2022 3rd Int. Conf. on Innovations in Science, Engineering and Technology (ICISET)
26-27 February 2022, Chittagong, Bangladesh

Real Time Hand Gesture Based User Friendly Human Computer Interaction System

Koushik Roy[1] and Md. Akiful Hoque Akif[2]
Department of Electrical, Electronic and Communication Engineering (EECE)
Military Institute of Science & Technology (MIST), Dhaka, Bangladesh
rkoushikroy2@gmail.com[1]; mohammadaxif5717@gmail.com[2];

***Abstract*—The demand for interface technologies such as hand gesture detection is growing as virtual reality technology advances. In our study, we employed the state of art hand-tracking technology to construct an accurate and robust human-computer interaction(HCI) system. Our proposed method is cross-platform, lightweight, and effectively utilizes the resources from both CPU and GPU to achieve low latency performance. The MediaPipe framework, which yields a mean precision of 95.7 percent in palm identification, is used in this research to construct a real-time human-computer interaction interface that works properly regardless of difficult conditions such as different skin color, shape of hand, lighting condition, background, etc. Additionally, the speed of the proposed system is close to 40 fps acquired in a computer with weak hardware, which is sufficient for real-time HCI applications.**

***Index Terms*—Human-Computer Interaction, MediaPipe, Hand Gesture, Hand Landmarks, Real-Time Execution**

I. Introduction

Human-computer interaction has grown dramatically, and the subject is always evolving, with new methodologies and strategies being developed. Hand gesture recognition is one of the most sophisticated fields in which computer vision and artificial intelligence have aided not only in improving communication with deaf individuals but also in assisting gesture-based signaling systems [1], [2]. Sign language recognition [3], [4], specific signal language recognition for sports [5], human action identification [6], stance and posture detection [7], [8], physical activity monitoring [9], and controlling smart home or assisted living applications using hand gesture recognition [10] are all sub-domains of hand gesture recognition.

With the onset of Covid-19, hygiene has become a vital element in our life. Significant emphasis has been put on maintaining social distance and reduce the touching of various objects. In such a situation, technology can be introduced to facilitate various services and the functioning of various machines without any physical interaction with the surface of the machine. The hand gesture technique is the perfect solution in this regard where a person can receive many services without touching anything.

To collect hand gesture data, several previous studies required users to wear a data glove. However, the data glove's sophisticated sensors are pricey, limiting its use in real life. The authors employed a Kinect sensor based TOF camera to capture the depth of the surroundings in their work [11], as well as a specific tape worn across the wrist to find the hand area. Our method merely requires the use of a regular camera to collect the observed information of the hand gesture and does not require the use of a specific tape to find hand areas.

Hands by MediaPipe [12] has a solution for high-resolution finger and hand recognition. Machine learning is used to deduce 21 3D landmarks of a hand from a single shot. By opening up these hand perception skills to the rest of the research and development community, new applications and research pathways will arise, inspiring new applications and study. Their solution delivers real-time performance on a cell phone, and even scales to several hands, whereas existing state-of-the-art systems rely mostly on high-equipped desktop environments for inference. They personally tagged 30 thousand real and original images with 21 three-dimensional coordinates. They also create a high-quality synthetic hand model and map it to the corresponding 3D coordinates to better cover the available hand positions and provide further supervision on the nature of hand geometry.

Our research is unique in the sense that we are avoiding the use of any supporting material i.e. hand gloves, sensors, or other equipment. Moreover, a normal webcam is going to be used in our proposed method. We are not including any complex gestures of various sorts which convey different commands. Rather, we are mapping the common gestures of our day-to-day life to reduce variation and complexity. Thus the project will be simpler and user-friendly.

The following sections of this paper are organized as follows. In Section II, we presented the relevant findings of related literature. Then the methodology of this research work has been discussed in Section III. The experimental results were described in Section IV. Finally, In section V, the conclusion along with future work has been presented.

II. Related works

Agrawal et al. [13] proposed a method that can successfully replace the usage of a keyboard or mouse to interface with a computer. The approach employs a Senz3D commercial depth and RGB camera, which is inexpensive in comparison to other depth cameras. The suggested technique works by evaluating 3D data in real-time and classifying the number of convexity defects into gesture classes using a set of classification rules. This produces real-time results and eliminates the need for any

978-1-6654-8397-1/22/$31.00

training data. The proposed approach provides commendable results while consuming very little processor power.

Xu et al. [14] created a real-time gesture-based HCI system that recognizes motions with just a single monocular camera, and they apply it to the HRI application. To learn features and recognize gestures, the created system uses a CNN classifier. They also demonstrated the use of the Kalman filter to smooth the motion of the hand-controlled mouse cursor. Only static gestures are supported by their designed system.

An efficient and successful approach is presented by Chen et al. [15] for hand gesture recognition. The background subtraction method is used to detect the hand region. The fingers and palm are then divided to identify the fingers. Following the recognition of the fingers, the hand gesture can be categorized using a simple rule classifier. The proposed approach has a 96.6 percent average accuracy.

Specifically for mobile applications, Yin et al. [16] suggested a lightweight method for estimating object and hand pose estimation. They demonstrated how their methodology delivers real-time performance, comparable accuracy and 81 percent reduction of the model size compared to state-of-the-art approaches, justifying the model's suitability for deployment on mobile systems.

The user can interact with a computer without using a marker by moving their hands proposed in this study by Patidar et al. [17]. To follow the movement of the hand, employ hand segmentation and a color model. The segmentation and tracking procedure is carried out by employing a color model to extract or recognize hand motions in order to develop better, faster, and more accurate real-time systems.

Sziladi et al. [18] introduces mouse cursor control based on hand movements, as well as the evaluation of mouse cursor movement, as implemented by The Leap Motion device. During the mouse cursor movement analysis, the actual movements made by a traditional mouse cursor and the detection of hand gestures are compared with the participation of test subjects. Differences in the course of controlling done by a conventional mouse and hand gesture detection were presented based on the findings of the mouse cursor movement analysis.

A lightweight hand gesture recognition model was proposed by Mujahid et al. [19] based on the YOLOv3 and DarkNet-53 deep learning models. With an accuracy of 97.68 percent, the created hand gesture recognition system recognises both real-time objects and gestures from video frames. They also compared the YOLOv3 model's performance and execution with various methods, and found that their proposed method produced better results by extracting features from the hand and recognizing hand gestures, with accuracy, precision, recall, and F-1 scores of 97.68, 94.88, 98.66, and 96.70 percent, respectively.

Rautaray et al. [20] creates a hand gesture detection system for browsing photos in an image browser, and it offers a useful method for creating a user-friendly interface between humans and computers through hand gestures. The study effort proposed here could be very useful in a wide range of applications where human-computer interaction is a frequent need.

The gesture detection system proposed by Haria et al. [21] was to create gestures that covered practically all aspects of HCI, such as system functionality, application start, and opening popular websites. We included a total of seven motions in their gesture detection system, six of which are static gestures and one of which is dynamic. The average recognition accuracy for static gestures is 94%. Their method transforms the detected gesture into actions like opening websites and running VLC Player and PowerPoint. In a presentation, the dynamic gesture is used to slide between the slides.

III. Methodology

The developed gesture-based HCI system is robust and intuitive to the new users. Every image frame taken by the camera goes through the predefined steps to provide us with the necessary output to run the system. Firstly, the captured image from webcam goes through a hand landmark detector, and 21 landmark points are detected. Secondly, the coordinates given by the landmark detector are used as input for the custom finger counter method. The output of this method will, in turn, be used for the appropriate mode selection such as single click, double click, and click & drag. Finally, the location of the index fingertip needs to be extracted in order to locate it relative to the fixed position of the camera feed. Based on the location of the index fingertip the mouse pointer will be controlled. The speed and sensitivity can be adjusted by the user. The overall block diagram of the system is shown in Fig. 1. The human computer interaction interface has been represented visually in Fig. 2. Details of the execution stages are described in the subsections below.

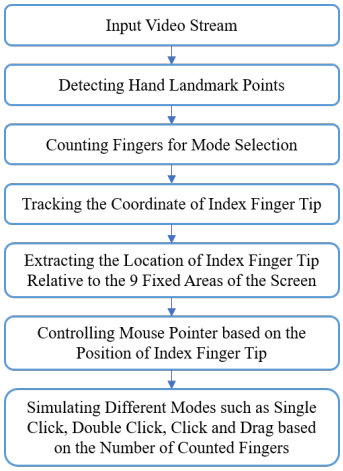


Fig. 1. Overall System Execution Stages Presented in a Block Diagram

A. 'MediaPipe Hands' Overview

Recognition of pattern and motion can assist enhance user engagement across a wide range of technological disciplines and platforms. It can, for example, establish the groundwork for reading sign language and directing hand movements, as

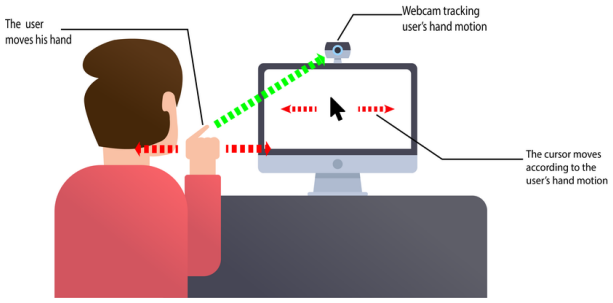


Fig. 2. Visualization of the Hand Gesture Controlled Human Computer Interaction System

well as allowing augmented reality to overlay digital content and information on top of the real world. Because hands frequently constrict themselves or one other (for example, finger/palm partial occlusion and handshaking), and because they lack high dynamic range patterns, robust real-time hand perception is a tough computer vision task.

'Hands' by MediaPipe is a hand and finger tracking system with excellent resolution. From a single image, machine learning (ML) was utilized to derive 21 3D landmarks of a hand. While previous state-of-the-art systems rely heavily on sophisticated desktop environments for inference, this method delivers real-time performance on a mobile phone and is even scalable to many hands [12].

B. Machine Learning Pipeline

MediaPipe Hands use a deep learning workflow that consists of a number of interconnected models: A palm detection model that works on the entire picture returns an oriented hand bounding box. A hand landmark generator that generates high-fidelity 3D hand keypoints from the palm detector-cropped image area.

Providing a correctly cropped hand image to the hand landmark model reduces the need for data augmentation and allows the network to focus on accuracy in coordinate prediction. Furthermore, crops may be created in the pipeline based on hand landmarks recognized in the previous frame, with palm detection only being employed to relocalize the hand when the landmark model can no longer identify its presence.

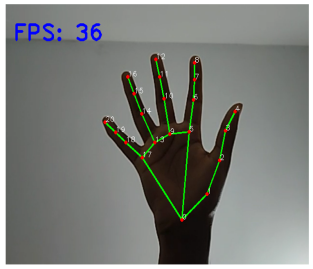
The pipeline is constructed as a MediaPipe graph that leverages the hand landmark module's hand landmark tracking subgraph and renders it with a specialized hand renderer subgraph. Internally, the tracking subgraph of hand landmark utilizes the same module's hand landmark subgraph and the palm detection module's palm detection subgraph [12].

C. Models

1) Palm Detection Model: A single-shot detector model was created to identify initial hand positions, and it was optimized for mobile real-time use. Hand detection is a difficult problem since the model must function with a broad range of hand shapes and sizes and identify occluded hands. Whereas faces contain strong contrast patterns, such as around the eyes and mouth, hands lack similar traits, making it more difficult to consistently recognize them based on their visual features alone. Providing extra contexts, such as arm, body, or human characteristics instead helps with precise hand localization.

'MediaPipe Hands' employs a variety of techniques to overcome the aforementioned issues. First, a palm detector was utilized instead of a hand detector since estimating bounding boxes of rigid objects such as palms and fists is considerably easier than detecting hands with articulated fingers. Furthermore, because palms usually are smaller objects, the non-maximum suppression approach works well in instances where two hands are involved, such as handshakes. Square border boxes can also be used to depict palms. Finally, because of the high scale variance, the focus loss during training was reduced in order to accommodate a large number of anchors. The various approaches resulted in an average accuracy of 95.7 percent in palm detection. With no decoder and a normal cross-entropy loss, the baseline is just 86.22 percent.

2) Hand Landmark Model: Following palm detection, the hand landmark model utilizes regression to achieve accurate keypoint localization of the hand-knuckle coordinates inside the identified hand areas, which is known as a direct coordinate prediction. The model produces a consistent inner hand positioning approximation also with partially visible hands or self-occlusions. Around 30 thousand real world pictures were manually labeled to provide ground truth data. A high-quality artificial hand model was projected over a variety of surroundings and converted to 3D coordinates. The 21 hand landmark points' positions are shown in Fig. 3.



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. 3. Hand landmark points with their corresponding positions overlayed on the hand image

D. Hand Landmark Detection Module

We created our custom hand landmark detector module using the MediaPipe package using Python. The detection confidence and tracking confidence of our landmark detector module were both kept at 50% as our application requires real-time tracking and detection. A higher confidence value will make the detection more accurate by reducing the tracking capability, thus potentially hamper usability. We used OpenCV [22] for real-time video capture and presentation after processing. OpenCV is a major open-source framework for computer vision, deep learning, and image analysis that is now used in real-time processes [22].

The coordinate points of the detected hand landmark points have been shown in Fig. 4a and the index fingertip position

has been highlighted in Fig. 4b. Additionally, we calculated real-time Frames Per Second(fps) and displayed it in the demo output. We achieved an average fps of 32, which is plenty for any real-time usage requirements.

(a) Landmark Detector Output (b) Index Fingertip is Located and Marked

Fig. 4. Hand landmark detector module output

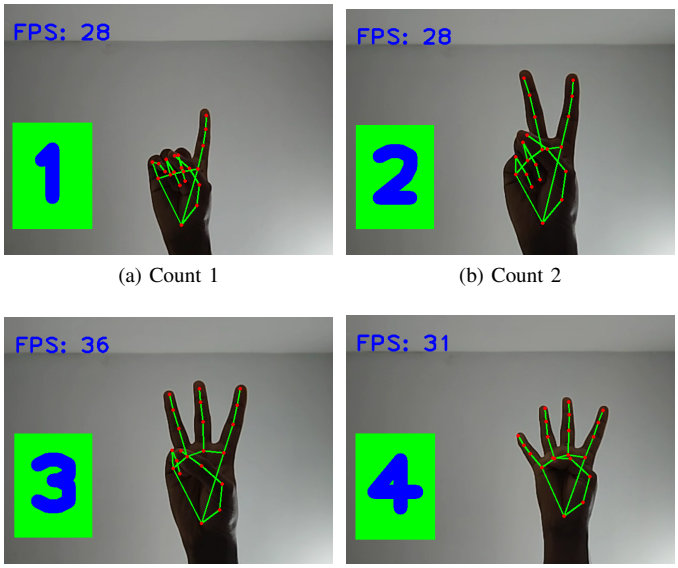
E. Finger Counter Module

We created our custom finger counter module that can count the number the fingers up in any instance. This is necessary because we want to use the number of fingers as an effective gesture to simulate different useful actions. From Fig. 3, we can see that the landmark points of the tip of the fingers are 4, 8, 12, 16 & 20. In case of all the fingers except for the thumb, we will compare the Y-coordinate of the corresponding fingertip with the Y-coordinate of the knuckle which sits 2 points below the fingertip. If the value of the fingertip is greater than the value of the knuckle, then the finger is up and vice versa. For the thumb, we compare the X-coordinate of the fingertip with the X-coordinate of the knuckle which sits 1 point below the tip to make the decision whether the thumb is open or closed. The output provided by the finger counter module is shown in Fig. 5.

F. Human-Computer Interaction Mechanism

Our HCI system works by providing a mechanism that can control mouse movements using hand gestures. We used hand tracking and landmark detector modules to correctly figure out the location of each fingertip along with the coordinate value for all other landmark points. Our mouse movement control works based on the position of the index fingertip in the designated positions on the image frame captured by the camera. We divided the image frame into 9 distinct areas each of which represents a separate action to be carried out by the mouse cursor, represented in Fig. 6.

The position of the index finger in these separate areas will in turn drive the mouse cursor to move towards that direction. We used the mouse package in Python to move the mouse cursor and simulate clicking. A sensitivity multiplier(K) has been multiplied with the incremental value of the mouse cursor coordinates and the value will be user-configurable so that the user can adjust the system's speed and sensitivity according to their own likings. The value of the mouse cursor coordinate will shift by the increment of 10 pixels multiplied with K



(a) Count 1 (b) Count 2

(c) Count 3 (d) Count 4

Fig. 5. Finger Counter Module Output

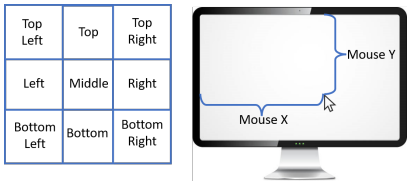


Fig. 6. Different Areas of the Screen Representing Different Actions

and the action based on the position of index finger has been presented in Table I.

The custom finger counter module will aid in carrying out the clicking actions and click and drag functionalities. The finger counter is very accurate and reliable. Moreover, it does not add any additional computational complexity to the existing landmark detector module. Because it works based on the data given by the landmark module. So, we are able to perform these extra actions without losing any performance, contrary to the methods that employ separate gesture recognition models for detecting distinct gestures. The actions and their respective finger counts have been described

TABLE I
ACTION BASED ON THE POSITION OF THE INDEX FINGER IN THE IMAGE FRAME

Position of Index Finger	Mouse X	Mouse Y
Top Left	$-10 * K$	$-10 * K$
Top	$0 * K$	$-10 * K$
Top Right	$+10 * K$	$-10 * K$
Left	$-10 * K$	$0 * K$
Middle	$0 * K$	$0 * K$
Right	$+10 * K$	$0 * K$
Bottom Left	$-10 * K$	$+10 * K$
Bottom	$0 * K$	$+10 * K$
Bottom Right	$+10 * K$	$+10 * K$

TABLE II
REQUIRED ACTION BASED ON FINGER COUNT

Gesture	Action
Two Fingers Count	Single Left Click
Three Fingers Count	Double Left Click
Four Fingers Count	Right Click
Five Fingers Count	Click and Drag

in Table II.

IV. EXPERIMENTAL RESULTS AND COMPARISON

We constructed the gesture-based HCI system and employed it in a windows machine. Since our underlying mechanism is cross-platform, it will work on other operating systems just as easily and reliably. A synthetic demonstration of our system's working procedure has been presented in Fig. 7. From the perspective of initial adopter of our system, this offers an intuitive approach of HCI mechanism that is easy to pick up for anyone who is using it for the first time.

The fps we get varies between 11 and 85. But the average fps we got in a recorded iteration of 811 was 32.05, which is sufficient for our use case. A line chart has been presented in Fig. 8, showing the time series representation of fps count over a certain number of instances. The necessary metrics of the fps data has been presented in Table III.

TABLE III
NECESSARY METRICS OF THE COLLECTED FPS DATA

Metrics	Value
Counted Instances	811
Average	32.05
Median	31.59
Mode	35.80
Min	11.66
Max	85.63

The hardware spec in which the training and evaluation has been done is shown below:

- GPU: 1xGeForce 930MX, Base Frequency 952 MHz, 384 CUDA cores, 24 TMUs, 16 ROPs and Memory Capacity 2GB
- CPU: Intel Core i5-8250U, Number of Cores 4, Number of Threads 8, Base Frequency 1.60 GHz, Max Turbo Frequency 3.40 GHz, 6 MB Smart Cache, Bus Speed 4 GT/s, TDP 15 W
- RAM: 8.0 GB Available
- Disk: 256 GB Available

Our hand gesture based HCI system is based on the state of art technologies and compare fairly well with the existing methods of similar application found from other research works. We can see that there are a variety of equipment used in these works, each with different methodology and system model. Our work varies from the other works in the method, number of gesture and the used equipment scenarios. The novelty of our approach lies in the simplicity in the usability standpoint along with the superior performance in a weak piece of hardware. The computational complexity has been minimized by the exclusion of cascaded heavy deep learning models. Instead we used simple and lightweight model which is accurate and reliable at the same time.

V. CONCLUSION AND FUTURE WORKS

Our motivation for making a touchless HCI system that will be as reliable as a touch-based or physical hardware-based HCI system was to provide a low-cost alternative for the devices that require a touchless input mechanism. The solution we provided will be particularly useful for the safe usages of the end devices that reside in public places such as transit stations. We believe that the widespread adoption of touchless HCI systems will help us to minimize the spread of diseases such as Covid-19. Our method of system modeling and interfacing relies on the core aim of making the system as lightweight as possible, also retaining the same or more additional features that a traditional heavy model can provide. Our system is fast, reliable and can easily be implemented in all kinds of operating system. All the data processing is done internally and used in real time. No video is recorded or saved from the webcam, thus proving Superior privacy for the application in sensitive places such as ATM booth or Hospitals. The expansion of this technology in the medical research, augmented reality and gaming industry might be a future study endeavor.

REFERENCES

[1] Y. Fang, K. Wang, J. Cheng, and H. Lu, "A real-time hand gesture recognition method," in *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 2007, pp. 995–998.

[2] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: a review of techniques," *journal of Imaging*, vol. 6, no. 8, p. 73, 2020.

[3] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, M. A. Bencherif, T. S. Alrayes, H. Mathkour, and M. A. Mekhtiche, "Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation," *IEEE Access*, vol. 8, pp. 192 527–192 542, 2020.

[4] A. Vaitkevičius, M. Taroza, T. Blažauskas, R. Damaševičius, R. Maskeliūnas, and M. Woźniak, "Recognition of american sign language gestures in a virtual reality using leap motion," *Applied Sciences*, vol. 9, no. 3, p. 445, 2019.

[5] J. Žemgulys, V. Raudonis, R. Maskeliūnas, and R. Damaševičius, "Recognition of basketball referee signals from real-time videos," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 3, pp. 979–991, 2020.

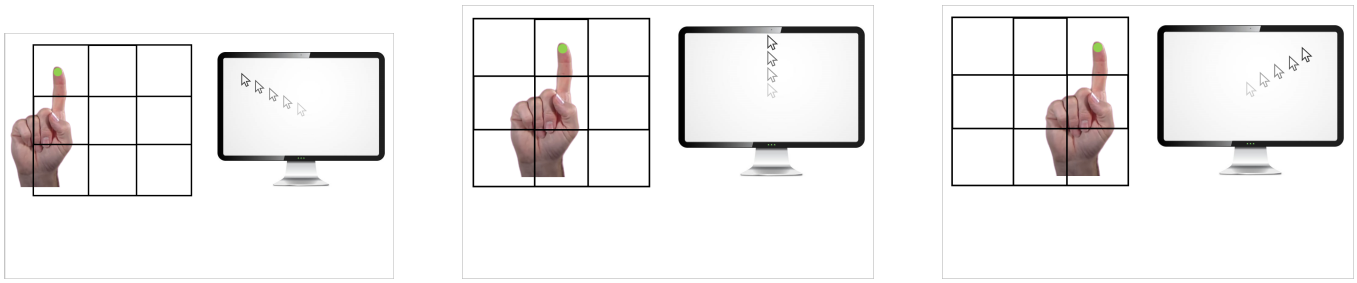
[6] F. Afza, M. A. Khan, M. Sharif, S. Kadry, G. Manogaran, T. Saba, I. Ashraf, and R. Damaševičius, "A framework of human action recognition using length control features fusion and weighted entropy-variances based feature selection," *Image and Vision Computing*, vol. 106, p. 104090, 2021.

[7] A. Nikolaidis and I. Pitas, "Facial feature extraction and pose determination," *Pattern Recognition*, vol. 33, no. 11, pp. 1783–1791, 2000.

[8] A. Kulikajevas, R. Maskeliunas, and R. Damaševičius, "Detection of sitting posture using hierarchical image composition and deep learning," *PeerJ computer science*, vol. 7, p. e442, 2021.

[9] K. Ryselis, T. Petkus, T. Blažauskas, R. Maskeliūnas, and R. Damaševičius, "Multiple kinect based system to monitor and analyze key performance indicators of physical training," *Human-Centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1–22, 2020.

[10] P. N. Huu, Q. T. Minh *et al.*, "An ann-based gesture recognition algorithm for smart-home applications," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 14, no. 5, pp. 1967–1983, 2020.



(a) Top Left (b) Top (c) Top Right

(d) Left (e) Middle (f) Right

(g) Bottom Left (h) Bottom (i) Bottom Right

Fig. 7. Synthetic Demonstration of our System's Working Procedure

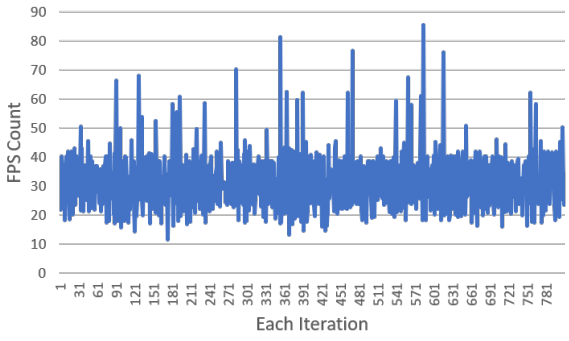


Fig. 8. FPS Count vs Number of Iteration

[11] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using kinect sensor," *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1110–1120, 2013.

[12] 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.

[13] R. Agrawal and N. Gupta, "Real time hand gesture recognition for human computer interaction," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. IEEE, 2016, pp. 470–475.

[14] P. Xu, "A real-time hand gesture recognition and human-computer interaction system," *arXiv preprint arXiv:1704.07296*, 2017.

[15] Z.-h. Chen, J.-T. Kim, J. Liang, J. Zhang, and Y.-B. Yuan, "Real-time hand gesture recognition using finger segmentation," *The Scientific World Journal*, vol. 2014, 2014.

[16] Y. Yin, C. McCarthy, and D. Rezazadegan, "Real-time 3d hand-object pose estimation for mobile devices," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3288–3292.

[17] S. Patidar and D. C. Satsangi, "Hand segmentation and tracking technique using color models," *International Journal of Software & Hardware Research in Engineering*, vol. 1, no. 2, pp. 18–22, 2013.

[18] G. Sziladi, T. Ujbanyi, J. Katona, and A. Kovari, "The analysis of hand gesture based cursor position control during solve an it related task," in *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE, 2017, pp. 000 413–000 418.

[19] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem, "Real-time hand gesture recognition based on deep learning yolov3 model," *Applied Sciences*, vol. 11, no. 9, p. 4164, 2021.

[20] S. S. Rautaray and A. Agrawal, "Real time multiple hand gesture recognition system for human computer interaction," *International Journal of Intelligent Systems and Applications*, vol. 4, no. 5, pp. 56–64, 2012.

[21] A. Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, "Hand gesture recognition for human computer interaction," *Procedia computer science*, vol. 115, pp. 367–374, 2017.

[22] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.