RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY,
NAGPUR, MAHARASHTRA

2021-2022

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING
IN
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted to

ANJUMAN COLLEGE OF ENGINEERING & TECHNOLOGY

REAL-TIME HAND GESTURE RECOGNITION REPORT

Submitted by

Tejas Sontakke (B-46) Tarul Barve(B-47)

Vaibhav Wankar(B-48) Abhishek vairagade (B-49)

Sufiyan Nawab(B-50)

Under the guidance of

Prof. Imteyaz Shahzad ,
Prof. Imran Ahmad.

CONTENTS

Real-time Hand Gesture Recognition

Abstract

We present an on-device real-time hand gesture recognition (HGR) system, which detects a set of predefined static gestures from a single RGB camera. The system consists of two parts: a hand skeleton tracker and a gesture classifier. We use MediaPipe Hands [14, 2] as the basis of the hand skeleton tracker, improve the keypoint accuracy, and add the estimation of 3D keypoints in a world metric space. We create two different gesture classifiers, one based on heuristics and the other using neural networks (NN).

1.INTRODUCTION

Hand gesture recognition (HGR) is a natural and intuitive method for humancomputer interaction (HCI), and has been an active research area [11, 10]. A wide variety of input devices and techniques have been investigated, and skeletonbased HGR is a popular choice due to its robustness to background and light variations [7, 6]. Many skeleton-based HGR systems rely on depth sensors, such as RGBD cameras, which are not nearly as common as RGB cameras on mobile devices. Our HGR, on the other hand, requires only a single RGB camera. It does so by first predicting 3D skeleton keypoints from a camera image, then running a gesture classifier on the keypoints.

We design two gesture classifiers with different use cases in mind. The heuristicsbased classifier is easier to create and extend, without the need for training data, and more intuitive to develop and troubleshoot. The NN-based classifier is more accurate and precise, especially for borderline cases. It’s also more forgiving of errors in skeleton keypoints.

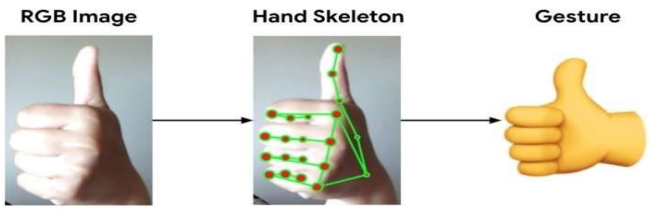


Figure 1. Our hand gesture recognition system

2.ARCHITECTURE

Our HGR consists of two parts: a hand skeleton tracker improved from MediaPipe Hands and a gesture classifier, as shown on Figure 1. The two-step approach has a few advantages:

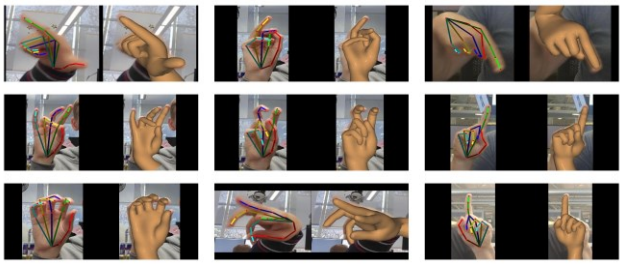
- Reduced engineering effort by leveraging the hand tracker which is already realtime, robust, and fair [4].
- Simpler gesture classifier design which processes skeleton keypoints instead of raw pixels.
- Optimized complexity by running the gesture classifier only when hands are tracked.

2.1.Hand Skeleton Tracker

Both our gesture classifiers operate on a keypoint level (not with RGB data), therefore accurate hand keypoint estimation is a key prerequisite for gesture classification. As a basis for our gesture recognition pipeline, we improve MediaPipe Hands [2] by training with a new sophisticated hand poses data (like American Sign Language). Analysis of various gestures indicates that robust estimation of hand rotation angle and normalization distance is key to an accurate hand tracker. The original hand rotation and scale estimation is based

on the 2D vector from the wrist to the middle finger knuckle. For various cases (like frontal view) such

Figure2.Handsrotationanglederivedfromthesumoftwovec - tors:indextopinkybaseknuckle(ingreen)andmiddlebase knuckletowrist(inred).



normalization becomes very unstable as the normalization distance approaches zero, which results in a significant degradation in tracking quality. To overcome this problem, we introduce a new algorithm similar to the approach taken by BlazePose [5].

For our case, we define two virtual keypoints to describe hand center, scale and rotation angle: a centerkeypoint and an alignment keypoint. The centerkeypoint is estimated as the average of the index, middle and pinky knuckles. The alignment keypoint location is estimated so it forms the rotation/scale vector with the centerkeypoint. The rotation angle is estimated from a sum of two vectors: from the middle base knuckle to the wrist, and from the index to the pinky base knuckle. As the component vectors tend to be orthogonal in the majority of cases; the resulting sum vector changes smoothly for any hand pose, and never degrades to zero, as shown on Figure 2. This increases overall hand tracking quality for frontal hand cases. The scalar value of the alignment vector is estimated as the distance from the centerkeypoint to the farthest knuckle of the same hand. The new rotation and scale normalization results in a significant quality boost for the whole hand pose estimation pipeline: 71.3 mAP vs 66.5 mAP (for the original MediaPipe Hands [2] pipeline) on our validation dataset with complex ASL hand poses.

Accurate hand pose estimation in the 3D space is a vital component for both angle based and few-shot learning gesture classification. It minimizes the ambiguity among projections of the same hand pose from different observer positions in space, and allows the gesture classification to be invariant to rotation. Therefore, in addition to predicting the hand pose in the screen “pixel” space, we also estimate the pose in a world metric space relatively to the hand wrist. To obtain a 3D hand pose ground truth in a metric space, we fit our 2D hand annotation with a statistical and highly realistic GHUM [12, 13] hand model as shown in Figure 3. Due to the nature of perspective projection, objects of different sizes may have the same projection on the image plane: two objects with the same shape (bigger and smaller) will have the same projection if placed respectively further and closer to the camera. Therefore we have to make the following assumptions when we perform the hand fitting from 2D key-

Figure 3. Sample images with overlaid 2D annotations and the corresponding GHUM [12] hand models fitted and rendered on top.

points: human hands have minor variations in size, those variations are always reflected in hand shapes, and thus covered by the statistical GHUM hand model. For images with unknown camera intrinsic parameters, we assume the focal length is the maximum of the image width and height, and the optical center is the center of the image. Since both the model training and inference operate on a cropped image, we normalize the absolute world coordinates such that the origin is at the middle finger knuckle, while the scale remains the same. We also normalize 3D coordinates in a roll plane using virtual keypoints to be consistent with input image transformations.

2.2. Heuristics Gesture Classifier

Based on the hand skeleton tracker, we build a singleshot, heuristics-based classifier for a small set of static gestures. We start with the simple gesture classification approach described in [14], which first derives a set of angles between various 2D hand keypoints, then applies thresholds to the derived angles to define a discrete state for each finger (e.g. bent or straight), and finally defines a static gesture as a logic expression based on the finger states. To get a more accurate set of underlying angles, we replace features based on the 2D hand keypoints with features based on the 3D world metric hand keypoints. In order to de-correlate the gesture classifier features and make manual threshold picking easier, we distinguish between extrinsic and intrinsic features with respect to the palm pose in the 3D world metric space during the preprocessing stage:

- The extrinsic features are composed of palm pose components such as rotation, scale and translation. For classification purposes, we only use the rotation pose component of the palm represented by its three Euler angles.

- The intrinsic features are angles between various hand keypoints. For classification purposes, we derive a single feature angle for each finger with a goal of thresholding them to define a discrete state (e.g. fully bent, fully straight or neither). In accordance with the underlying hand skeleton topology, each finger is represented by a base joint keypoint, two intermediate joint keypoints and a tip keypoint. To define an individual finger feature angle, we first introduce a 3D polygonal chain: starting from the wrist keypoint, to the finger's base joint keypoint, the first intermediate joint keypoint, the second intermediate joint keypoint, and finishing at the tip keypoint. Then, we define the feature angle as the maximum angle between the first chain segment and each of the remaining chain segments. Additionally, we derive a feature angle for each adjacent pair of fingers with the same goal of thresholding them to define a discrete state (e.g. fingers crossed, apart or neither).

As shown on Figure 4, removing the influence of the extrinsic features from the the intrinsic features produces a consistent 3D hand keypoint set for a fixed hand shape configuration, regardless of its position on the input frames. In turn, this significantly de-correlates features and makes it manageable to manually pick thresholds on a larger feature set in order to define a more complex static gesture.

As the final stage of the heuristics-based classifier, we establish a system of 6 gesture definitions based on the features derived from the pre-processed 3D world metric hand keypoints. Figure 5 showcases the supported gestures. This particular gesture set is chosen so that it covers some of the most recognizable and common static hand gestures.

To evaluate our approach, we collect and annotate an in-house gesture dataset. The dataset contains 1882 short video clips that cover various angles and lighting conditions for 21 static hand gestures (for the full list of gestures, please see Appendix A). The dataset contains all 6 gestures supported by the heuristics-based classifier (see Figure 5)
and those are used as positive samples. The remaining 15 gestures are used as negative samples. The presence of a diverse negative sample collection allows us to evaluate how often the classifier recognizes an unknown gesture as a known one. The limitation of this dataset is that it's collected from only 18 users with limited variation in background. The classifier achieves 0.86% false positive rate and 44.4% recall rate on the dataset.

2.3.NN Gesture Classifier

The dataset used in Section 2.2 is for evaluation only and too small for training NN models. We collect and mine another in-house dataset containing 7307 images from 6478 users, representing a rich variety of hand shapes of both gestures and non-gestures in the wild. In addition to regular positive samples we collect easy and hard Negative samples. Please see Figure 6 for some example images.

We train an NN classifier to distinguish among six static hand gestures, namely, OpenPalm, ClosedFist, PointingUp, Victory, ThumbUp, ThumbDownand a background Negative class. The NN model consists of 3 fully connected layers of 50 neurons each. The model inputs are the intrinsic and extrinsic features computed in Section 2.2. We use focal loss [8] to deal with the class imbalance, where there are a lot more negative samples than positive samples in our dataset.

The NN classifier achieves an average recall rate of 87.9% across the 6 static gesture classes at a false positive rate of 1%.


Hand Gesture Recognition
REAL-TIME HAND GESTURE RECOGNITION
Start
Close window
Terminate Recogniser
Hand Gesture Recognition
live long
Type here to search


Hand Gesture Recognition
REAL-TIME HAND GESTURE RECOGNITION
Start
Close window
Terminate Recogniser
Hand Gesture Recognition
rock


Hand Gesture Recognition
REAL-TIME HAND GESTURE RECOGNITION
Start
Close window
Terminate Recogniser
Hand Gesture Recognition
stop

Figure 5. Visualization of gestures supported by the heuristicbased classifier. Top-to-bottom: Live long ,rock,stop,thumbsUp

3.UML DIAGRAM

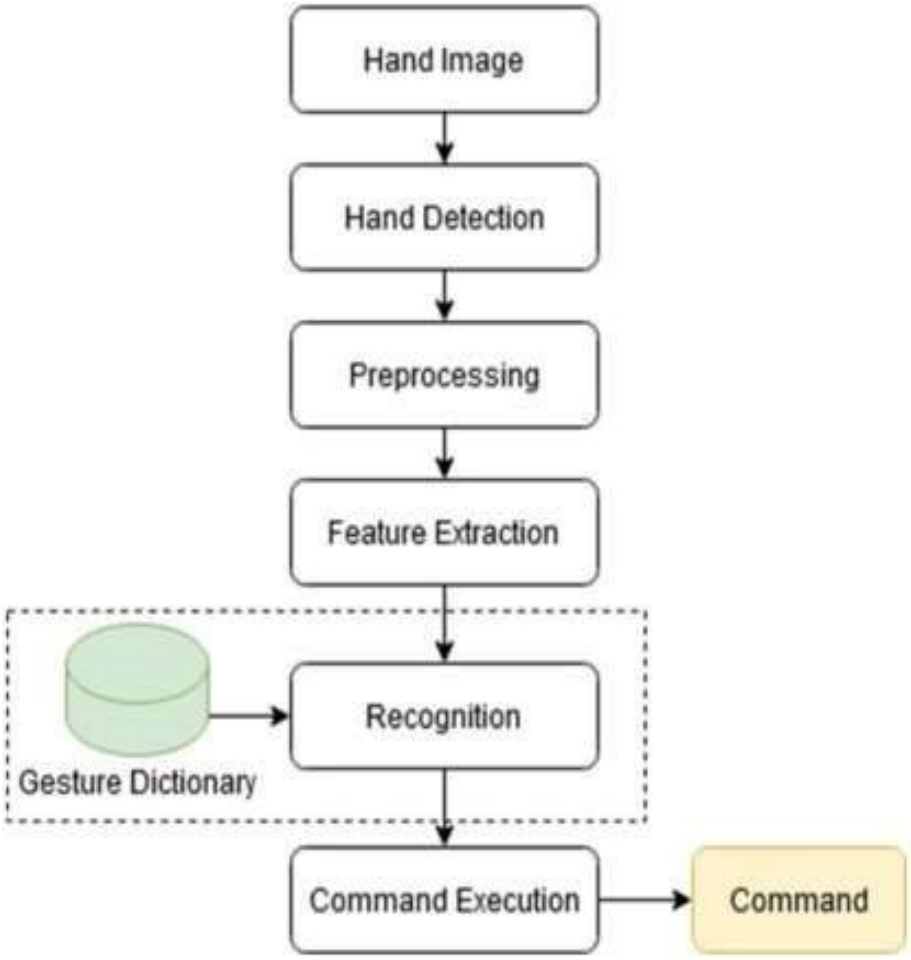


Figure 6. UML diagram for hand gesture recognition system

4.MEDIAPIPE IMPLEMENTATION

The proposed HGR system is implemented using the open source MediaPipe framework [1]. The system adds to MediaPipe Hands, primarily consisting of a handdetection and a hand-keypoint component [14, 2], an additional gestureclassification component as discussed in Section 2.2 and 2.3.

In many applications, such as remote control, the user gestures only once in a while but the HGR is always running in the background. In order to reduce average computation requirement and maintain real-time performance across a wide range of devices, the HGR system hand de-

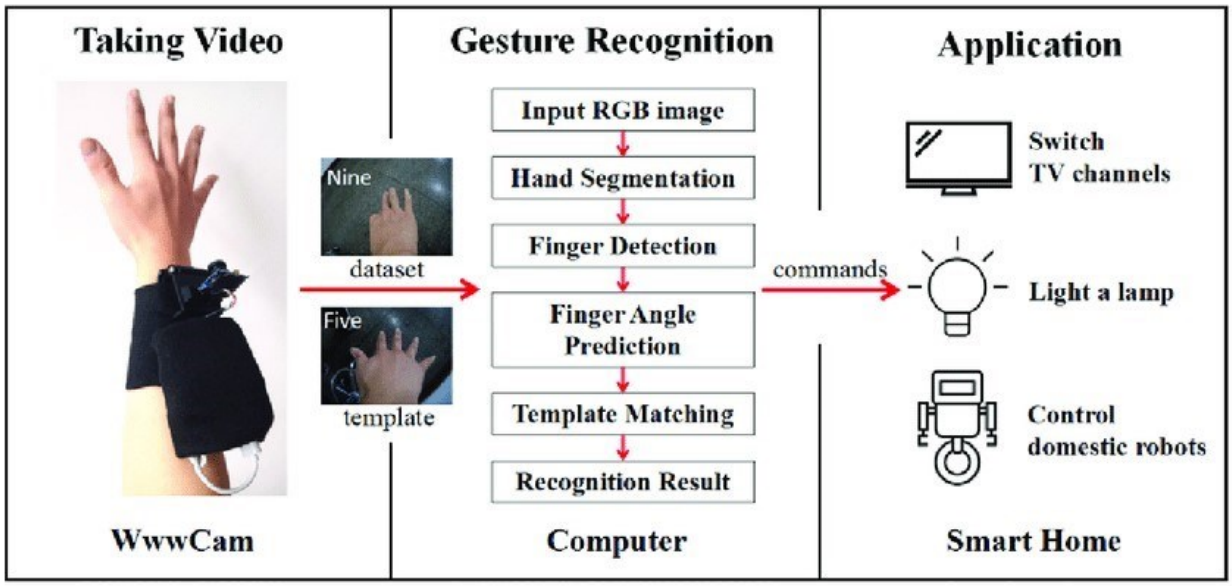
Figure 7. Some examples of true positive samples for gesture classes, easy samples for Negative hand shapes and subtle variations of hand shapes that should not be confused with the gesture class.

tection is configured to run only as needed, capped at a maximum frequency lower than the frequency of hand keypoint generation and gesture classification, by utilizing the flow-control and stream-synchronization support in MediaPipe [9] similar to [14]. When there's no hand in the camera view, the HGR runs hand detection at a lower frequency to save computation and power. As soon as a hand is detected, it's tracked at a higher frequency for better accuracy and temporal resolution. Furthermore, GPU acceleration is heavily exploited end-to-end, covering tasks like ML model inference as well as image and tensor processing, via OpenGL/OpenCL/Metal on mobile devices and WebGL locally in Web browsers (similar to the Web ML effort enabling background blur/replace in Google Meet [3]).

5.APPLICATIONS

Our HGR can be used as an HCI mechanism for various applications, such as a

- virtual touchscreen for desktop computers,
- sending visual commands for robots,
- a controller for virtual reality gaming systems, and
- a remote control for large screen displays .



Gesture Code Names

1. okay
2. peace
3. thumbs up
4. thumbs down
5. call me
6. stop
7. rock
8. live long
9. fist
10. smile

6.SOURCE CODE

```
# Group-5 ::: Hand Gesture Recognizer


# import necessary packages
 importcv2 importnumpyasnp importmediapipeasmp
importtensorflowastf fromtensorflow.keras.modelsimportload_model


# initializemediapipe mpHands = mp.solutions.hands hands = mpHands.Hands(max_num_hands=1,
min_detection_confidence=0.7) mpDraw = mp.solutions.drawing_utils
# Load the gesture recognizer model model =
load_model('mp_hand_gesture')


# Load class names f = open('gesture.names', 'r')
classNames = f.read().split('\n')
f.close() print(classNames)

# Initialize the webcam(OpenCV) cap =
cv2.VideoCapture(0)
 whileTrue:
    # Read each frame from the webcam
    _, frame = cap.read()
     x, y, c = frame.shape # Flip the frame vertically     frame = cv2.flip(frame, 1)
framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


    # Get hand landmark prediction     result =
hands.process(framergb)


    # print(result)
```

```
    className = ''

    # post process the result     ifresult.multi_hand_landmarks:
        landmarks = []
forhandslmsinresult.multi_hand_landmarks:
forlminhandslms.landmark:                 # print(id, lm)                 lmx
= int(lm.x * x)                 lmy = int(lm.y * y)
                landmarks.append([lmx, lmy])

            # Drawing landmarks on frames             mpDraw.draw_landmarks(frame, handslms,
mpHands.HAND_CONNECTIONS)
            # Predict gesture             prediction =
model.predict([landmarks])
            # print(prediction)
            classID  =  np.argmax(prediction)                          className  =
classNames[classID]

    # show the prediction on the frame
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,                          1,
(0,0,255), 2, cv2.LINE_AA)

    # Show the final output     cv2.imshow("Output", frame)
     ifcv2.waitKey(1) == ord('q'):
        break


# release the webcam and destroy all active windows cap.release()
 cv2.destroyAllWindows()
```

7.REFERENCES

[1] MediaPipe. https://mediapipe.dev/, 2019. 3

[2] MediaPipe Hands. https://solutions. mediapipe.dev/hands, 2019. 1, 2, 3

[3] Background Features in Google Meet, Powered by Web ML. https://ai.googleblog.com/2020/10/ background-features-in-google-meet.html, 2020. 4

[4] MediaPipe Hands Model Card. https://mediapipe. page.link/handmc, 2020. 1

[5] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. BlazePose: On-device Real-time Body Pose Tracking, 2020. 2

[6] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1–9, 2016. 1

[7]Guillaume Devineau, Fabien Moutarde, Wang Xi, and Jie Yang. Deep learning for hand gesture recognition on skeletal data. In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pages 106– 113. IEEE, 2018. 1

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In´ Proceedings of the IEEE international conference on computer vision, pages 2980–2988, 2017. 3

[8] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, ChuoLing Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. MediaPipe: A Framework for Building Perception Pipelines. In CVPR Workshop on Computer Vision for AR/VR, 2019. 4

[9] Munir Oudah, Ali Al-Naji, and JavaanChahl. Hand gesture recognition based on computer vision: a review of techniques. journal of Imaging, 6(8):73, 2020. 1

[10] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. Artificial intelligence review, 43(1):1–54, 2015. 1, 4

[11] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6184–6193, 2020. 2

[12] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly Supervised 3D Human Pose and Shape Reconstruction with Normalizing Flows. In Computer Vision – ECCV 2020, pages 465–481, 2020. 2

[13] Tkachenka. MediaPipe Hands: On-device Real-time Hand Tracking. In CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, 2020. 1, 2, 3, 4