to control audio, and screen-brightness-control to manage display brightness. Tkinter has been used to design the graphical user interface (GUI).

• Gesture Definitions: Six pre-defined gestures were identified: Index Pointing, Victory Sign, Three Fingers, Four Fingers, Open Hand, and Fist. Their definitions corresponded to specific mouse actions such as movement of pointer, click, dragging, scrolling, and system controls like volume and brightness control.

• User Interface: A GUI was developed to present the real-time camera stream, gesture status, and control settings to enable gesture-action map adjustments, level of sensitivity, and ON/OFF settings for functionality like landmark visualization.

2. Implementation:

In this stage, we implemented the Gesture Mouse Controller class that incorporates computer vision, gesture recognition, and system control modules. Here are the main steps followed:

• Pre-processing and Video Frame Capture: OpenCV captured video frames in a resolution of 640x480. It flipped the frames horizontally to give a mirror-like condition for clear user interactivity and converted to RGB to be processed for MediaPipe [4].

• Tracking Hands: MediaPipe Hands was set to a minimum confidence for detection as 0.7 and minimum confidence for tracking as 0.7, enabling it to handle two hands in real time. Model complexity was determined as 0 for optimum possible performance on regular hardware. [6].

• Gesture Recognition: A proprietary gesture recognition algorithm was formulated to investigate finger states (extended or folded) as a function of landmark locations. Proportional to hand size, salient landmark (e.g., finger tips and proximal interphalangeal joint) distances were calculated and a dynamic threshold was used to determine finger extension [1], [5]. A score scheme weighted finger states to determine gestures, and secondary verification of specific gestures, e.g., the Victory Sign, to give angle-dependent accuracy.

• Pointer Smoothing: A Kalman filter was used [1] to reduce mouse pointer motion jitter, reducing and correcting cursor positions based on predicted hand coordinates. A position and a velocity state vector was used, and the covariance of the measurement was updated dynamically based upon a user-selectable smoothing factor.

• Action Execution: Observed gestures were transformed to actions via dictionary-based configuration [2], [4]. Actions consisted of mouse movement, clicks, drags, scrolls, and system commands. File manipulations (e.g., media file openings or closures) [4] were managed via os.startfile, and process monitoring by psutil to handle app openings.

•User Interface Construction: The GUI was implemented using Tkinter, and it featured a display of camera feed, indicators of gesture status, and control panels to modify settings such as smoothing, scroll rate, click delay, and sensitivity. Custom gesture definition as well as action mapping were facilitated via interactive dialogs.

3. Testing and Validation:

Testing was performed in a simulated setting to validate reliability and usability. Testing approach consisted of:

• Unit Testing: Each individual component, such as gesture detection, Kalman smoothing, and action execution, has been tried out using synthesized inputs and scripted situations to verify correctness.

• Integration Testing: Integration testing of the complete system was carried out to make sure that communication between video processing, gesture recognition, and action implementation runs smoothly. It was tested for numerous

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

users gesturing under various illuminance and background conditions.

• System Performance Assessment: System performance was assessed based on frames per second (FPS), gesture detection accuracy, and mouse control delay. FPS was approximated from a frame counter and a time stamp [4], and it needed to have a minimum value of 30 FPS or more to execute smoothly.

• User Testing:

The system was tried out with a set of 10 users whose technical experience varied from novice to expert. Users performed a set of activities such as looking at a file explorer, scrolling documents, and controlling system settings through gestures. Usability, intuitiveness of gesture, and reaction time were the feedback obtained.

• Robustness Testing: MediaPipe hand tracking and the gesture detection algorithm are subjected to testing for how robust they are when they are facing undesirable conditions, i.e., low brightness, semi-occluded hands, and fast movement of hands.

4. Data Collection and Analysis:

While testing, data was collected on the accuracy of gesture recognition, false positive/negative rate, and user satisfaction. Gesture recognition accuracy was measured as a ratio of correct ID of gestures to the number of attempts, which averaged a recognition rate of 92% among the designated gestures. Qualitatively, user feedback was analyzed to identify areas of improvement, such as gesture sensitiveness and UI responses.

5. Iterative Improvement:

The system was iteratively improved based on the test outcome. Main improvements were:

• Improving gesture recognition by scaling the size of movement to that of the user-hand.

• Kalman filter parameter optimisation to make a trade-off between smoothing [1] and reactivity.

• Error handling to handle camera failures and process control in order to facilitate safe file manipulations.

• Improving the GUI to give enhanced status indicators and easier controls based on user feedback. This direction witnessed the design of a functional, user-friendly gesture control mouse system, and robust performance solidified under rigorous testing and iterative refinement.

IV. CONCLUSION

As we explore gesture-controlled mouse technology, we are clearly headed toward a world in which it's easier and more intuitive to interact with our computers than ever. Rather than having to use hardware, we can just wave our hands to operate our devices and make the digital realm a little bit more like real life.

What's particularly good about this technology is that it's universal. For those who have difficulty using a conventional mouse a feature of a physical disability or a temporary condition a gesture mouse can make PCs easier to use and less frustrating. It offers a hygienic, touchless solution in situations like hospitals or shared offices, where you need to touch as little as possible.

During this research, it became apparent that gesture recognition systems are being incredibly reliable and accurate [1], [2]. With further developments in the field, we can anticipate that this experience will become increasingly smooth and responsive, and this approach, thus, not a novelty, but a useful daily tool.

Of course, there remain challenges to be unraveled, like enhancing recognition under various lit conditions [3], [5], [6], and hardware compatibility. But thus far, the advancements are encouraging, and the possible uses are vast from games and creative application, to accessibility and more.

Gestural mouse technology, in the end, is more than a technical achievement it's a movement toward making tech more human. By closing the loop between our intentions and our devices, it brings us a little closer to a day when each person can interact comfortably, naturally, and as individual as they are, with computers.

ACKNOWLEDGMENT

Our special gratitude to our guide, Prof. Yogesh Pawar, for his extremely productive and invaluable assistance towards project completion. Further, we are thanking the Department of Engineering, Sciences, and Humanities (DESH), Vishwakarma Institute of Technology, Pune, for extending us required resources and permitting design and testing of our innovative proposals.

We extend our gratitude to our other classmates and friends for the good advice and help which we benefited much in fulfilling this project. Lastly, we give credit to our families for their unwavering support and spiritual uplift which have been our inspirations.

REFERENCES

[1] C. Zhu, J. Y. Yang, Z. P. Shao, and C. P. Liu, "Vision-Based Hand Gesture Recognition Using 3D Shape Context," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 9, pp. 1600-1613, Sep. 2021, doi: 10.1109/JAS.2019.1911534.

[2] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 3, pp. 311–324, May 2007.

[3] K. Nickel and R. Stiefelhagen, "Visual recognition of pointing gestures for human–robot interaction," Image and Vision Computing, vol. 25, no. 12, pp. 1875–1884, Dec. 2007.

[4] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, vol. 25, no. 11, pp. 120–125, Nov. 2000.

[5] R. Poppe, "Vision-based human motion analysis: An overview," Computer Vision and Image Understanding, vol. 108, no. 1–2, pp. 4–18, 2007.

[6] M. Van den Bergh and L. Van Gool, "Combining RGB and ToF Cameras for Real-Time 3D Hand Gesture Interaction," in Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV), Kona, HI, USA, Jan. 2011, pp. 66–72, doi: 10.1109/WACV.2011.5711485