Gesture Controlled Mouse Navigation : Hand Landmark Approach

Shaurya Guliani
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Uttar Pradesh
Uttar Pradesh, India
shauryaguliani@gmail.com

Renuka Nagpal
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Uttar Pradesh
Uttar Pradesh, India
rnagpal1@amity.edu

Rajni Sehgal
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Uttar Pradesh
Uttar Pradesh, India
rsehgal@amity.edu

Abhishek Singhal
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Uttar Pradesh
Uttar Pradesh, India
abhi087@gmail.com

***Abstract*— Euclidean Distance has been an important metric for the calculation of distances between points on a scaled space. Recent times have seen an increased use of this metric for the purposes of pattern recognition. Hand Landmarks are a group of twenty-one points which define the skeletal structure of a human's hand. Calculation of distances between Hand Landmarks using Euclidean Distance can cater to the need of an efficient algorithm which could recognize patterns using mathematical calculations. The last few years have witnessed the introduction of technologies which promoted virtualization of important hardware thereby, increasing the portability and handling of the systems. The Mouse is an important aspect of the computer which helps to easily interfacing with the system. History has seen the use of trackballs, laser sensors and track pads as means of controlling the system however, each one of them consisted of costly hardware equipment be it the material or the sensors used. This paper proposes a virtual mouse system which can be controlled and navigated using gestures made using any one hand, whether left or right, without the involvement of any external hardware or sensor. All this is done by means of Euclidean distances and hand landmarks. It allows the user to control all the functioning of the mouse including its movement, mouse clicks and drag and drop without physically touching any hardware. It uses only the system's inbuilt camera to carry-out the functioning of the mouse. The system is carried out using Python along with OpenCV, Mediapipe and the PyAutoGui libraries. All the gestures become visible on the camera screen with appropriate marking to facilitate representing the working of the system.**

Keywords — Open CV, Hand Landmarks, Mediapipe, PyAutoGui, Gesture, Recognition, Mediapipe Hands

I. INTRODUCTION

Introduction of Graphical User Interfaces (GUIs) changed the way users interfaced with the computer. It made the handling and control of the systems very easy and interesting. However, in this modern era, day to day life is becoming smarter and interlinked with technology. The world is coming up with new ideas and techniques for making systems and devices more portable and interactive. In times of such advancement where portability is all around and computer systems are almost being wed in each field, the use of an actual hardware mouse might be difficult in areas where computers are of great importance but the use of a mouse to control them might be challenging. The concept of virtualization has evolved the landscape of the technological industry with the introduction of systems with minimal hardware and increased functionality. The proposed system removes a major hardware component, the mouse, from the system without terminating its functionalities.

The aim of the paper is to facilitate the development of systems which could be controlled without the involvement of a physical mouse but could be navigated just using the hand of the user. This increases the accessibility of the system to be used at places where being near to the system might not be always feasible. The system would only require a connected camera to record the user's input. This even helps in cost cutting of the systems as no actual hardware or sensor is being used to facilitate the system's functioning.

The paper is organized into four sections. Section II, "Related Work", gives a brief account of all the previous work and developments that has been done with the objective of solving the mentioned problem. Section III, "Methodology", gives detailed step-by-step insights onto the process of hand gesture recognition and performing the required task followed by "Results and Discussion" in section IV which discusses about the working of the system along with some snaps demonstrating its use. Section V, "Conclusion and Future Scope", summarizes the whole paper into few lines specifying its need in the upcoming technologies and discusses some future ideas which can be worked upon the project.

II. RELATED WORK

Gesture Recognition is the main component of the virtual mouse. Many techniques and different algorithms have been used for effective recognition of different gestures which can be categorized into two main categories - Attached Sensor based and Computer Vision based.

The sensor-based approach that makes use of coordinal positioning of palm and fingers was published by Munir Oudah et al [1]. This is done with the help of physical sensors incorporated in a glove. Inertial Measurement Unit (IMU) sensors along with Dynamic Time Warping technique have

been previously used while incorporating the Restricted Column Energy (RCE) Neural Networks to carry out Real-time Gesture Recognition [2]. Enea Ceolini et al [3] used electromyography for gesture recognition. They published their work on transmission of signals by muscles during different positions they take and how they can help in recognition of certain gestures. Deep Gesture Algorithm which uses Deep Learning for gesture recognition with the help of motion sensors was published by Ji-Hae Kim et al [4]. This approach used Recurrent Neural Networks (RNNs) to incorporate long term time dependencies and used data obtained through gyroscope and accelerometer sensors.

Computer vision has seen a rise of different approaches for gesture recognition over time. Ojha et al [5] showcased the use of histograms created using the outlines of the gesture made for gesture recognition by comparing them to records of a predefined database. Deep Learning allowed incorporation of neural networks in gesture recognition. implemented. Gesture recognition with use of convolutional neural networks using color segmentation and polygonal approach to classify gestures into categories was implemented by Raimundo F. Pinto et al [6]. The 3D hand model-based approach incorporating Convolutional Neural Networks, published by A. Mohanrathinam et al [7] which converts 3-D input into a 2-D contour which is then processed to recognize the gesture. The introduction of image processing allowed the use of gestures as an effective way to navigate control of a system. Dinh-Son Tranet al [8] proposed a system which uses RGB (Red green Blue) and Convolutional Neural Networks for the driver program to understand the gestures. Here, colored caps or tapes are used on fingertips for the system to recognize the gesture.

The proposed system makes use of hand landmarks to recognize gestures using MediaPipe as discussed by Sachin Agrawal et al [9] in his research and using them to perform different mouse functions. The methodology regarding the mentioned approach is discussed in the following section.

III. Methodology

The system is developed using the Python Programming language along with OpenCV, Mediapipe Hands and PyAutoGui. OpenCV plays an important role in capturing the frames from the system's camera and further process them so that they could be used effectively by Mediapipe for gesture capturing. Mediapipe Hands capture the gestures in the processed frames provided by OpenCV to recognize the gesture created by the user and use it as the basis of the action to be performed by the mouse. Mediapipe Hands processes the image of the hand as a network of twenty-one points and uses their positioning and relative distance between them for gesture recognition. Once the gesture is recognized, PyAutoGui takes control of the system and performs the required operation as mapped to the recognized gesture.

A. Computer Vision

Open CV stands for "Open-Source Computer Vision". It is a library of algorithms which are pre-defined and ready to use mainly for computer vision applications like object detection, face detection, people/vehicle counting, biometric recognition etc. Open CV has made it easier for the new AI recognition systems to adopt previous model frameworks. The adopted frameworks can be used unchanged or with minor modifications hence, saving the time to make a similar code again [10]. The Open CV library takes control of the camera and is used for capturing input provided by the user as frames.

B. Mediapipe

The MediaPipe Framework is used to create machine learning pipelines for processing time-series data, including audio, video, and other types. This cross-platform framework is compatible with embedded devices like the Raspberry Pi and Jetson Nano as well as desktop/server, Android, and iOS systems [11]. MediaPipe consists of the 'Mediapipe Hands' library which is used during the Hand Detection and Gesture Recognition processes. It also provides with the 'Mediapipe Solutions Draw Utils' library which is used for mapping different landmarks to different gestures.

C. Mediapipe Hands

Mediapipe Hands provides an accurate solution for hand and finger tracking. It uses Machine Learning (ML) to capture and track twenty-one points from the face of the hand. These twenty-one points make up the skeletal structure of the bones in the hand which are seen as places having different intensities of light. Mediapipe Hands uses two different models for correct implementation of Gesture Recognition:

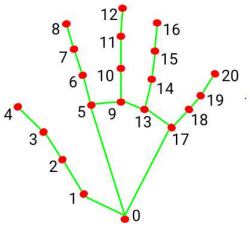
1) Palm detection Model

The Palm detection model uses Harr-like features, as used for tradition face detection for detecting the palm. The palm detector primarily identifies the hands, and the hand-skeleton finger tracking model then does exact key-point localization, 3D hand key-points per detected hand. The model is a single-shot detector model intended for mobile real-time usage to detect initial hand positions. The issue of detecting hands is incredibly challenging since both - lite model and complete model - must be able to recognize occluded and self-occluded hands while operating across a wide scale span (twenty times) relative to the image frame. It is somewhat challenging to identify hands correctly from their visual characteristics alone, in contrast to faces, which exhibit high contrast patterns, such as in the region of the lips and eyes. Accurate hand localization is instead made possible by adding additional information, such as aspects of the arm, torso, or person. The approach employs many techniques to overcome the forementioned problems. First, a palm detector rather than a hand detector is trained since it is far easier to estimate the bounding boxes of stiff objects like fists and palms than it is to recognize hands with articulated fingers. The non-maximum suppression technique also performs well in two-hand self-occlusion scenarios like handshakes since palms are smaller objects. The number of anchors may be reduced by a factor of three to five by modelling palms with square bounding boxes, or anchors in machine learning language. In order to provide more scene context awareness, even for little objects, an encoder-decoder feature extractor is implemented. Lastly, in order to support a high number of anchors, the focus loss during training is limited. With the forementioned methods, the model can detect palms with an average precision of 95.7%. With no decoder and a standard cross entropy loss, the baseline is just 86.22% [12]. The Palm Detection Model is used for the segmentation of the palm so

that it could be processed for detecting landmarks and recognizing gestures.

2) Hand Landmark Model

The Hand landmark Model returns the coordinates of the twenty-one points on the palm to be processed and used. After detecting the palm over the whole picture, the subsequent hand landmark model uses regression, or direct coordinate prediction, to accomplish exact key-point localization of twenty-one 3D hand-knuckle coordinates inside the identified hand areas. The model acquires a reliable internal hand posture representation and is unaffected by self-occlusions or partially visible hands. Twenty-one 3D coordinates to around thirty thousand real-world photos are added to acquire ground truth data, as seen below (we take Z-value from image depth map, if it exists per corresponding coordinate). A high-quality synthetic hand model is rendered over a variety of backdrops and mapped to the associated 3D coordinates in order to better cover the range of potential hand positions and give extra oversight on the nature of hand geometry [12]. The Hand Landmark Model is the core of the system as it helps in extracting the twenty-one landmarks from the face of the palm segment and use them for recognizing the gestures. Fig. 1 shows the mapping of all the landmarks retrieved by Mediapipe Hands from the image of the palm.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fig. 1: Hand Landmarks retrieved from input image by Mediapipe Hands

D. PyAutoGui

PyAutoGUI is a python module which takes over the control of the mouse and keyboard and co-ordinates with python to perform necessary executions [13]. The platform can handle and control almost each operation of the mouse and keyboard from python with accuracy. The PyAutoGui library performs the execution of mouse functions according to the recognized gestures.

The mentioned libraries and frameworks integrate the whole system of recognition of gestures and implementation of mouse functions which is carried out in the following steps as shown in Fig. 2:

- Frame Capturing
- Hand Recognition
- Gesture Recognition
- Gesture Representation
- Euclidian Distance
- Mouse Function Execution

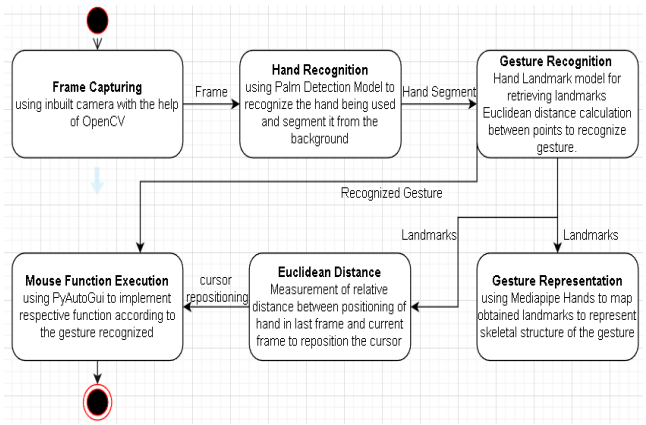


Fig. 2. Flow Chart describing the flow of processes during the gesture recognition and execution of mouse function.

1) Frame Capturing

Frame Capturing is carried out by OpenCV for pre-processing to be sent forward for gesture recognition. The "VideoCapture" function of OpenCV continuously uses the camera for recording frames. Once the frame is recorded, it is stored as an object to be used by Mediapipe Hands for gesture recognition and Hand detection.

2) Hand Recognition

Once the frame is obtained, Mediapipe with its "mediapipe solutions hands" module comes into play to recognize the hand, whether left or right, being used for the navigation purpose to avoid confusion. This is done by observing the positioning of the twenty-one points on the palm which help understand where the thumb and other fingers are present.

3) Gesture Recognition

Mediapipe Hands provides the co-ordinates of twenty-one points of the hand extracted from the frame. The positioning of these twenty-one points along with the relative Euclidean distance between them is used to identify the gesture being made. All the gestures are assigned a numeric value for easy recognition and mapping of mouse controls. The gestures are predefined with numeric values by Mediapipe and these values are returned according to the gestures recorded on the screen. For example, the gesture of showing the little finger is assigned with the value "1" while the gesture showing the index finger is assigned with the value "8". These numeric values are mapped to different mouse controls for facilitating full functionality.

4) Gesture Representation

Once the gesture is recognized, it is represented on the camera screen using twenty-one points and connected lines which showcase how the gesture looks like graphically. This task is accomplished by the "Mediapipe Solutions Drawing Utils" module of Mediapipe. It is used for tracing hand position of objects of Mediapipe hands to establish an accurate representation.

5) Euclidian Distance

Once the gesture is recognized, the movement of the cursor according to the movement of the hand is to be calculated. For this, the displacement within one single frame of one single coordinate returned by Mediapipe hands is to be calculated. This distance is calculated using the traditional

Euclidian Distance of the co-ordinate system. The process of gesture detection and Euclidian distance measurement cycles through-out the operation of the system.

6) Mouse function Execution

Final step after gesture recognition is to execute the command associated with the gesture. This can be accomplished by using the "PyAutoGui" library which takes control of the system and operates it according to the command given. This is the final step of gesture-controlled mouse execution which generates the final output.

The proposed methodology facilitated the creation of a system which allows the user to interface with the system using his hand without the need of any hardware. The result and implementation of the whole system is discussed in the subsequent section.

IV. Results and Discussion

The whole system makes use of the camera of the system to take the gesture input from the user. The executable file of the system launches the camera and opens a screen displaying the camera's point of view of the user as shown in Fig. 3. It allows the user to estimate the workable area within which the system can take gestures as input.

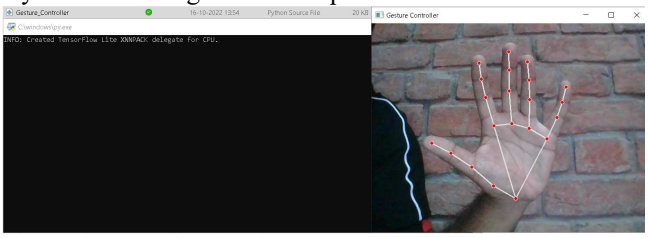


Fig. 3: The system launches the camera on startup to take gesture input

Once the camera is launched, the system is ready for taking gestures as input. The hand landmarks can be observed on both hands if they are put in front of camera (stretched out). When the mouse is not to be used but the hand is in front of the camera, the neutral gesture can be used (stretched-out hand) as demonstarted in Fig. 4.

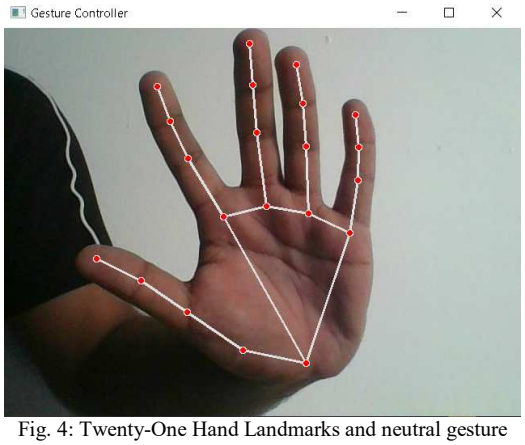


Fig. 4: Twenty-One Hand Landmarks and neutral gesture

The following getsures are used to carryout the specified mouse functions :

1) Cursor Movement

The victory gesture (Fig. 5) is required to activate the cursor movement and the movement of the hand moves the cursor. The victory getsure is a semi-neutral gesture which does not perform any mouse clicks but only repositions the mouse according to the user's will.

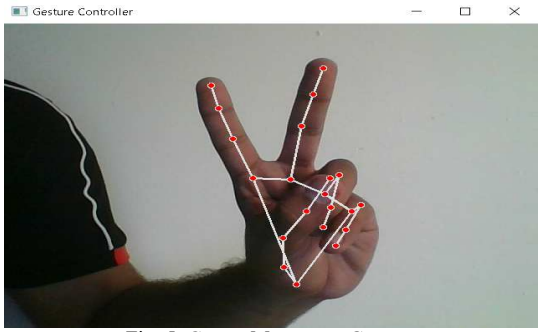


Fig. 5: Cursor Movement Gesture

2) Left Click

Left click is the major action click of the mouse. It performs a single click only and needs to be repeated to perform multiple clicks. It is performed by using the victory gesture and then folding the index finger (Fig. 6).

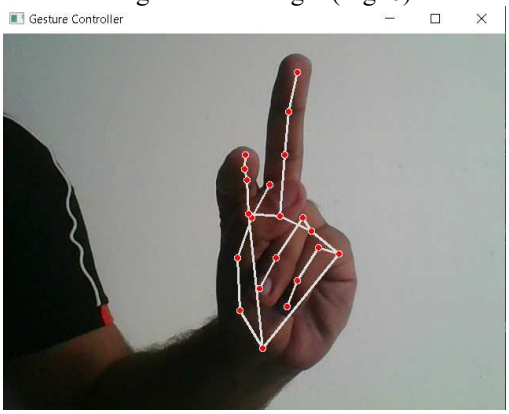


Fig. 6 : Left Click Gesture

3) Right Click

Right click is also known as the secondary click and majorly aims at openeing the context menu of any file or interface. Similar to the left click, the right click also performs a single click only. It is performed by using the victory gesture and then folding the middle finger (Fig. 7).

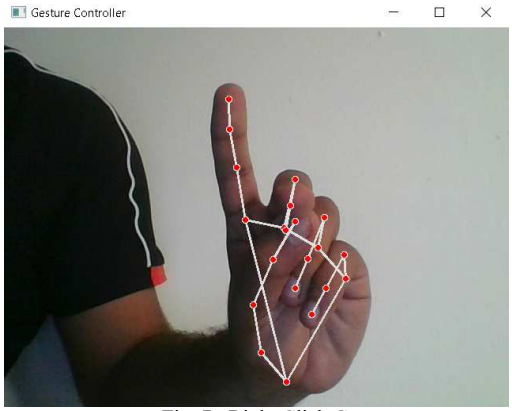


Fig. 7 : Right Click Getsure

4) Double Click

Double click is traditionally used to execute any function associated with the item being clicked like

executing a button or opening a file. It executes two left clicks one after the other with the help of a single gesture. It is performed by using the victory gesture and then joining the index and the middle finger together (Fig. 8).

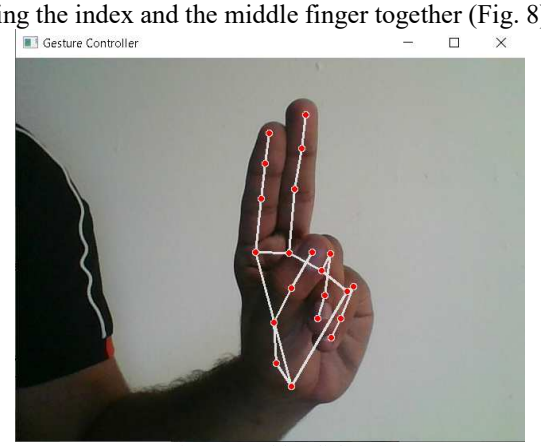


Fig. 8 : Double Click Gesture

5) Drag and Drop

Drag and Drop is performed by using the cursor movement to go onto the item to be dragged and then creating the fist getsure (Fig. 9) to select the item. Then the fist should be moved to drag the item accordingly and the fist is released into any other getsure to complete the drag and drop function.

Fig. 9 : Drag and Drop Gesture

6) Multiple Item Selection

Similar to Drag and Drop, multiple item selection can be perfomed by making a fist and then dragging the mouse to create a selection box to select multiple items.

These mouse functions combined allows the user to perform mouse opereratiosn upto a huge extent. However, there is scope for more functions to be introduced which could make the screen navigation easier like separate functions for scrolling and other important fuctions like brightness and volume control. A conclusive summarization along with an overview of different ideas regarding the future scope of the project is discussed in the next section.

V. Conclusion and Future Scope

The paper proposed the idea of virtual mouse that can be controlled with one hand only without the need of any sensors or external hardware. It is better than all other virtual techniques as it lacks the use of any external entity to facilitate gesture recognition. It uses three basic libraries for implementation of the functions accurately: OpenCV – for capturing of video frames and using them for detection purposes, Mediapipe Hands – for using the captured frames for hand and gesture recognition using the twenty-one co-ordinate points on the hand returned by it and PyAutoGui – which implements a rule-based algorithm to perform mouse implementations. The input images are processed through four stages for the recognition of the gesture namely: Frame Capturing, Hand Recognition, Gesture Recognition and Gesture Representation, followed by the stage to calculate the Euclidean distance moved to reposition the cursor. Finally, the gathered information is used in the function execution stage to execute the mouse function. Real-time visual representation of gestures along with the positioning of the hand landmarks are thoroughly discussed along with supporting images.

The above proposed model on Hand Gesture Controlled mouse navigation still has a lot to explore. As Image processing is being used to retrieve Hand Landmarks from the user's hand, the system fails if used while wearing external wearables like gloves as the model fails to incorporate different features of the hand except the shape. Hence, a different gesture-based approach which could work with wearables can be synthesized. The functionality of the system can be improved by introducing more gestures which could improve the user's control on the computer. The Hand Landmark approach helps in recognition of multiple gestures, many which are unassigned with any functionality in this model, thereby leaving scope for inclusion of many other important functions of the computer system. The system uses the mathematical approach of gesture recognition with limited use of image processing. Advancements in Machine Learning has shown that each solution has an efficient and faster option. New techniques should be constantly studied for the betterment of the gesture recognition problem.

References

[1] Oudah, Munir, Ali Al-Naji, and Javaan Chahl. "Hand gesture recognition based on computer vision: a review of techniques." *journal of Imaging* 6, no. 8 (2020): 73.

[2] Kim, Minwoo, Jaechan Cho, Seongjoo Lee, and Yunho Jung. 2019. "IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces" *Sensors* 19, no. 18: 3827. https://doi.org/10.3390/s19183827

[3] Ceolini, Enea, Charlotte Frenkel, Sumit Bam Shrestha, Gemma Taverni, Lyes Khacef, Melika Payvand, and Elisa Donati. "Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing." *Frontiers in Neuroscience* 14 (2020): 637.

[4] Kim, Ji-Hae, Gwang-Soo Hong, Byung-Gyu Kim, and Debi P. Dogra. "deepGesture: Deep learning-based gesture recognition scheme using motion sensors." *Displays* 55 (2018): 38-45.

[5] Ojha, Dharmaraj, and Rajesh George Rajan. "Histogram based human computer interaction for gesture recognition." In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 263-266. IEEE, 2019.

[6] Pinto, Raimundo F., Carlos DB Borges, Antônio Almeida, and Iális C. Paula. "Static hand gesture recognition based on convolutional neural networks." *Journal of Electrical and Computer Engineering* 2019 (2019).

[7] Mohanarathinam, A., K. G. Dharani, R. Sangeetha, G. Aravindh, and P. Sasikala. "Study on Hand Gesture Recoginition by using Machine Learning." In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1498-1501. IEEE, 2020.

[8] Tran, Dinh-Son, Ngoc-Huynh Ho, Hyung-Jeong Yang, Eu-Tteum Baek, Soo-Hyung Kim, and Gueesang Lee. "Real-time hand gesture spotting and recognition using RGB-D camera and 3D convolutional neural network." *Applied Sciences* 10, no. 2 (2020): 722.

[9] Agrawal, A. Sachin, Agnishrota Chakraborty, and M. Rajalakshmi. "Real-Time Hand Gesture Recognition System Using MediaPipe and LSTM." *Journal homepage: www. ijrpr. com ISSN* 2582 (2022): 7421.

[10] Gupta, Bhumika, Ashish Chaube, Ashish Negi, and Umang Goel. "Study on object detection using open CV-Python." *International Journal of Computer Applications* 162, no. 8 (2017): 17-21.

[11] Lugaresi, Camillo, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang et al. "Mediapipe: A framework for building perception pipelines." *arXiv preprint arXiv:1906.08172* (2019)

[12] Zhang, Fan, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. "Mediapipe hands: On-device real-time hand tracking." *arXiv preprint arXiv:2006.10214* (2020)

[13] Sweigart, Al. "PyAutoGUI Documentation." *Read the Docs* (2020): 25.