Hand gesture recognizer

Darya YARCHAK

Erasmus

An application in Python that uses neural networks to recognize a palm and to recognize few basic gestures.

# Methodology

## Libraries

The main library that I have used is mediapipe which is built upon another library that is used for machine learning, TensorFlow. Both of above-mentioned libraries are developed and maintained by Google and open source community. Also, for video input from the webcam and further output to a window with some drawings on it I use OpenCV.

## Environment

I have decided to use Python 3.10.13 as it is the latest version supported by TensorFlow. To set up the environment I have used miniconda which allows me to pick whatever version of any library I wish.

As a development tool I have used Visual Studio Code but later switched to PyCharm due to its superior out-of-box functionality. Later I have migrated the project to Visual Studio 2022

## Approach for solving the problem

1. The model I used is a [pretrained model made by Google](https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/index#models)
2. Recognition
   1. I open webcam to start reading frames using OpenCV

capture = cv2.VideoCapture(0)

* 1. I read a frame and perform some changes to it, namely: change of colour format and flip of the image horizontally so it appears to the user as if they look into a mirror.

ret, frame = capture.read()

frame = cv2.flip(frame, 1)

mp\_image = mp.Image(image\_format=mp.ImageFormat.SRGB, data=frame)

* 1. I use a predefined class GestureRecognizer with options such as the used model, mode, maximum number of hands in frame and confidence for hand recognition. Using the initialized object of the class I call the function recognize\_for\_video and pass to it the current frame and current timestamp.

options = GestureRecognizerOptions(

base\_options = BaseOptions(model\_path),

running\_mode = VisionRunningMode.VIDEO,

num\_hands = 3,

min\_hand\_presence\_confidence = 0.4)

with GestureRecognizer.create\_from\_options(options) as recognizer:

gesture\_recognition\_result = recognizer.recognize\_for\_video(mp\_image, int(capture.get(cv2.CAP\_PROP\_POS\_MSEC)))

1. Processing of the results
   1. I use a loop to go through all the hands on the current frame and then to go through all the landmarks recognized by the model. I use OpenCV to draw the landmarks and connections between them.

for landmark in hand\_landmark:

h, w, \_ = frame.shape

cx, cy = int(landmark.x \* w), int(landmark.y \* h)

cv2.circle(frame, (cx, cy), 4, (255, 0, 0), -1)

for connection in connections:

start\_point = (int(hand\_landmark[connection[0]].x \* w), int(hand\_landmark[connection[0]].y \* h))

end\_point = (int(hand\_landmark[connection[1]].x \* w), int(hand\_landmark[connection[1]].y \* h))

cv2.line(frame, start\_point, end\_point, (0, 0, 255), 1)

* 1. Render text that specifies the recognized gesture.

text\_position = (int(hand\_landmark[0].x \* w), int(hand\_landmark[0].y \* h))

cv2.putText(frame, gesture\_name(gesture\_recognition\_result.gestures[i][0].category\_name), text\_position, cv2.FONT\_HERSHEY\_DUPLEX, 1, (255, 255, 255), 2, cv2.LINE\_AA)

# Results

The application can recognize hand and 7 gestures namely:

* None 🫳
* Closed fist ✊
* Open palm 🖐️
* Pointing up ☝️
* Thumb down 👎
* Thumb up 👍
* Victory ✌️
* Love 🤟

# Conclusion

This is a simple proof-of-concept application that can be improved in a multitude of ways. For instance, another trained model that will recognize bigger amount of gestures or even gestures that require two palms. Another option may be also addition of face emotions recognition that will combine with gesture recognition to better understand person’s emotion.

This can be used in various spheres, for example in China street cameras can detect peoples’ mood and based on that algorithmically decide which advertisements to show them or change their life in another way. Also I can think of a use case for VR technology for virtual online chats where such a model may be used to portray people more realistically.

# Start-up guide

The project is launched either in Visual Studio or in terminal. If you launch Visual Studio and get an error about environment, please press on “Add existing environment”, “Existing environment”, put name as “env”, and set prefix as the env folder inside the project. For some unforeseen to me reason whenever the folder is moved the environment needs this re-setup, and this is not caused because of absolute path as I opened the project files in notepad and it for sure uses relative path, so I think this is just a bug.

Another way of starting the application is using terminal, to do that you will need to navigate to the project directory and launch the Python interpreter located inside the environment, so the command should look like *.\env\python main.py.* The usage of environment ensures that the interpreter version is compatible and well tested with the libraries used.

# References

* The main article I used: <https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/python>
* The one I didn’t end up using but it contained information on which libraries I should use and it generally guided me: <https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/>
* Also used mainly for guidance: <https://mudgalvaibhav.medium.com/real-time-gesture-recognition-using-googles-mediapipe-hands-add-your-own-gestures-tutorial-1-dd7f14169c19>
* Google’s presented way of doing this: <https://mudgalvaibhav.medium.com/real-time-gesture-recognition-using-googles-mediapipe-hands-add-your-own-gestures-tutorial-1-dd7f14169c19>
* General guidance: <https://medium.com/@odil.tokhirov/how-i-built-a-hand-gesture-recognition-model-in-python-part-1-db378cf196e6>
* Drawing in OpenCV: <https://docs.opencv.org/3.4/dc/da5/tutorial_py_drawing_functions.html>