HUMAN COMPUTER INTERACTION
(CSE 518)

HAND GESTURE RECOGNITION

SUBMITTED BY

PREETAM REDDY LINGAREDDYGARI (114189888)

Table of Contents

ABSTRACT

Gesture recognition is a popular topic in Human-Computer Interaction research. The physical contact between the system and the user is becoming obsolete in this age of technological transformation, where everything is getting automated. Gesture recognition comes into play here and plays a crucial role. It has a wide range of applications, including virtual environment control, sign language translation, music creation and robot control. We will create a real-time Hand Gesture recognition using OpenCV and Python.

INTRODUCTION

HAND GESTURE RECOGNITION

Gesture recognition is a topic in computer science and language technology with the goal of using mathematical algorithms to analyze human gestures. It is a computer vision subdiscipline. Gestures can come from any physical move or state, but they are most typically made with the hands or the face. Facial Emotion identification and hand gesture recognition are the two current priorities of the field. Simple motions can be used to control or interact with devices without having to physically touch them. Cameras and computer vision algorithms have been used in a variety of ways to read this sign language.

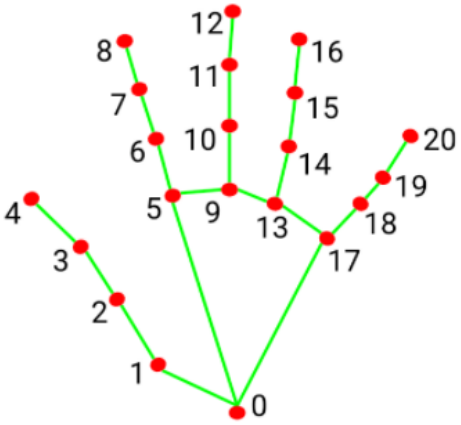
FRAMEWORKS USED

I used the MediaPipe framework to develop a real-time Hand Gesture Recognizer. Google's MediaPipe is a customizable machine learning solution architecture. It's an open-source, cross-platform framework with a small footprint. Face detection, position estimation, hand recognition, object detection, and other pre-trained ML solutions are included with MediaPipe.

Tensorflow model and OpenCV are employed for object identification. The Google Brains team created TensorFlow, an open-source machine learning and deep learning library. It can be used for a variety of tasks, but it focuses on deep neural networks in particular. OpenCV is a real-time computer vision and image processing library. OpenCV-python library is used here.

MediaPipe Hand is a machine-learning employed high-fidelity hand and finger tracking technology. It recognizes 21 Landmark points. With the help of numerous models operating concurrently, these crucial points are recorded from a hand in a single frame.These key points will be fed into a pre-trained gesture recognizer network to recognize the hand pose.

MEDIAPIPE KEY POINTS



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

IMPLEMENTATION

1. Import all required packages
2. Prepare the models.
3. Frames from a camera are read.
4. Detect the key points on the hands.
5. Recognize the hand gestures

Step 1 – Import necessary packages

To build this Hand Gesture Recognition project, we'll need four packages. So first import these.

- **mediapipe >= 0.8.3**
- **OpenCV >= 4.5.4**
- **Tensorflow >= 2.6.2**
- **Numpy >= 1.19.3**
- **Python >= 3.6.0**

Step 2 – Initialize models

Initialize MediaPipe:

- Mp.solution.hands module performs the hand recognition algorithm. So we create the object and store it in “mediapipeHand”.
- Using mediapipeHand.Hands method we configured the model. The first argument is max_num_hands, that means the maximum number of hand will be detected by the model in a single frame. MediaPipe can detect multiple hands in a single frame, but we’ll detect only one hand at a time in this project.
- Mp.solutions.drawing_utils will draw the detected key points for us so that we don’t have to draw them manually.

Initialize Tensorflow:

- Using the load_model function we load the TensorFlow pre-trained model.
- Gesture.names file contains the name of the gesture classes. So first we open the file using python’s inbuilt open function and then read the file.
- After that, we read the file using the read() function.
- Output :
- [‘okay’, ‘peace’, ‘thumbs up’, ‘thumbs down’, ‘call me’, ‘stop’, ‘rock’, ‘live long’, ‘fist’, ‘smile’]
- The model can recognize 10 different gestures.

Step 3 – Read frames from a webcam

- We build a VideoCapture object and feed it the value '0' as a parameter. It is the system's camera ID. In this scenario, the system is connected to one webcam. If you have many webcams, update the argument to the camera ID for each one. Otherwise, it's best to leave it alone.
- Each frame from the webcam is read using the FrameCapture.read() function.
- The function cvision.flip() flips the frame.
- cvision.imshow() opens a new openCV window with the frame.
- Until the key 'x is pushed, the cvision.waitKey() function keeps the window open.

Step 4 – Detect hand keypoints

- OpenCV reads BGR images, but MediaPipe works with RGB images. We transform the frame to RGB format using the cvision.cvtCOLOR() function.

- A RGB frame is passed to the process function, which returns a result class.
- The processed_result.multi_hand landmarks method is then used to see if any hands have been detected.
- Following that, we loop through each detection and save the coordinates in a landmarks list.
- Because the model gives a normalized result, the image height (y) and image width (x) are multiplied with the result. This signifies that the result's values are all between 0 and 1.
- Finally, we draw all of the landmarks in the frame using the mediaPipeDraw.draw_ landmarks() function.

Step 5 – Recognize hand gestures

- The model.predict() function takes a list of positions and produces a list of 10 prediction classes for each one.
- [[2.0691623e-18 1.9585415e-27 9.9990010e-01 9.7559416e-05 1.6617223e-06 1.0814080e-18 1.1070732e-27 4.4744065e-16 6.6466129e-07 4.9615162e-21]] [[2.0691623e-18 1.9585415e-27 9.9990010e-01 9.7559416e-05 1.6617223e-06 1.0814080e
- Np.argmax() returns the index of the list's maximum value.
- We may just take the gesture from the Gestures list after retrieving the index.
- The detected gesture is then shown in the frame using the cvision.putText method.

OUTPUT


peace
live long
thumbs up
thumbs down
call me
smile
fist
okay

CONCLUSION:

Based on the results/output, a simple Hand Gesture recognition application using OpenCV and Python could detect and recognize hand gestures like smile, okay, peace, live-long etc. This application also works for static images. We can use this application in various fields like robotics, controlling devices, artificial intelligence etc.

REFERENCES:

1. R.C. Gonzales and R.E. Woods, *Digital Image Processing,* Prentice Hall, 2008.
2. M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*
3. R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2011.
4. Md. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno, "Adaptive visual gesture recognition for human-robot interaction using a knowledge-based software platform", *Robotics and Autonomous Systems*, vol. 55, Issue 8, 31 August 2007, pp. 643-657. http://dx.doi.org.ezproxy.massey.ac.nz/10.1016/j.robot.2007.03.002