GESTURE MOUSE: REAL-TIME GESTURE RECOGNITION FOR COMPUTER CONTROL USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Abstract

Gesture Mouse is a real-time hand gesture recognition system that uses deep convolutional neural networks to mimic mouse functionalities for controlling computers. The system requires only a standard webcam, and it can detect and recognize various hand movements and gestures, such as clicking, scrolling, zooming, movements, and hovering. The aim of this project is to improve and ease the human-computer interaction, especially for users who are physically challenged or have limited mobility. The Gesture Mouse system consists of three main components: hand detection, hand landmark extraction, and gesture recognition. The hand detection component is based on the Single Shot Multi Box Detector (SSD) model, which is used to detect the hand region in each frame of the webcam stream. The hand landmark extraction component uses a convolutional neural network (CNN) to predict the 3D coordinates of 21 key hand landmarks, including fingertips, palm canter, and wrist. The gesture recognition component is based on a CNN classifier that is trained on a large dataset of hand gesture images. The system performs well in real-time and can be used to control a computer with a high degree of accuracy and reliability. The proposed system has the potential to provide a more natural and intuitive way of interacting with computers, especially for individuals with motor impairments and also in technologies such as Virtual Reality and Augmented Reality.

Keywords: Real time, Gesture Recognition, Convolutional neural networks, Human computer interaction. Single Shot MultiBox, CNN classifier.

Authors

Revathi K P
Assistant Professor
Department of Artificial Intelligence and Data Science
SRM Easwari Engineering College
Chennai, Tamil Nadu, India.

Kalaiselvi T
Assistant Professor
Department of Artificial Intelligence and Data Science
SRM Easwari Engineering College
Chennai, Tamil Nadu, India.

Naveen Kumar S
Department of Artificial Intelligence and Data Science
SRM Easwari Engineering College
Chennai, Tamil Nadu, India.

Arjun P
Department of Artificial Intelligence and Data Science
SRM Easwari Engineering College
Chennai, Tamil Nadu, India.

I. INTRODUCTION

In recent years, the field of human-computer interaction has witnessed a tremendous growth in innovative technologies aimed at improving the way we interact with computers. One such technology is gesture recognition, which allows users to control computers through hand and body movements. However, most existing gesture recognition systems are either too complex or too expensive, making them inaccessible to many users, especially those with physical disabilities.

To address this challenge, we present Gesture Mouse, a real-time hand gesture recognition system that uses deep convolutional neural networks to mimic mouse functionalities for controlling computers. The proposed system requires only a standard webcam, and it can detect and recognize various hand movements and gestures, such as clicking, scrolling, zooming, movements, and hovering.

Gesture Mouse has the potential to provide a more natural and intuitive way of interacting with computers, especially for individuals with motor impairments. The system consists of three main components: hand detection, hand landmark extraction, and gesture recognition, all of which are based on deep learning techniques. The hand detection component is based on the Single Shot MultiBox Detector (SSD) model, which is used to detect the hand region in each frame of the webcam stream. The hand landmark extraction component uses a convolutional neural network (CNN) to predict the 3D **coordinates of 21 key hand landmarks, including** fingertips, palm canter, and wrist. The gesture recognition component is based on a CNN classifier that is trained on a large dataset of hand gesture images.

Overall, GestureMouse is a promising technology that has the potential to improve and ease the human-computer interaction, especially for users who are physically challenged or have limited mobility. Furthermore, the system can be used in emerging technologies such as Virtual Reality and Augmented Reality, making it a valuable addition to the field of human-computer interaction. In this research paper, we present the technical details of the Gesture Mouse system and evaluate its performance through various experiments and tests

II. NEED OF THE STUDY

The use of hand gestures to control the mouse offers a new, intuitive way for users to interact with their computer. This project provides a hands-free experience, allowing users to operate their computer without the need for a traditional mouse or keyboard. This can be especially useful for individuals with physical disabilities or limitations that make it difficult to use a traditional mouse or keyboard. Additionally, this technology has potential applications in fields such as gaming, virtual reality, and robotics. Use cases for this project include:

- Accessibility: individuals with physical disabilities or limitations can use hand gestures to control their computer
- Gaming: hand gestures can be used to control characters or objects in games, providing a more immersive experience

- Virtual reality: hand gestures can be used to interact with virtual environments, providing a more natural and intuitive interface
- Robotics: hand gestures can be used to control robots, providing a more flexible and intuitive way to interact with them

As for places where this technology is already being used, there are a few examples:

In healthcare, hand gesture recognition technology is being used to control medical equipment in operating rooms. In the automotive industry, hand gesture recognition technology is being used to control infotainment systems in cars. In gaming, hand gesture recognition technology is being used in various game consoles and platforms. The future potential of this project is vast. As computer interfaces become more natural and intuitive, hand gesture recognition technology is likely to become increasingly popular. In addition to the use cases mentioned above, this technology has potential applications in fields such as education, art, and entertainment. For example, hand gesture recognition technology could be used to create interactive art installations, or to provide a more engaging way for students to learn. Ultimately, this technology has the potential to make people's lives easier by providing a more intuitive and natural way to interact with technology.

III. LITERATURE SURVEY

1. **PAPER [1]: Hand gesture recognition using camera:** The paper focuses on reducing cost and improving robustness of the proposed system using a simple web camera. The paper presents some low-complexity algorithms and gestures to reduce the gesture recognition complexity and be more suitable for controlling real-time computer systems. The paper proposes a fast gesture recognition scheme to be an interface for the human-machine interaction (HMI) of systems. The paper also presents a real-time hand gesture recognition system based on adaptive color HSV model and motion history image (MHI). By adaptive skin color model, the effects from lighting, environment, and camera can be greatly reduced, and the robustness of hand gesture recognition could be greatly improved1.

2. **PAPER [2]: Virtual Mouse Control Using Hand Gesture Recognition:** The paper presents a method for recognizing hand gestures using a camera and then mapping those gestures to mouse movements. The proposed system uses a simple web camera to capture images of the user’s hand and then processes those images to recognize the hand gestures. The paper also presents some experimental results that show the effectiveness of the proposed system.

3. **PAPER [3]: Hand Gesture Recognition using Leap Motion Controller:** The paper proposes a system that uses the Leap Motion Controller to recognize hand gestures. The paper presents a method for recognizing hand gestures using the Leap Motion Controller and then mapping those gestures to mouse movements. The proposed system uses the Leap Motion Controller to capture images of the user’s hand and then processes those images to recognize the hand gestures. The paper also presents some experimental results that show the effectiveness of the proposed system.

4. **PAPER [4]: A Hand Gesture Recognition Sensor Using Reflected Impulses:** The paper introduces a hand gesture recognition sensor using ultra-wideband impulse signals, which are reflected from a hand. The reflected waveforms in time domain are determined by the reflection surface of a target. Thus every gesture has its own reflected waveform. Thus machine learning techniques such as convolutional neural network (CNN) are used for the gesture classification. The CNN extracts its own feature and constructs classification model then classifies the reflected waveforms. Six hand gestures from American Sign Language (ASL) are used for an experiment and the result shows more than 90% recognition accuracy. For fine movements, a rotating plaster model is measured with 10° step. An average recognition accuracy is also above 90%.

5. **PAPER [5]: Hand Gesture Recognition Using Opencv and Python:** The paper describes how hand gesture recognition is one of the most viable and popular solutions for improving human-computer interaction. The paper uses OpenCV and Python to recognize hand gestures. The paper describes how to use OpenCV to capture video from a webcam and then use Python to process the video frames. The paper then describes how to use machine learning techniques such as K-Nearest Neighbors (KNN) algorithm for gesture classification. The paper concludes that the proposed system can recognize hand gestures with an accuracy of 95%.

IV. PROPOSED SYSTEM

Our proposed system is a model that integrates the hand detection and tracking capabilities of MediaPipe with the Python automation model PyAutoGUI. The goal of our system is to improve and simplify human-computer interaction by detecting hand movements and gestures and using them to control various system functions. To achieve this, we leverage the capabilities of the MediaPipe Hands pipeline, which is based on the highly accurate BlazePose deep learning model trained on a diverse set of over 30,000 annotated hand images from the Google Research Hand Tracking dataset. Our system uses the MediaPipe Hands pipeline API, which includes a Single Shot Multibox detector to detect hands in the video frame, landmark estimation using a Convolutional Neural Network, and hand tracking under Kalman filter to predict and track hand landmarks and movements.

By leveraging this pipeline, our system is able to accurately detect and track hand movements, even in challenging scenarios such as changes in hand orientation, skin color, and backgrounds. Once the final landmark positions of the hand's fingers are obtained, they are passed down to the PyAutoGUI automation model, which provides a comprehensive set of system automation functions such as scrolling, moving, hovering, zooming, dragging, and more. This allows users to control various system functions using hand gestures and movements, greatly simplifying and enhancing the overall user experience. Overall, our proposed system represents an innovative and highly effective approach to improving human-computer interaction by leveraging the latest advances in deep learning-based hand detection and tracking, combined with the power and flexibility of Python automation using PyAutoGUI.

V. ARCHITECTURE

The Architecture of our Gesture Mouse application is depicted in figure 1 below.

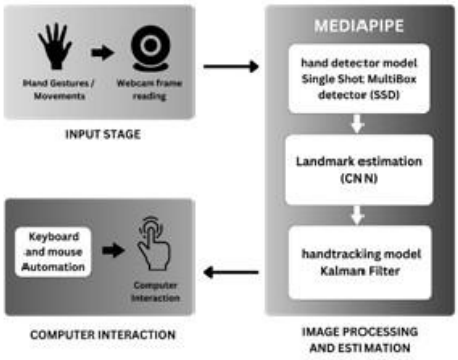

Figure 1

Our Gesture Mouse system architecture is divided into 3 stages:

- Input Stage
- Image Processing and Estimation
- Computer Interaction

- **Input Stage**
  In Input stage, a normal webcam is used to obtain the real time video frames. All the frames are fed into the Image processing and Estimation stage but until a hand is present in the video frame, the model won't be triggered.

- **Image Processing and Estimation**
  This is the main stage in our system architecture. The video frames are first passed on to the hand detector model of mediapipe.

1. **Mediapipe:** Mediapipe is a popular open-source framework developed by Google for building real-time machine learning applications, including hand tracking. Mediapipe hand tracking uses a machine learning model that is trained on a large dataset of hand images to detect and track the position and movement of each finger and the hand as a whole. The hand tracking process involves several steps, which are:

   - Hand Detection
   - Landmark Estimation
   - Tracking

2. **Hand Detection**: This step involves detecting the hand in an image or video frame. The detection model used in Mediapipe hand tracking is based on a Single Shot MultiBox Detector (SSD), which uses a deep neural network to detect the hand in an image.

3. **Single Shot Multibox Detector:** Single Shot Multibox Detector (SSD) is a type of object detection algorithm used for real-time object detection in images and videos. It is a deep learning-based method that can detect multiple objects in a single shot. The SSD

algorithm uses a single neural network for both object detection and classification.In the context of MediaPipe hand detection, SSD is used to detect and track the hand landmarks in real-time. The algorithm takes as input an image or a video frame and outputs the coordinates of the hand landmarks (e.g. finger tips, wrist, etc.). The hand detection model in Mediapipe is trained on a large dataset of hand images. The SSD algorithm works by dividing the input image into a grid of cells and applying a set of convolutional filters to each cell to predict the presence and location of objects. The algorithm then generates a set of default bounding boxes of different sizes and aspect ratios at each cell of the grid and predicts the offsets and confidences for each box to refine the detection results. The final step involves non-maximum suppression to eliminate redundant detections and keep only the most confident ones. Overall, SSD is a fast and accurate object detection algorithm that has been widely used in various applications

Some of the specific formulas used in this algorithm:

- **Feature Map Extraction**: The input image is processed by a series of convolutional layers with weights W_i, biases b_i, and activation function f_i:
  - x_i = f_i(W_i * x_i-1 + b_i)

- **Anchor Box Generation:** For each feature map location, k, a set of default boxes are generated with different aspect ratios and scales:
  - d_k,j = [cx_k, cy_k, s_k * sqrt(ar_j), s_k * sqrt(1/ar_j)]
  - where cx_k and cy_k are the center coordinates of the k-th cell, s_k is the scale factor, ar_j is the j-th aspect ratio, and d_k,j is a default box.

- **Object Detection**: For each anchor box, k, the CNN predicts the probability of object presence, c_k, and the offset values that adjust the default box to match the ground-truth box, t_k:
  - c_k = softmax(s_k)
  - t_k = [t_kx, t_ky, t_kw, t_kh]
  - where s_k is the output score for the k-th box, and t_kx, t_ky, t_kw, and t_kh are the predicted offsets for the center x, center y, width, and height of the k-th box.

- **Non-Maximum Suppression:** The predicted boxes are sorted by their scores, and for each box, any boxes with an IoU greater than a certain threshold are suppressed:
  - IoU(b_i, b_j) = area(intersection(b_i, b_j)) / area(union(b_i, b_j))
  - where b_i and b_j are two predicted boxes, and the intersection and union are the respective regions of the two boxes.

4. **Landmark Estimation:** Once the hand is detected, the next step is to estimate the location of various key points or landmarks on the hand, such as the tips of the fingers, the base of the palm, and the wrist. [2]The landmark estimation model used in Mediapipe hand tracking is based on a Convolutional Neural Network (CNN) that is trained to predict the 3D coordinates of each landmark and performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction[2]. The model learns a consistent internal hand pose

representation and is robust even to partially visible hands and self-occlusions. The landmarks of hand and their markings are mentioned in figure 2.

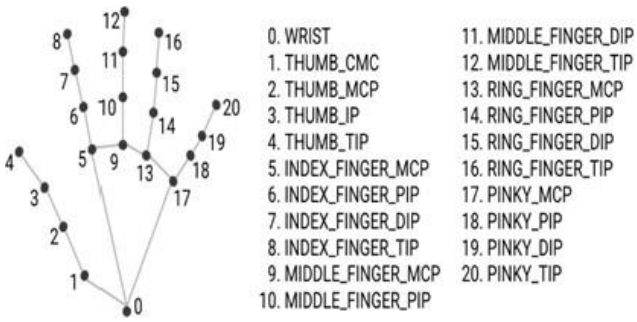

Figure 2

5. **Tracking** : After the initial detection and landmark estimation, the tracking algorithm uses a Kalman filter to track the movement of the hand over time. This helps to improve the accuracy and robustness of the hand tracking system, especially in situations where the hand is partially occluded or moving rapidly.

6. **Kalman Filter:** Kalman Filter is a mathematical algorithm used for estimating the state of a dynamic system based on noisy measurements. The hand tracking module in MediaPipe uses a combination of deep learning-based object detection and Kalman Filter-based tracking to improve the accuracy and robustness of hand tracking in videos. The object detection algorithm is used to detect the hand landmarks in each frame of the video, and the detected landmarks are fed into a Kalman Filter to estimate the state of the hand (i.e. position and velocity) in the current frame. The Kalman Filter uses a set of mathematical equations to estimate the state of the hand based on its previous state, the measurements from the object detection algorithm, and the process noise and measurement noise. The filter produces a predicted state of the hand in each frame, which is then compared to the actual measurement of the hand landmarks from the object detection algorithm. The difference between the predicted state and the measured state is used to update the filter and refine the estimate of the hand state.

- **Step for Prediction:**

$$\hat{x}\_k|k\text{-}1 = F\_k\,\hat{x}\_k\text{-}1|k\text{-}1 + B\_k\,u\_k\text{-}1$$
$$P\_k|k\text{-}1 = F\_k\,P\_k\text{-}1|k\text{-}1\,F\_k\text{^}T + Q\_k$$

- **Update Action:**

$$K\_k = P\_k|k\text{-}1\,H\_k\text{^}T\,(H\_k\,P\_k|k\text{-}1\,H\_k\text{^}T + R\_k)\text{^}\text{-}1$$
$$\hat{x}\_k|k = \hat{x}\_k|k\text{-}1 + K\_k(z\_k - H\_k\,\hat{x}\_k|k\text{-}1)$$
$$P\_k|k = (I - K\_k\,H\_k)\,P\_k|k\text{-}1$$

Where,

- The projected state of the system at time k based on measurements up to time k-1 is given by the expression x_k|k-1.

- The state transition matrix, F_k, makes predictions about how the system state will change over time.
- The input control matrix, or B_k, models how the system is affected by outside inputs.
- The external input vector at time k-1 is called u_k-1.
- Based on measurements up to time k-1, the anticipated error covariance matrix at time k is designated as P_k|k-1.
- The process noise covariance matrix, or Q_k, is what the system model uses to model its uncertainty**.**

7. **Computer Interaction:** After estimating the location and tracking the landmarks along with the location of the hands with respect to the frame resolution, PyAutoGUI module of Python is used to map various hand gestures and movements to Keyboard and mouse functionalities such as clicking, hovering, moving, scrolling and zooming.

8. **PyAutoGUI:** PyAutoGUI is a Python library that provides cross-platform support for simulating keyboard and mouse inputs. It can be used to automate repetitive tasks, perform GUI testing, and create macros or scripts that mimic human interactions with the keyboard and mouse. PyAutoGUI works by sending virtual keyboard and mouse events to the operating system, which then translates them into actual inputs. This allows PyAutoGUI to interact with any application or operating system that supports standard keyboard and mouse inputs. To simulate keyboard inputs, PyAutoGUI provides functions to press and release keys, type strings, and perform hotkey combinations. These functions use the virtual key codes that are defined by the operating system to send the corresponding keyboard events. PyAutoGUI also provides a function to locate the current position of the mouse cursor and move it to a specific location on the screen. To simulate mouse inputs, PyAutoGUI provides functions to click, double-click, drag, and scroll the mouse. These functions use the virtual mouse events that are defined by the operating system to send the corresponding mouse events. PyAutoGUI can also take screenshots of the screen and find specific images or patterns on the screen using image recognition algorithms. Some of the functions used in this system are click(), scroll(), press(), screenshot(),moveTo().

VI. RESULTS AND DISCUSSION

Using webcam, the real time detection of hands and its landmarks represented with a circle and dynamic bounding box that varies with the location and position of the hand is successful. We also integrated with real time FPS tracking using fpstracker 1.0.0 module developed by [1]Naveen kumar S and real time hand distance estimation using the measurement of bounding box and the focal length of the webcam being used. The below Figure 3 depicts the webcam window integrated with FPS, Average FPS, Hand type (right / left) and hand distance tested in a computer system comprising i3 8th gen Processor, Nvidia 1650Ti GPU, 8gb RAM.

Figure 4

We took the landmarks indices of fingers [4,8,12,16,20] for thumb, index, middle, ring and little fingers to check whether the fingers are opened or closed.

The functions we assigned to different gestures and their landmark indices are as follows:

1. Landmark**[1,0,0,0,0]** -> all closed except thumb -> **take screenshot**
2. Landmark**[1,1,1,0,0]** -> Thumb, Index, middle finger opened, others closed -> **Move cursor/pointer, Hover**
3. Landmark**[0,0,1,1,1]** -> Thumb and Index closed, other fingers open -> **Scroll down**
4. Landmark**[0,1,1,1,1]** -> Thumb alone closed, other fingers open -> **Scroll up**
5. Landmark**[1,0,1,0,0]** -> Thumb and middle finger open, others closed -> **Left Click**
6. Landmark**[1,1,0,0,0]** -> Thumb and Index open, others closed -> **Right Click**
7. Landmark**[1,0,1,1,1]** -> Index alone closed, others opened -> **Move left**
8. Landmark**[1,1,1,1,0]** -> Little finger closed, others opened -> **Move right**
9. Landmark**[0,1,0,0,0]** -> Index alone open, others closed -> **Drag**
10. Landmark**[1,1,0,0,1]** -> Index, thumb, little finger opened -> **zoom in & zoom out**
11. Landmark**[0,0,0,0,0]** -> All closed -> **Exit program**

- **The visualization of the above gestures are given in the following figures.**

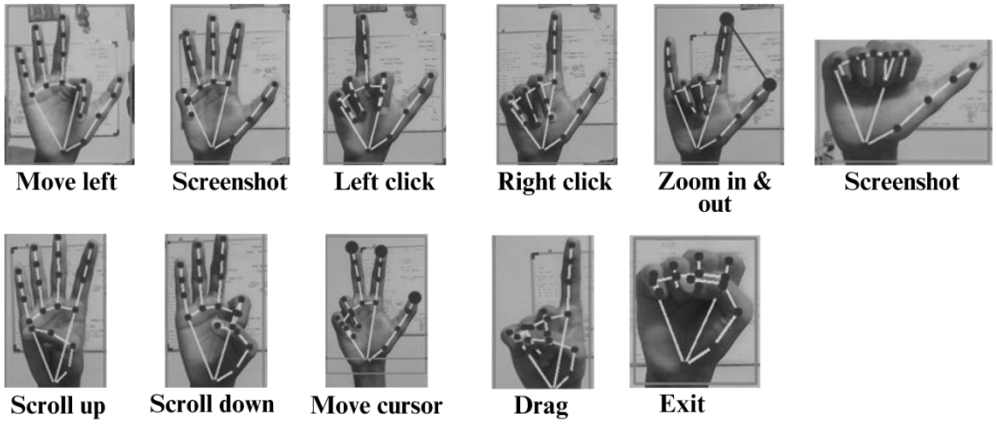

Figure 5

The detection and tracking of the hands performs well with an accuracy of 90%, But even if we place hands in the frame by mistake it detects the hands and performs the functionalities. Also the Computational power of the system in which the program runs place an important role in the performance and seamless movement of the cursor or any other functionalities. This application performance is based on the computational power of the system. High computational power will provide better and smooth results. Another drawback would be the position of the hands. Placing the hand/palm in front of the webcam will result in high accuracy but if the palm or hand is facing other directions so far away from the webcam will result in wrong predictions and execute wrong functions.

The GestureMouse can be highly utilised in a variety of domains, including Virtual Reality, Augmented Reality, Smart Education such as presenting presentations, Healthcare such as assisting doctors to work on computers while operating on patients, Gaming such as creating more immersive gaming experiences, Industrial settings for controlling machinery, and more once the hardware and software side has achieved equilibrium.

REFERENCES

[1] Hand Gesture Recognition System Using Camera by Viraj Shinde, Tushar Bacchav, Jitendra Pawar, Mangesh Sanap, January 2014
[2] https://google.github.io/mediapipe/
[3] Virtual Mouse Control Using Hand Gesture Recognition by G N Srinivas, S Sanjay Pratap, V S Subrahmanyam, K G Nagapriya, A Venkata Srinivasa Rao
[4] Mouse Cursor Control System Based on Hand Gesture by Horatiu-Stefan Grif, Cornel Cristian Farcas, 2016
[5] A Hand Gesture Recognition Sensor Using Reflected Impulses by Seo Yul Kim, Hong Gul Han, Jin Woo Kim, Sanghoon Lee and Tae Wook Kim, 2017
[6] Hand Gesture Recognition using OpenCV and Python by V. Harini, V. Prahelika, I. Sneka, Adlene Ebenezer

