Gesture Controlled Interface Manipulation

Tom Barthelmeh
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
tba105@uclive.ac.nz

Richard Green
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

***Abstract*—This paper presents a method for gesture-controlled interface manipulation using hand tracking and gesture recognition. Utilizing Google's MediaPipe Hands library, the proposed solution detects hand landmarks from a standard-resolution webcam, allowing the solution to work on common hardware. By defining specific hand gestures, users can control common mouse actions, such as clicking and scrolling. As previous limitations cite jitter as being hard to overcome, this system incorporates an adaption on a one Euro filter for two dimensional co-ordinates through a L2 norm to reduce jitter, enhancing the accuracy and usability of cursor control. The system demonstrates a high precision in hand landmark detection and gesture recognition, making this approach accessible and efficient for everyday use.**

***Index Terms*—hand tracking, machine learning, gesture recognition, human-computer interaction**

I. Introduction

As the computer has evolved from simple computational machines to a staple in most households that is at the center of our social lives, methods to control and interact with the device using more intuitive methods have started to become more prominent. Instead of being confined to the standard keyboard and mouse, this paper offers a method to interact with the computer in a more intuitive way by using hand gestures. Previous methods show some promise but often require GPU processing for 3D reconstruction of the hand, or require the collection of training data to recognise gestures [1] [2]. This paper proposes a simpler method using a machine learning model for pose recognition and defined gestures for interface manipulation. It aims to improve on limitations of previous studies including jitter.

II. Background

Previous work by Jones and Rehg proposed the use of histogram models to detect skin pixels, achieving a detection rate of 80% [3]. The key to the 80% detection rate was to use a very large dataset of nearly 1 billion labelled pixels. The use of a large dataset is sometimes inhibiting, especially if trying to run a real-time application. They also note that improving the accuracy is likely to require the use of spacial recognition in addition to colour-based techniques. A different method to detect the hand used coloured gloves. Originally proposed by Wang and Popović for real-time hand tracking, Lamberti and Camastra used a woolen glove shown below in Figure 1 to track the hand and recognize its gestures [4] [5].

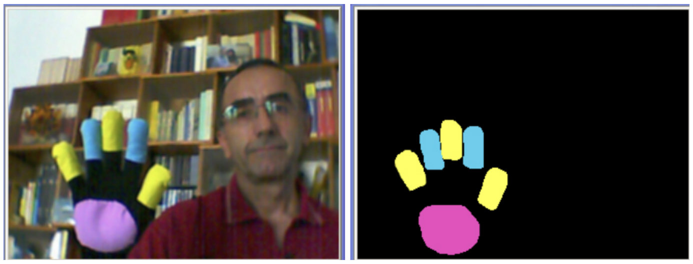


Fig. 1. Woolen glove (left) and segmented image (right)

Lamberti used the brightly coloured glove and threshold mapping with k-nearest-neighbours to extract the pixels of the hand, as well as morphological open followed by morphological close. The centroids of each of the shapes then gave way to the joint angles of each of the fingers from the palm. These values were then fed into a Learning Vector Quantization to classify the gestures. This method worked well in same-lighting conditions but is heavily dependant on the thresholding and therefore the lighting conditions. The user also had to have a woolen glove with the same coloured objects in order to use the algorithm. This makes it less accessible to the average user and can be cumbersome and innaccurate.

More recent work has improved on some of the issues previously mentioned. Many methods use depth cameras to accurately get the landmarks of the hand [6] [7] [8]. Even though these papers boast good results, the use of a depth camera makes the approach harder to use for the average consumer as depth cameras are not very common.

Methods that use neural network models have become a lot more common and are a good way to get generalised hand tracking in real-time if done right. OpenPose is one of the most widely used pose recognition libraries, with the official version likely being the most used version of OpenPose. However, this version of OpenPose requires high-end hardware to enable real-time pose recognition and is highly inefficient [9]. Lightweight versions of OpenPose don't offer certain features including hand-recognition [10]. OpenPose hand recognition algorithm also requires the position of the wrist, elbow, and shoulder to work properly.

MediaPipe is another, much more recent, pose recognition library that includes hand landmarks and was designed to enable fast processing even on low-end hardware [11]. MediaPipe abstracts individual perceptron models into separate pipelines to be run in parallel and leverages GPU and CPU depending on what is available. Figure 2 below shows a pipeline for object detection using MediaPipe.

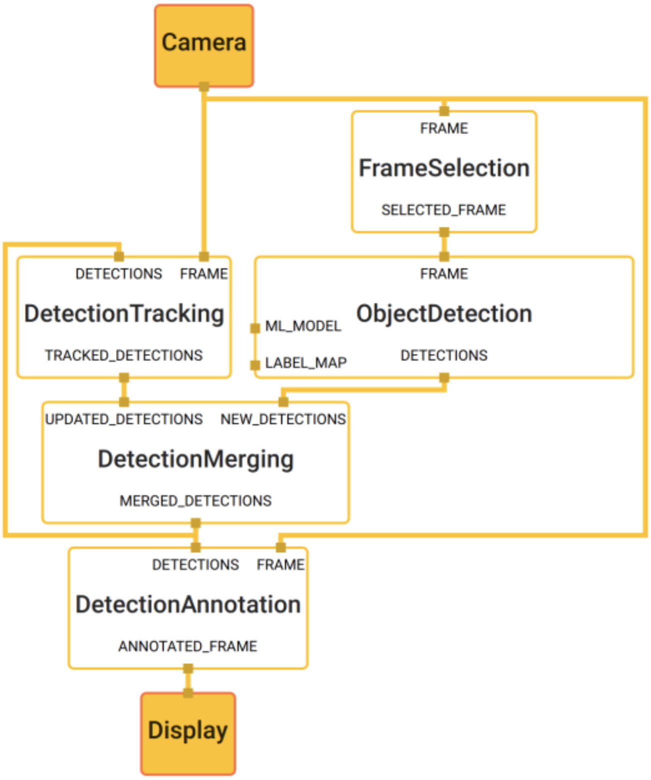


Fig. 2. Object detection using MediaPipe

An article by Coronado et al. performed a comparison between using MediaPipe and OpenPose for hand-landmark recognition and detection [12]. They reported MediaPipe to have a reduction in recognition by about 15% which was to be expected. However, MediaPipe had a latency of only 0.025 (40fps) whereas OpenPose had a latency of approximately 0.2131 (approx. 4.7fps). This shows that MediaPipe was around 8.5 times faster than OpenPose.

A recent article by Kavana & Suma published in 2022 used Google's MediaPipe hand tracking library with a Support Vector Machine to classify sign language gestures [13]. This shows that it is extremely possible to use Google's MediaPipe library along with some form of gesture recognition to control an interface.

With research into how to control mouse events and interface with the computer, Park used joint angle thresholding to recognise events such as left clicking, right clicking, and scrolling [14]. Park used a convex hull algorithm to detect the fingertips. Once the positions of the fingertips were obtained, mouse events were defined by the movements of the fingers in relation to the palm. This approach is much simpler than training a classification model for hand gestures. Difficulties in Park's approach include the jitter of the fingers and the segmentation of the hand due to changes in lighting. Another approach to recognise gestures from the positions of the fingers was done by Argyros and Lourakis. They recognised events such as left and right clicking by comparing the number of extended fingers [15]. This method was simple and allowed for accurate movement of the mouse as shown by the image drawn using their program in MS Paint in Figure 3. Argyros and Lourakis also attempted to reduce the movement of the cursor due to the jitter of the hand by controlling the mouse with the centroid of the hand.

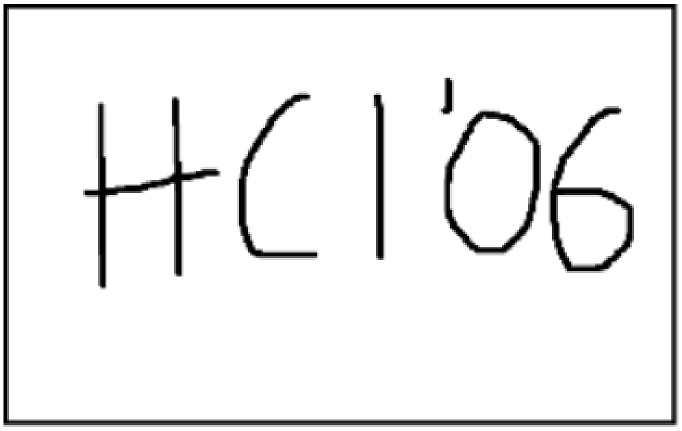


Fig. 3. Image drawn using Argyros and Lourakis' 2D mouse controller

There is still room for a more intuitive way to interact with a system using just hand gestures and common hardware. Aiming to reduce jitter and have a low memory usage real-time application are takeaways from the background research.

III. Proposed Method

The proposed method uses Google's MediaPipe Hands library with OpenCV to detect, track, and generate landmarks on a hand from the computer's video feed. The hand landmarks are then interpreted to allowed for the user to control mouse events and interact with the computer. This method aims to be very accessible, improve on prior limitations, and would only require a standard-resolution webcam and simple hardware.

A. Hand Landmark Detection

The landmark detection is performed using Google's MediaPipe Hands library. MediaPipe Hands generates keypoint localizations of 21 hand-knuckle coordinates through two separate machine learning models and they are described below [16] [17].

1. The first machine learning model in the pipeline is a palm detection model. It operates on the full image and returns an oriented hand bounding box. This first step is crucial to speeding up the pipeline as providing the cropped hand image to the next model reduces the need for data augmentation allowing the network to focus on landmark prediction accuracy. To initially detect the hand the pipeline uses a single-shot detector model that is optimized for real-time on as little hardware as a mobile device.

2. The next machine learning model is passed the cropped hand bounding box achieved from the previous model. This model uses regression to directly predict 21 2.5D landmark co-ordinates inside the detected hand regions. Each hand detected generates 21 x and y values between 0 and 1, and relative depth co-ordinates, as well as the ”handedness” (left hand or right hand).

The landmark detection predicts 21 landmark points on the hand in 2.5D. This is because, without a depth camera, the model estimates the relative depth with respect to the wrist point making each point 2.5D. Figure 4 and Figure 5 below show the output of the hand landmarks.

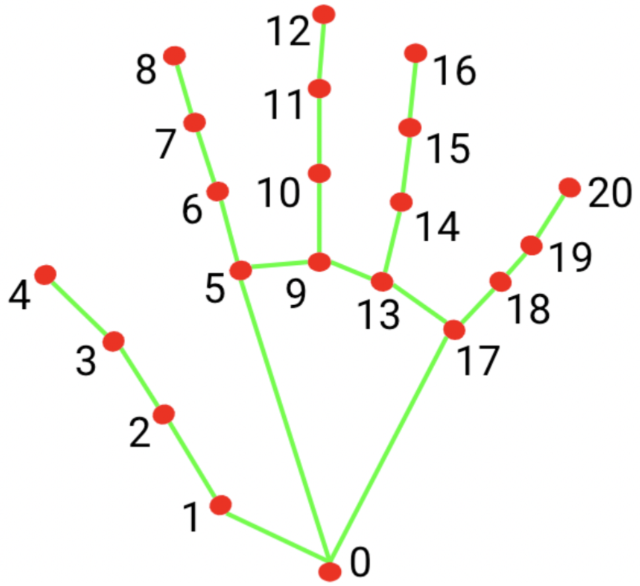
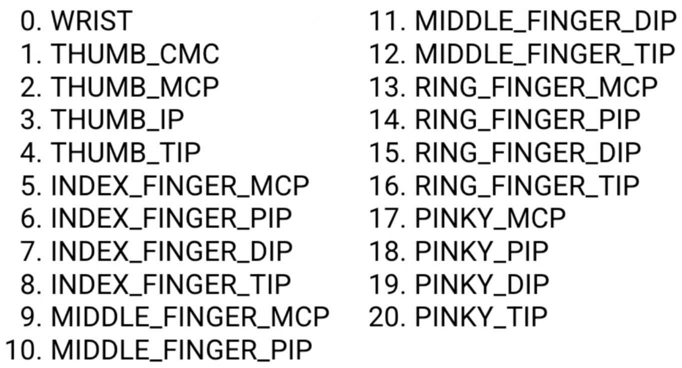


Fig. 4. Hand landmarks returned from MediaPipe Hands [16]



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. 5. Key for each hand landmark returned from MediaPipe Hands [16]

B. Cursor Control

The hand landmarks retrieved in the previous step for the user’s right hand are then passed on to allow the user to control the mouse. To reduce the jitter that was described as being a limiting factor in the papers published by Park, and Argyros and Lourakis, the proposed method will control the mouse by using the average of the hand landmarks to reduce the jitter [14]. To further reduce jitter, the proposed method uses a one Euro filter. The one Euro filter has less lag compared to other filters and is comprised of a low pass filter with an adaptive cutoff frequency [18]. It was chosen as it proves to have less lag compared to other filters, such as the Kalman filter, and is recommended for use in interactive systems. The one Euro filter was used on the (x, y) values returned from the average of the hand landmarks. To do this, the algorithm was altered slightly to work on a 2x1 matrix. The original algorithm found that a linear relationship between cutoff frequency the absolute speed worked well; however, to adapt the filter to work on a 2x1 matrix, instead of using the absolute value, the L2 norm of the matrix is used. This is shown in equation 1 below. The rest of the algorithm is left unchanged and all equations are performed instead with matrix operations.

$$\sqrt{x^2 + y^2} \tag{1}$$

On top of the one Euro filter, a threshold was used to reduce jitter even further. By making sure that the next mouse position was a given distance away from the previous, small jitter in the hand movements was removed altogether.

One limitation of the mediapipe landmarks was found that when the hand is only partially in the frame it has a hard time detecting it. This makes it difficult for the user to get to the edges of the screen. The hand co-ordinates returned from MediaPipe were clipped to be between 0.2 and 0.8 and then mapped to fill the screen width and height. This made it much easier to access the edges of the screen as the whole hand was still in the frame. It also had enough control to allow users to click small buttons.

C. Gestures

The proposed solution allows for control of 4 common actions on a mouse. As the mouse is solely controlled using the position of the right hand, the left hand is used to create the gestures. The defined gestures for each action are defined below.

1) Left Click: To allow users to left click, the user needs to place their forefinger and thumb together. The euclidean distance of the pointer and thumb points from the hand landmarks are checked and if they are within a threshold, the application will left click. If the user keeps their forefinger and thumb together, the application will hold down left click until they are released.

2) Right Click: Similarly to left clicking, the user simply needs to place their middle finger and thumb together. The distance between the middle finger and thumb is checked and if it is within a threshold, the application will right click.

3) Double Left Click: Much like left and right clicking, to double click the user needs to place their thumb and pinky finger together. Again, the euclidean distance is used with a threshold to determine if the user wants to perform the double left click.

4) Scrolling: To perform scrolling, the user must place all their fingers and thumb together. Again, euclidean distance between the finger points with a threshold is used to determine if the user is performing this gesture. While the user is performing the scrolling action, they can move their hand vertically up or down to scroll up or down by the distance that the landmarks moved.

Figure 6 below shows the gestures for each of the defined actions. (1) shows a left click, (2) shows a right click, (3) shows a double left click, and (4) shows the scrolling action. Note that in order to scroll the user needs to perform the action as shown in (4) and move their hand up or down.

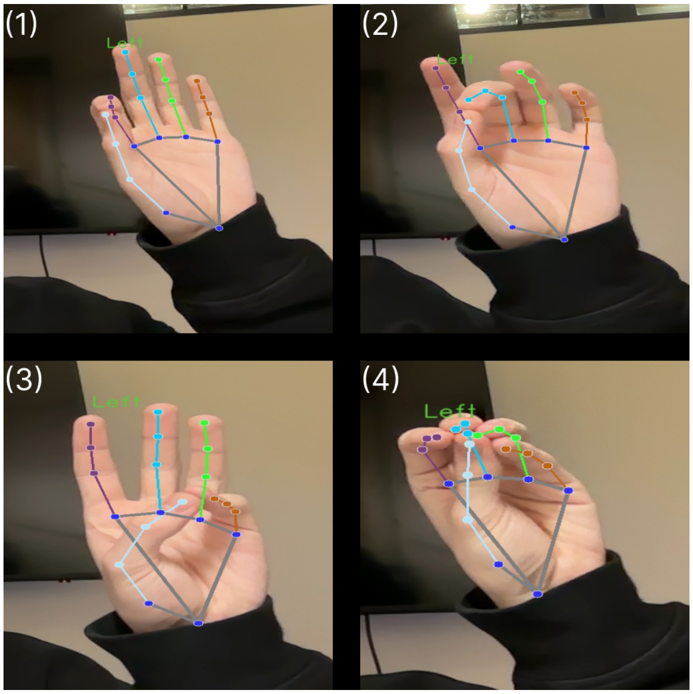


Fig. 6. The defined gestures for each action.

IV. Results

The hardware used in this report is defined in Table I below. It is assumed that the results obtained by the methods described in this paper are on the hardware defined below unless specified otherwise.

TABLE I
Hardware and Software Specifications

Operating System	Mac OS Sonoma
CPU	Apple M3 Pro 12-core CPU
GPU	Apple M3 Pro 18-core GPU
Device	2023 Macbook Pro
Camera	1080p 30FPS FaceTime HD
Programming Language	Python 3.10.12
IDE	JetBrains PyCharm

A. Hand Tracking

The use of MediaPipe hands for detecting the landmarks allowed quick development and a high average precision of 95.7% [17]. Previous work by Hung-Yuan Chung, Yao-Liang Chung, and Wei-Feng Tsai achieved similar results with an accuracy of 95.6% using a custom convolutional neural network [19]. The out of the box solution of MediaPipe hands offered state of the art accuracy as well as required less computing power compared to other solutions.

B. Gesture Recognition

To accurately assess the performance of the gesture recognition, a small study was done with 5 participants. Each participant performed each of the gestures 30 seperate times, and the number of gestures recognised by the system was recorded. This number could then be divided by the total number of gestures completed, 30, to get a percentage value for the accuracy of the system to detect specific gestures. The results of this small study are described in table II below.

TABLE II
Accuracy of each gesture

Gesture	Accuracy
Left Click	98.0%
Right Click	92.6%
Double Left Click	94.0%
Scrolling	84.0%

This table clearly shows an average accuracy of 92.15%. A similar study by Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen achieved and average recognition rate of 95.96% by using a custom convolutional neural network [20]. The use of a CNN meant that a higher accuracy could be acheived. However, this required a total of 800 images to be annotated which can be very time consuming. When using a CNN it also means that adding new gestures is difficult, it would require retraining of the model. By using the approach outlined in this paper, adding a new gesture is simple as it only requires defining the positions of the landmarks in relation to each other to recognise a gesture. In the study by Lin *et al.* the gestures were also with the hand palm side down with a camera above the hand making this application more difficult to use in the scope of cursor control.

To aid in the quantitative analysis of the gesture recognition, a qualitative study was also performed with the same group of participants. In this study, participants had to simply open an application from the task bar of a MacBook (as defined in table I), perform a scrolling action on the opened application, and close the application. This required accurate cursor control and proved to be a good robustness test of the system. All participants were able to complete the tasks within a reasonable time frame. The use of the one Euro filter meant that the jitter was mostly removed from the cursor control and allowed the participants accurate control of the mouse. There were difficulties in performing the scrolling as the scroll speed was originally too slow but was increased and got positive feedback. Another difficulty was seen when participants would move their hand out of the frame of the camera to use the keyboard or similar. In this instance, the program seemed to recognise hands that weren't there and the mouse had some erratic behaviour. This was an inconsistent phenomenon but did reduce the usability of the application.

C. Performance

To assess the performance of the application, the python tracemalloc library was used [21]. By starting the tracemalloc at the start of the program before any variables are initialised, and ending it when the program exits, the peak memory usage can be retrieved. The peak memory usage for the program was determined to be 18MB $(18901906/(1024 \times 1024))$. This includes all packages used in the program. This number is low which is what is expected as MediaPipe Hands was written to work on small hardware such as mobile phones. The low memory usage of this program means that it works well on low quality hardware, making it more readily available for the average consumer.

V. Conclusion

This paper presents a real-time method to interact with a computer in a more intuitive way using a standard web camera and hand gestures. The proposed method uses MediaPipe Hands for efficient and accurate hand landmark detection with an average precision of 95.7%. Previous work achieved a similar accuracy [19]. The system also boasts a high average accuracy for gesture recognition of 92.15% across various gestures. Previous studies achieved slightly higher results with the use of a CNN; however, this approach allows easy creation of new gestures, and doesn't require annotating training and testing data [20]. A similar study by Park, and Argyros and Lourakis found that jitter was a main limitation [14]. This paper aimed to improve on this by using an adaption of a one Euro filter to a 2 dimensional input by using L2 norm. In a simple participant study, the use of a one Euro filter and thresholding for movement effectively reduced jitter and allowed for accurate control of the mouse. Additionally, the use of MediaPipe along with custom defined gestures allowed for the program to be run with a peak memory usage of 18MB.

Some limitations of this study include limited number of usable gestures as the hand needs to be upright and in the view of the camera to work properly with MediaPipe. Another limitation is when the hands leave the frame the application can act erratically. The thresholds for gesture recognition are also defined statically in the code meaning that the recognition works best at specific distances for specific people.

Overall, this method offers a intuitive way of controlling the mouse on limited hardware.

A. Future Research

As the thresholds for gesture recognition are currently defined statically, improving on this may improve the general usability of the application. Calibration could be added to the application so the user needs to perform the given gestures and the system could automatically calculate accurate thresholds for each gesture. Improvements on the gesture recognition by a simple Support Vector Machine could allow for more natural and accurate gestures. More gestures could be defined and aren't limited to the position of the fingers relative to one another. Multiple camera angles could be tested to allow for more comfort of the user to interact with the system without having to hold their hands in-front of the screen, although this makes it less accessible.

References

[1] V. A. Prisacariu and I. Reid, "3d hand tracking for human computer interaction," *Image and Vision Computing*, vol. 30, no. 3, pp. 236–250, 2012.

[2] Y. Zhu, G. Xu, and D. J. Kriegman, "A real-time approach to the spotting, representation, and recognition of hand gestures for human–computer interaction," *Computer Vision and Image Understanding*, vol. 85, no. 3, pp. 189–208, 2002.

[3] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International journal of computer vision*, vol. 46, pp. 81–96, 2002.

[4] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM transactions on graphics (TOG)*, vol. 28, no. 3, pp. 1–8, 2009.

[5] L. Lamberti and F. Camastra, "Real-time hand gesture recognition using a color glove," in *Image Analysis and Processing–ICIAP 2011: 16th International Conference, Ravenna, Italy, September 14-16, 2011, Proceedings, Part I 16*. Springer, 2011, pp. 365–373.

[6] S. Sridhar, A. Oulasvirta, and C. Theobalt, "Interactive markerless articulated hand motion tracking using rgb and depth data," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2456–2463.

[7] I. Oikonomidis, N. Kyriazis, A. A. Argyros *et al.*, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[8] T. Sharp *et al.*, "Accurate, robust, and flexible real-time hand tracking, chi'15, apr. 18-23, 2015, seoul, republic of korea," ACM 978-1-4503-3145-6/15/04, Tech. Rep., 2004.

[9] D. Groos, H. Ramampiaro, and E. A. Ihlen, "Efficientpose: Scalable single-person pose estimation," *Applied intelligence*, vol. 51, no. 4, pp. 2518–2533, 2021.

[10] D. Osokin, "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. arxiv 2018," *arXiv preprint arXiv:1811.12004*, vol. 12004.

[11] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee *et al.*, "Mediapipe: A framework for perceiving and processing reality," in *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, vol. 2019, 2019.

[12] E. Coronado, K. Fukuda, I. G. Ramirez-Alpizar, N. Yamanobe, G. Venture, and K. Harada, "Assembly action understanding from fine-grained hand motions, a multi-camera and deep learning approach," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2628–2634.

[13] K. Kavana and N. Suma, "Recognization of hand gestures using mediapipe hands," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 06, 2022.

[14] P. Hojoon, "A method for controlling mouse movement using real-time camera," *Brown Computer Science*, 2010.

[15] A. A. Argyros and M. I. Lourakis, "Vision-based interpretation of hand gestures for remote control of a computer mouse," in *Computer Vision in Human-Computer Interaction: ECCV 2006 Workshop on HCI, Graz, Austria, May 13, 2006. Proceedings 9*. Springer, 2006, pp. 40–51.

[16] Google, "Google mediapipe hand landmark," https://developers.google.com/mediapipe/solutions/vision/hand_landmarker, accessed on 11-05-2024.

[17] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[18] N. R. Casiez, Géry and D. Vogel, "1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems," in *SIGCHI Conference on Human Factors in Computing Systems*, vol. 2012, 2012.

[19] Y.-L. C. Chung, Hung-Yuan and W.-F. Tsai, "An efficient hand gesture recognition system based on deep cnn." in *IEEE International Conference on Industrial Technology (ICIT)*, vol. 2019, 2019.

[20] M.-H. H. Lin, Hsien-I. and W.-K. Chen, “Human hand gesture recognition using a convolution neural network,” in *IEEE International Conference on Automation Science and Engineering (CASE)*, vol. 2014, 2014.

[21] P. S. Foundation, “tracemalloc — trace memory allocations,” 2023, accessed: 2024-06-26. [Online]. Available: https://docs.python.org/3/library/tracemalloc.html