Vision-Based Hand Tracking System Development for Non-Face-to-Face Interaction

Isaiah Jassen Tupal and Melvin Cabatuan
Electronics and Computer Engineering Department
Gokongwei College of Engineering
De La Salle University
Manila, Philippines

2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM) | 978-1-6654-0167-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/HNICEM54116.2021.9731873

Abstract— **Human-computer interaction (HCI) focuses on the interaction between humans and computers and it exists ubiquitously in our daily lives, especially in post COVID era where non-face-to-face interaction is common. Since HCI usually uses a physical controller such as a mouse or a keyboard, it hinders National User Interface, giving a middle ground between the user and the computer. This paper presents a vision-based hand tracking system development for non-face-to-face interaction, which aims to improve HCI by being able to track the hand which will act as the pen and functioning as a reusable writing surface for creating texts, drawings, and such as well as removing or erasing using the user's hand as the pen, and utilizing Open Computer Vision Library (OpenCV) and Mediapipe. Using the computer's camera the hand will be tracked as the pen for creating basic drawings and handwriting. The vision-based board where the user can draw on and the pen or marker will be the user's hand. The results indicate that this system is accurate enough to be a feasible application for handwriting ad basic drawings.**

Keywords—OpenCV, computer vision, human-computer interaction, natural user interface

I. Introduction

This paper presents a machine vision-based smart electronic board software system development for future learning spaces. Furthermore, this paper covers the specifications, interface, features, and overall functionality of the program. Moreover, this is the first version of this application and is open for constant improvement and the addition of features.

Vision-based technology of hand gesture recognition is an important part of human-computer interaction (HCI). HCI is a multidisciplinary field of study focusing on the design of computer technology and, in particular, the interaction between humans and computers. Since then, HCI has grown to be broader, larger, and much more diverse than the computer itself. It no longer makes sense to regard HCI as a specialty of computer science and HCI expanded from its initial focus on individual and generic user behavior to include social and organizational computing, accessibility for the elderly, the cognitively and physically impaired, and for all people, and for the widest possible spectrum of human experiences and activities. It expanded from desktop office applications to include games, learning and education, commerce, health, and medical applications, emergency planning and response, and systems to support collaboration and community. It expanded from early graphical user interfaces to include myriad interaction techniques and devices, multi-modal interactions, tool support for model-based user interface specification, and a host of emerging ubiquitous, handheld, and context-aware interactions [1].

In this paper, the researchers present a vision-based smart hand tracking-based board system for future learning spaces utilizing Python and OpenCV for minimal hardware requirements. This program aims to provide basic functionalities of an electronic board system and promote HCI and NUI within the user and the computer.

II. Related Works

There are various techniques used for hand detection, examples include background subtraction, color detection, image segmentation, finger-tip detection, and so on. One hand detection technique is known as the Viola and Jones detector and scale-invariant feature transform based hand detection algorithm which provides the result with high accuracy but with sensitivity to the background [2]. On the other hand, image segmentation is another technique used for hand detection, which uses HSV color space model rather than RGB color space for determining the color of the human skin which provides better background separation and region boundary. However, this technique will encounter problems when the skin color is of the same color as the background [3]. Another common technique is known as learning-based recognition in the Adaptive Boosting algorithm that can integrate the information of the same category of objects. It trains the network by combining all weak classifiers into one strong classifier. The Adaptive Boosting learning algorithm selects the best weak classifier from a set of positive and negative image samples. This algorithm provides the result with better accuracy and fast speed but sometimes the training period is more to train the network [4]. The mean-shift algorithm and the cam-shift algorithm (which is considered an improvement over mean-shift) is another object tracking technology designed to find color densities in feature space (color space specifically). In this algorithm, the specific color (the chroma) matters in specific object tracking. The object is treated as one of the colors of the object that is desired to be tracked falls under its shade of color, so color intensity doesn't matter too much [5]. Another object-tracking algorithm, the homography algorithm, is utilizing a "generated matrix" which is a mapping of a given set of points in a provided image - to the corresponding set of points in another image (in this case corresponding it with the real-time "video"). In the algorithm's practical case in this context, key features of the sample image of the object are to be automatically set to as those points; and correspond them with the real-time video, tracking desired object [6].

.

Numerous existing algorithms are available for hand detection, another technique is finding convex hulls and there are multiple existing algorithms for finding convex hulls and most of the time, more than one is algorithm is used for finding convex hulls since tracking hands and their motion can be complicated and some gesture and its motion can be quite complicated [2]. The reason for these algorithms since recent trends of computer vision techniques are easy, natural, and less costly since most of the laptops have an integrated webcam along with it so it is an easily available device [7]. This is an important feature as gestures can contain more information than simple words and it reveals more information that sometimes cannot be found in speech. Hand gesture recognition has been widely used in human-computer interaction, like in virtual reality, robotics, and computer gaming, and others [8].

Recently, Google introduced a multimodal - audio, video, & time-series data - framework for constructing a machine learning pipeline [9]. Mediapipe can be used for hand tracking [10] and other vision-based applications like micro-expressions analysis [11] and sign language recognition [12]. In this paper, we will also explore the capabilities of Mediapipe in a vision-based hand tracking system for non-face-to-face interaction.

III. METHODOLOGY

A. Planning Stage

In designing vision-based hand tracking system development, the following has been considered during the planning stages, namely:

a) The program can track an object which will serve as the pen for the electronic board.
b) The program can draw by tracking the object in real-time.
c) The program can utilize an erase mode function to delete a portion or all the writings made by the user.
d) The program is limited to separating the color of the object from the background.
e) The program can export files to an image file or do a screen recording mode and export to a video file.

Furthermore, a user-case story was developed to help the designers in developing the software from the end-user perspective and help meet the requirements needed.

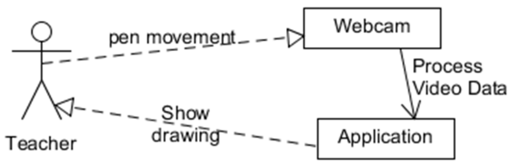


Figure 1.1 Art Teacher Case Story

As a creative preschool teacher, I want to be able to write on the screen when I move my pen.

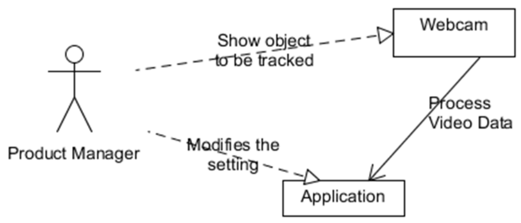


Figure 1.2 Manager Case Story

I have a bunch of markers in my house, so I want the app to fit my specific marker color. I also want it to track whatever I have at the moment so settings for which to track is what I need.

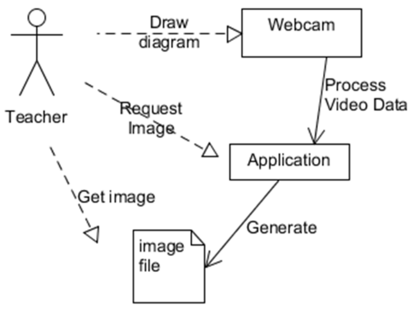


Figure 1.3 Science Teacher Case Story

I want my diagrams to be saved to an image file so I can share my diagrams with my students when I conduct an online lecture.

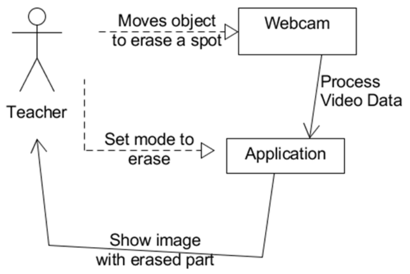


Figure 1.4 Teacher Case Story

I make a lot of mistakes, so I want it to have a feature that gives me the ability to erase sections of my diagrams.

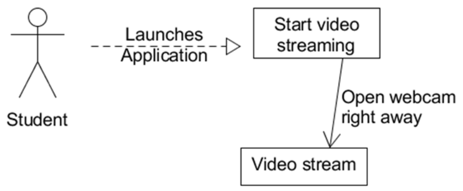


Figure 1.5 Student Case Story

I am always busy, so I want the program to go straight into the board when launched.

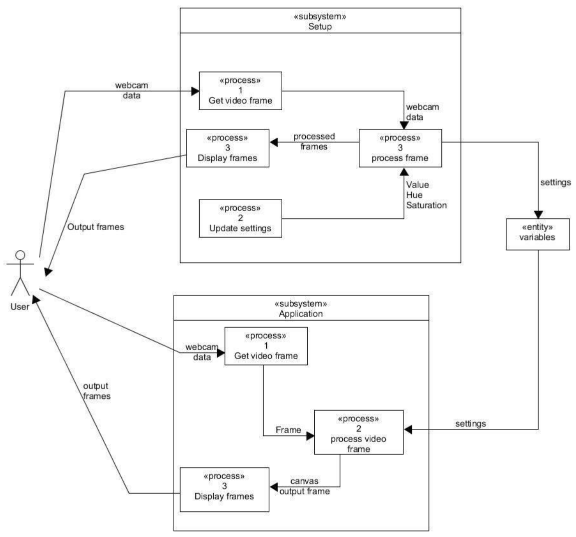


Fig. 2. Use-Case Diagram

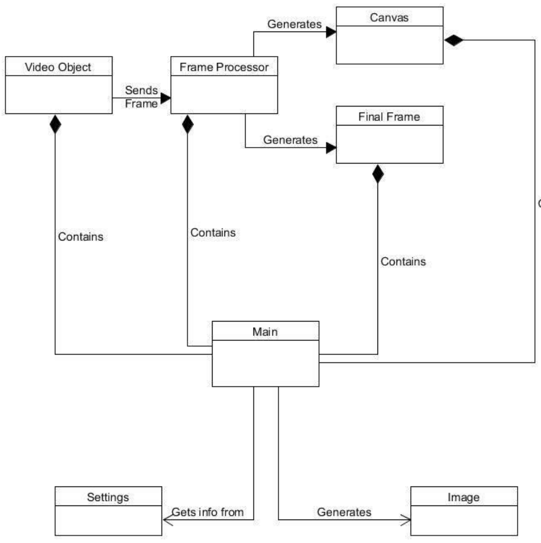


Fig. 3. Object Diagram

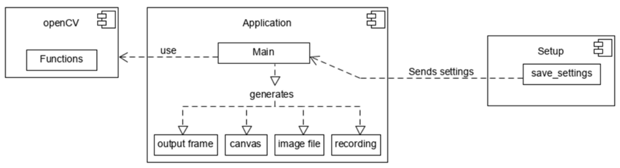
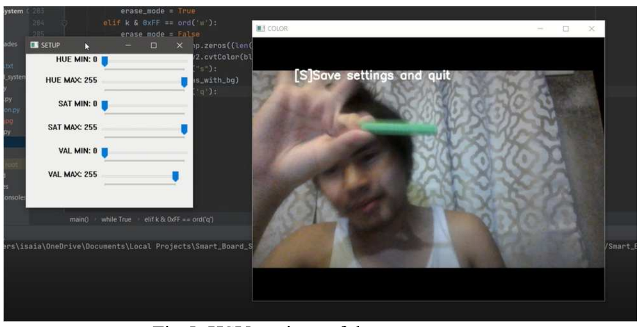


Fig. 4. Prototype Architecture

IV. RESULTS AND DISCUSSION

A. *Color Thresholding*

Isolating the color was implemented using the inRange function OpenCV library. The algorithm accepts images in HSV color space and inRange function removes the pixels that do not fit within certain HSV parameters, by setting it to zero value. Since the objective is to let the user decide which object to tracking, the HSV parameters should be set by the user, through a calibration phase. The idea behind the setup phase is to let the user isolate the color of the object in their way. The key here is to provide instructions on which value to change and make it simple to use. Fig 5 shows the sample UI of the setup program.



```
erase_mode = True
elif k & 0xFF == ord('e'):
erase_mode = False
```

SETUP

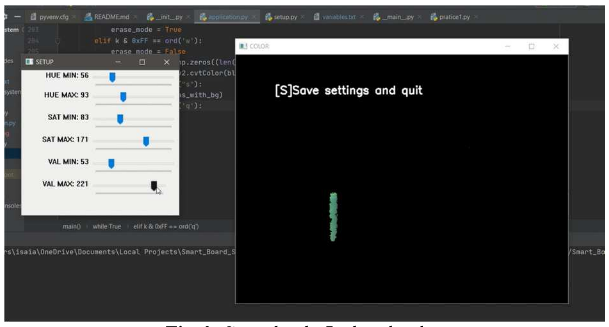
HUE MIN: 0
HUE MAX: 255
SAT MIN: 0
SAT MAX: 255
VAL MIN: 0
VAL MAX: 255

COLOR

[S]Save settings and quit

Fig 5. HSV settings of the setup program

The inRange function takes in 6 parameters which are the minimum and the maximum of the following: Hue, Saturation, and Value. When the setup is launched, the maximum of each parameter is set to the absolute max (which is 255) while the minimum is set to the absolute min (which is 0). The process of isolating the color is simple, the minimum is moved closer to the other side until the color of the object disappears. This is also what should be done to the maximum just in a different direction. Fig. 6 shows a completely color-isolated object. This frame in which the color is isolated is referred into this paper as the mask frame.



```
erase_mode = True
elif k & 0xFF == ord('e'):
erase_mode = False
```

SETUP

HUE MIN: 56
HUE MAX: 93
SAT MIN: 83
SAT MAX: 171
VAL MIN: 53
VAL MAX: 221

COLOR

[S]Save settings and quit

Fig 6. Completely Isolated color

After setting the values for the program, the next step is tracking the object and making it write on the screen or canvas. This entire process is broken into two: tracking and drawing. The idea behind tracking is discussed in the previous section. After the color of the object is isolated, the edges of that object are then detected using the getContours method from the OpenCV library. The contour method returns a contour object wherein its location, perimeter, area, and rectangle that bounds it can be computed via a simple function. Algorithm 1 presents this process.

Algorithm 1 Get pen point

```
Input mask_frame, previous_point
Output point
1.  contours = cv2.findContours( mask_frame )
2.  try
3.      if cv.contourArea(contour) > 10 do
4.          x,y,w,h = cv2.boundingRect(contour)
5.          return ( x+w/2 , y+h/2 )
6.      else
7.          return point
8.  except  noContour
9.      return ( -1. -1)
```

After getting the point, the program decides whether or not the program will draw a line or a point on the screen. If the pen point function returns a negative value, nothing will be written. If the previous point is the negative value then the only thing that would be written would be a point. A line will be drawn into the canvas if the previous point is not a negative value and the get contour function returns a valid point. This process is presented in algorithm 2.

Algorithm 2 Draw on canvas
Input canvas, erase_canvas, point, previous_point, erase_mode
Output canvas

```
1.  if point == (-1, -1)
2.      pass
3.  else if prevous_point == (-1, -1)
4.      return draw_point_canvas_point(canvas, erase_canvas,
    erase_mode)
5.  else
6.      if not erase_mode
7.          return cv2.line(canvas,poin1,point2)
8.      else
9.          return cv2.line(erase_canvas,point1,point2)
```

Erase mode is a Boolean value and erase canvas is the canvas wherein if erase mode is true, the point or line will be written there instead. When erase mode is enabled by the user, the drawings will be written there instead of the actual canvas. The point of having two canvases is for erasing. When erase mode is enabled, the drawing will be written on erase_canvas. Using OpenCV functions, erasure can be done by using the erase_canvas as a subtraction to the actual canvas. This is done for every loop which means that regardless if erase mode is true or false, the erase_canvas is always subtracting the canvas producing the final_canvas. This is presented in algorithm 3.

Algorithm 3 Get final canvas

Input canvas, erase_canvas
Output final_canvas

```
1.  erase_canvas = cv2.bitwise_and(canvas, erase_canvas)
2.  final_canvas = cv2.bitwise_xor(canvas,erase_canvas)
```

The first bitwise operation means that if a certain pixel has the same color between the two frames, it stays. If that pixel, however, has no color in one of the two canvases, that pixel is converted to a zero value. The idea behind here is that everything that is not overlapping is removed. The second line on the other hand is a xor function which entails that the pixel will only remain if only one of the two has color. So the overlapping part of the erase_canvas is now subtracted from the final_canvas. This is the very principle of erasure in this program. Additionally, there is also an option of one-time erasure or clearing the whole canvas, which can be used when moving on to the next illustration.

To enter record mode, the keyboard button 'R' must be pressed to start recording everything the webcam and the electronic board see and wait for the user to press the keyboard button 'T' to stop the recording and export the file to a video file. This is useful for creating videos showing how a certain drawing is made using the smart board application.

The most important part of the project is its overall usability. The authors tested this prototype and are pleased with the results as it tracks the object accurately when the background is clear. Fig. 7 shows the sample usable scenario.

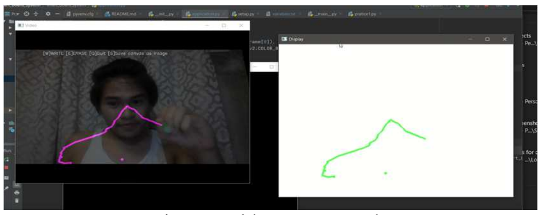


Fig 7. Usable state example

The issue arises there are objects in the background with the same color as the object being tracked. This falsely detects random images as the object which can ruin the drawing.

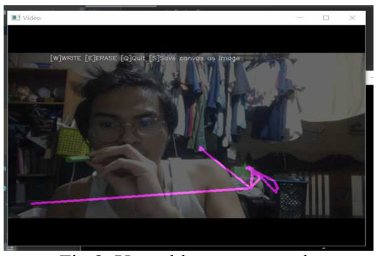


Fig 8. Unusable state example

B. Mediapipe-based Finger Tracking

Although the color thresholding method is simple, it is most prone to error as shown in the previous section. Thus, we also utilize an alternative method with the MediaPipe Hands tracking solution [9]. Additionally, since writing on air in front of the screen is not practical for some illustration and handwriting, a camera facing down setup, as shown in Fig. 9, is also introduced.

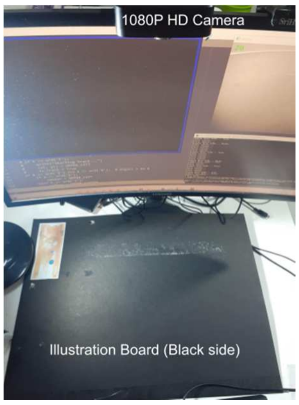


Fig 9. Top Camera Facing Down Setup

In particular, MediaPipe Hands object was instantiated with the following parameters: static_image_mode – False to handle input images as a stream or sequence, model_complexity – 1 for better landmark accuracy, min_detection_confidence and min_tracking_confidence of 0.75, as observed to compromise between hand detection/tracking accuracy and detection.

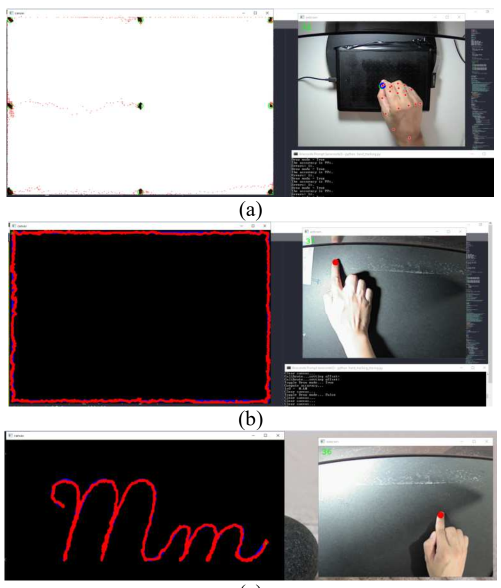


Fig 10. Testing the proposed system - (a) Example rectangle pattern calibration, (b) Corner points and center calibration; (c) Example handwriting tracing, e.g. letter 'M'

There are three parts of testing proposed system: (a) Static test or calibration in which the finger is positioned to the corners - upper-left, upper-right, lower-left, lower-right, center-left, center-right, upper-center, lower-center, and center, as shown in Fig. 10-a; (b) Rectangle pattern testing, in which the index fingertip is used to trace the border rectangle (Fig. 10-b); (c) Alphabet tracing using OpenCV's Hershey script simplex font, in which the index fingertip will trace the capital and small letters of the English alphabet while being tracked by the Mediapipe instance. The alphabet tracing is also exemplified in Fig. 11, where the left upper-lower case letter pair is the synthetic image, while immediately right to it depicts an example of hand tracing from the proposed system. It can be observed that the latter is jittery compared to the original letter pattern. The tracing accuracy was computed by dividing the intersection (between original and traced) over the original pixels, called Intersection-over-Pattern (IoP). The results indicate that letter D has the lowest IoP with 75, while letters N, F, L, have 86, 85, and 85, respectively. The average IoP for the alphabet with the inclusion of the rectangle pattern is 81.52 % which suggests good hand detection and tracking capability with our top camera setup.

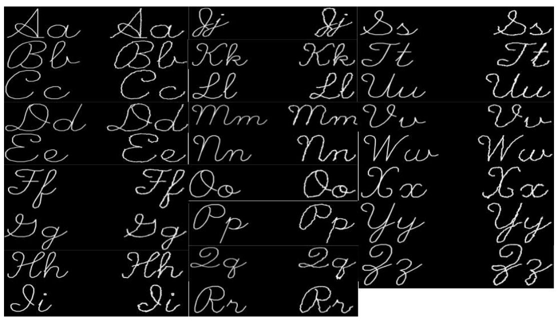


Fig 11. All alphabet letters example tracing - Right Letters: Synthetic (OpenCV Hershey script simplex font), Left to it is the vision-based hand tracing result.

TABLE I. ALPHABET / PATTERN TRACING ACCURACY VALUES

Patterns	**Tracing Accuracy (IoP)**
Rectangle	82
A	80
B	79
C	83
D	75
E	80
F	85
G	80
H	80
I	80
J	82
K	82
L	85
M	81
N	86
O	83
P	82
Q	81
R	83
S	80
T	83
U	83
V	82
W	83
X	79
Y	83
Z	79
AVERAGE	**81.52**

V. CONCLUSION

In this paper, a vision-based hand tracking system for non-face-to-face interaction was designed and developed. The system was able to meet the requirements set during the planning phase – i.e. to detect and track a hand that will serve as the pen or marker, allow that marker to make drawings and handwritings based on the movements made by the user, have an eraser function that will allow the user to erase or correct his or her drawing, and have a function where the work made on the canvas be exported to an image file. In future research, better cameras will also lead to better performance for the software since the image and video capture is clearer. The program does what it does, promote HCI with minimal hardware requirements. The software can be used as a baseline for future software developments, feature-wise. Since given its current state, the features to track an object, by color, draw, erase, clear, and capture an image or video are already present, thus these features can be further expanded upon to improve the overall functionality of this software. For example, the feature to draw can possess "pen" color-changing capabilities, depending on the user's desire. The feature to capture multimedia (image or video capture), can also possess the ability of media playback as well. Most notably, the ability of the program to track objects may include not only object recognition and tracking but gesture recognition as well - to perform program functions.

ACKNOWLEDGMENT

The authors of this paper would like to thank the Philippine Department of Science and Technology for their continued support through the Engineering Research and Development for Technology (ERDT).

REFERENCES

[1] J.M. Caroll, Human-computer interaction in the new millenium, 2002, New York: ACM Press.

[2] K. Oka and Y. Sato "Real-Time Fingertip Tracking and Gesture Recognition" IEEE proceeding on Computer Graphics and Applications Nov/Dec 2002.

[3] Q.C. Nicolas, D. Georganas, and E.M. Petriu, "Hand Gesture Recognition Using Haar-Like Features And A Stochastic Context-Free Grammar" IEEE,Vol. 57, No. 8, August 2008.

[4] Soowoong Kim, Jae-Young Sim, And Seungjoon Yang "Vision-Based Cleaning Area Control For Cleaning Robots", IEEE Vol. 58, No. 2, May 2012.

[5] D. Demirovic. An Implementation of the Mean Shift Algorithm. *Image Processing On Line*, pp. 251–268, 2019.

[6] D. DeTone, T. Malisiewicz, & A. Rabinovich. Deep Image Homography Estimation. arXiv preprint arXiv:1606.03798.

[7] G. R. S. Murthy & R. S. Jadon "A Review of Vision-Based Hand Gestures Recognition" International Journal of Information Technology and Knowledge Management, July-December 2009, Volume 2, No. 2, pp. 405-410.

[8] M. Novack and S. Goldin-Meadow, "Learning from Gesture: How Our Hands Change Our Minds," Educational Psychology Review, vol. 27, no. 3, pp. 405–412, 2015.

[9] 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.

[10] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.L. Chang, Yong, M.G., Lee, J. and Chang, W.T., 2019. Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.

[11] A.V. Savin, V.A. Sablina, and M.B. Nikiforov, Comparison of Facial Landmark Detection Methods for Micro-Expressions Analysis. In 2021 10th Mediterranean Conference on Embedded Computing (MECO) (pp. 1-4). IEEE, 2021, June.

[12] A. Halder, and A. Tayade, Real-time vernacular sign language recognition using mediapipe and machine learning, ISSN, 2582, p.7421, 2021.