Lab03 (Image) Screenshots

1. Introduction (10 minutes)
   Computer vision has been a very popular field since the advent of digital systems. However, computer vision on the edge devices such as Raspberry Pi is challenging due to resource constraints. Edge Computer Vision (ECV) has emerged as a transformative technology, with Gartner recognizing it as one of the top emerging technologies of 2023. ECV offers several benefits such as 1) they can operate in real-time or near-real-time, providing instant insights and enabling immediate actions, 2) they offer enhanced privacy and security and 3) It reduces dependency on network connectivity or relaxes the bandwidth requirements as some processing will be done within. In this lab, few basic and advanced image processing tasks on edge devices is introduced.

2. Setting up the Raspberry Pi

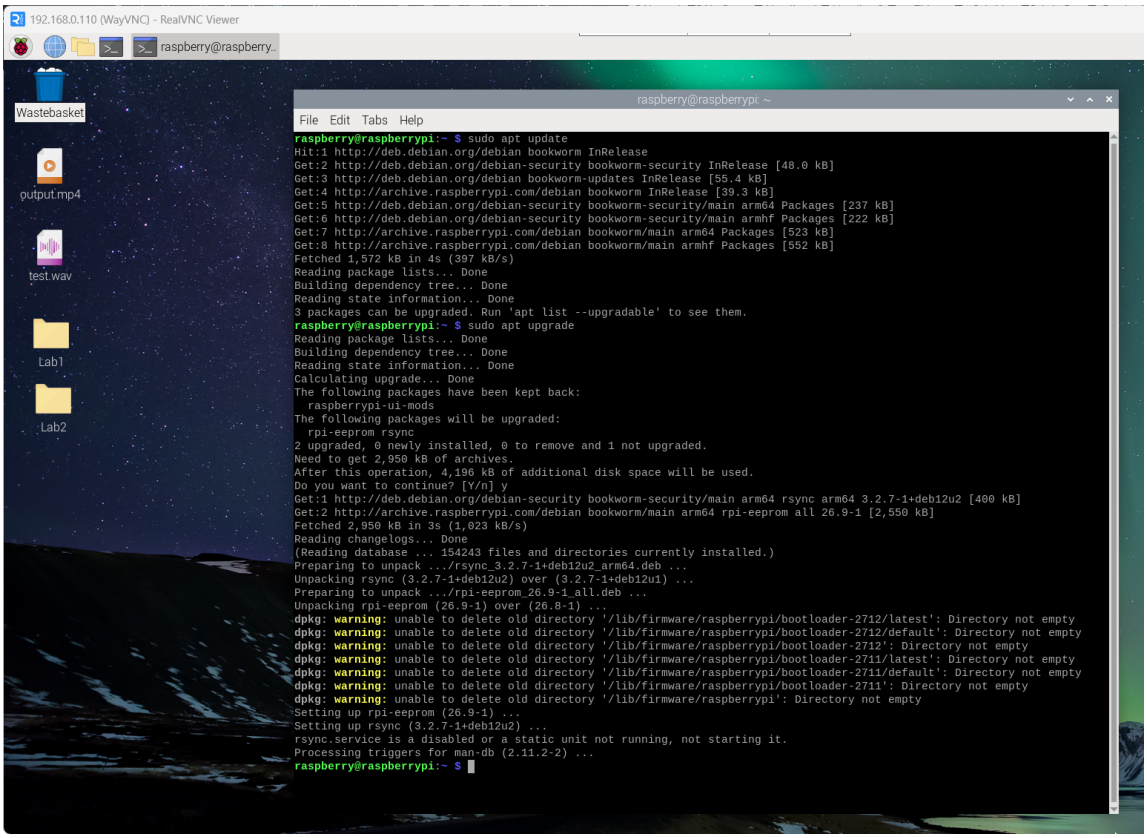

Figure 1. Screenshot of system update of Raspberry Pi OS.

Lab03 (Image) Screenshots

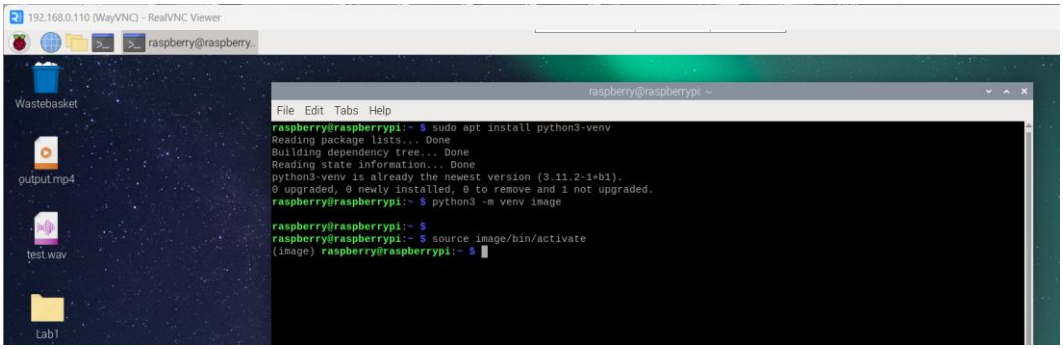

Figure 2. Screenshot of activating a virtual environment named “image” to avoid conflicts in libraries.

3. Connecting and Testing the Web Camera (15 minutes)

Figure 3. Screenshot of physically connect the web camera to the Raspberry Pi via USB connection.

4. Introduction to Real-time Image Processing with Python (25 minutes)

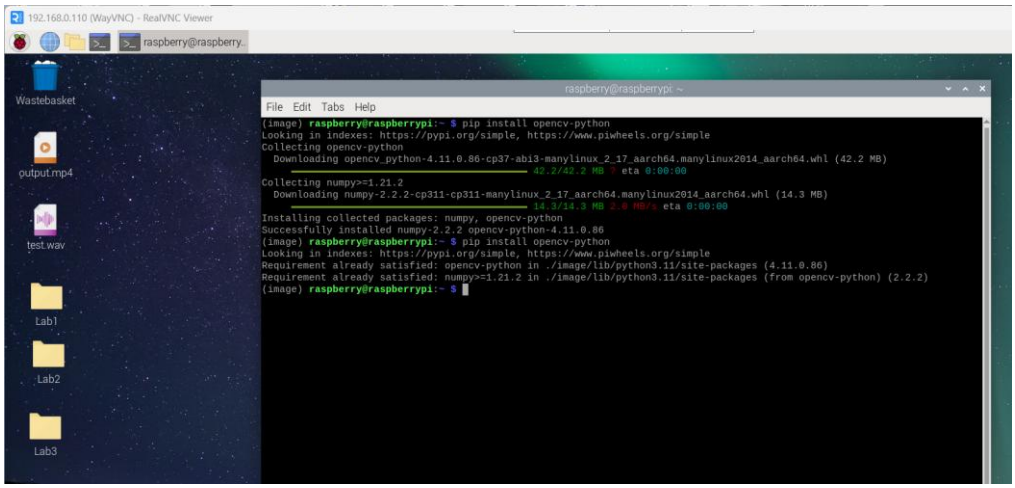

Figure 3. Screenshot of installation of OpenCV library.

```python
# Reference: https://pyimagesearch.com/2014/08/04/opencv-python-color-detection/
import cv2
import numpy as np

#%% Defining a list of boundaries in the RGB color space
# (or rather, BGR, since OpenCV represents images as NumPy arrays in reverse order)
# Refer to https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html
boundaries = [
    ([17, 15, 100], [50, 56, 200]), # For Red
    ([86, 31, 4], [220, 88, 50]), # For Blue
    ([25, 90, 4], [62, 200, 50]), # For Green
    ([0, 190, 190], [80, 255, 255]) # For Yellow
]

#%% Normalize the Image for display (Optional)
def normalizeImg (Img):
    Img= np.float64(Img) #Converting to float to avoid errors due to division
    norm_img = (Img - np.min(Img))/(np.max(Img) - np.min(Img))
    norm_img = np.uint8(norm_img*255.0)
    return norm_img

#%% Open CV Video Capture and frame analysis
cap = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")

# The loop will break on pressing the 'q' key
while True:
    try:
        # Capture one frame
        ret, frame = cap.read()

        output=[]

        # loop over the boundaries
        for (lower, upper) in boundaries:
            # create NumPy arrays from the boundaries
            lower = np.array(lower, dtype = "uint8")
            upper = np.array(upper, dtype = "uint8")

            # find the colors within the specified boundaries and apply the mask (basically segmenting for colours)
            mask = cv2.inRange(frame, lower, upper)
            output.append(cv2.bitwise_and(frame, frame, mask = mask)) #Segmented frames are appended

        # Output is appeneded to be of size Pixels X 4 (for R, G, B and Y)
        red_img = normalizeImg(output[0])
        green_img = normalizeImg(output[1])
        blue_img = normalizeImg(output[2])
        yellow_img = normalizeImg(output[3])

        # horizontal Concatination for displaying the images and colour segmentations
        catImg = cv2.hconcat([frame,red_img, green_img, blue_img, yellow_img])
        cv2.imshow("Images with Colours",catImg)

        if cv2.waitKey(1) & 0xFF == ord('q'):
           break

    except KeyboardInterrupt:
        break

cap.release()
cv2.destroyAllWindows()

#end
```

Figure 4. Screenshot of expanding the sample code to segment the colour yellow, on top of the Red-Green-Blue (RGB) colour channels.

5. Real-time Image Analysis (25 minutes)

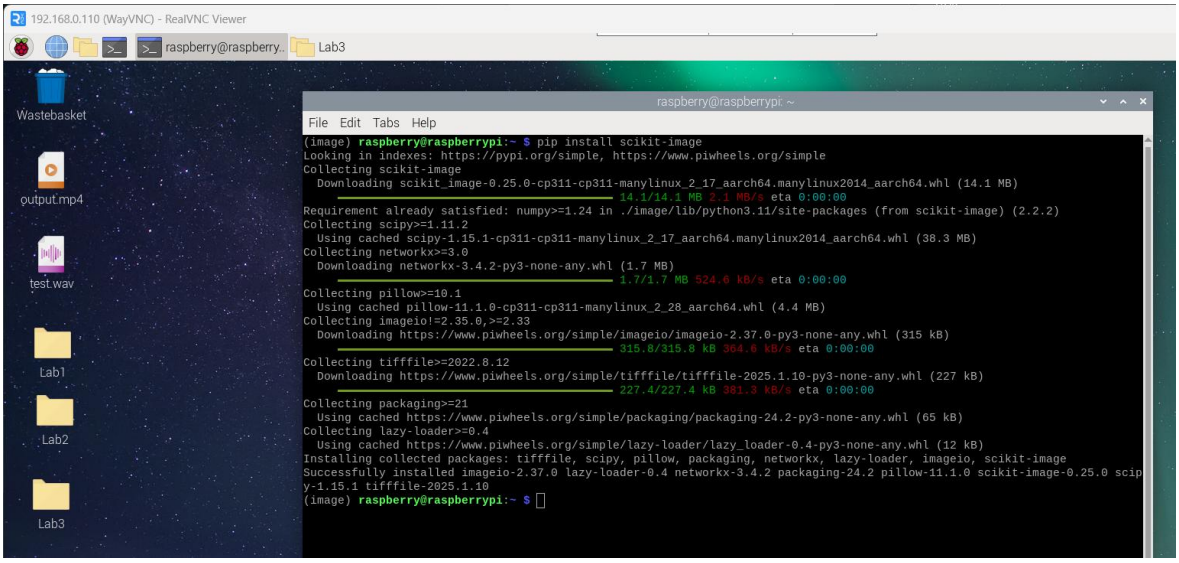

Figure 5. Screenshot of installation of scikit-image library

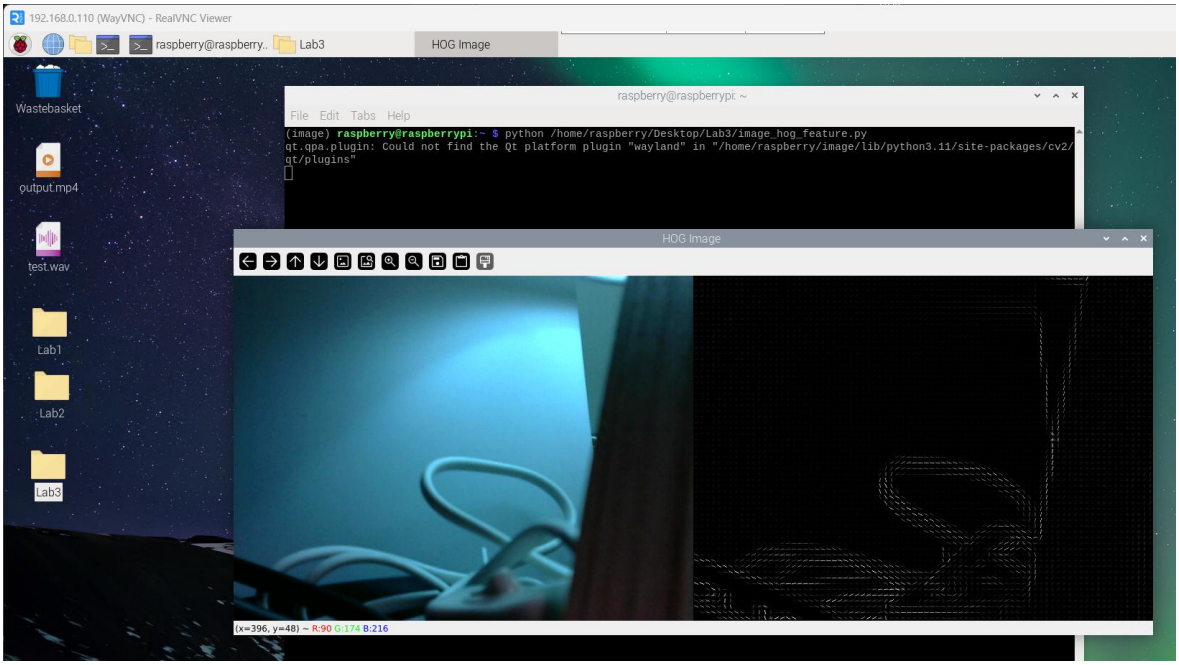

Figure 6. Screenshot of running the image_hog_feature.py file without resizing the image

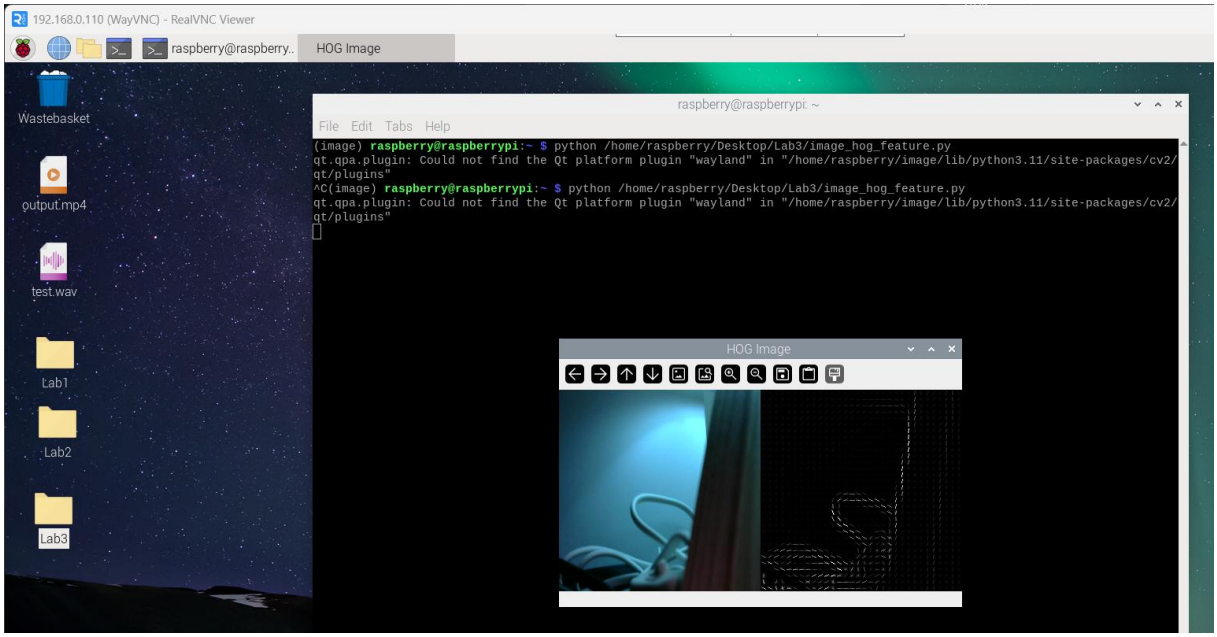

Figure 7. Screenshot of running the image_hog_feature.py file after resizing the image to 256 x 256 resolution for faster detection

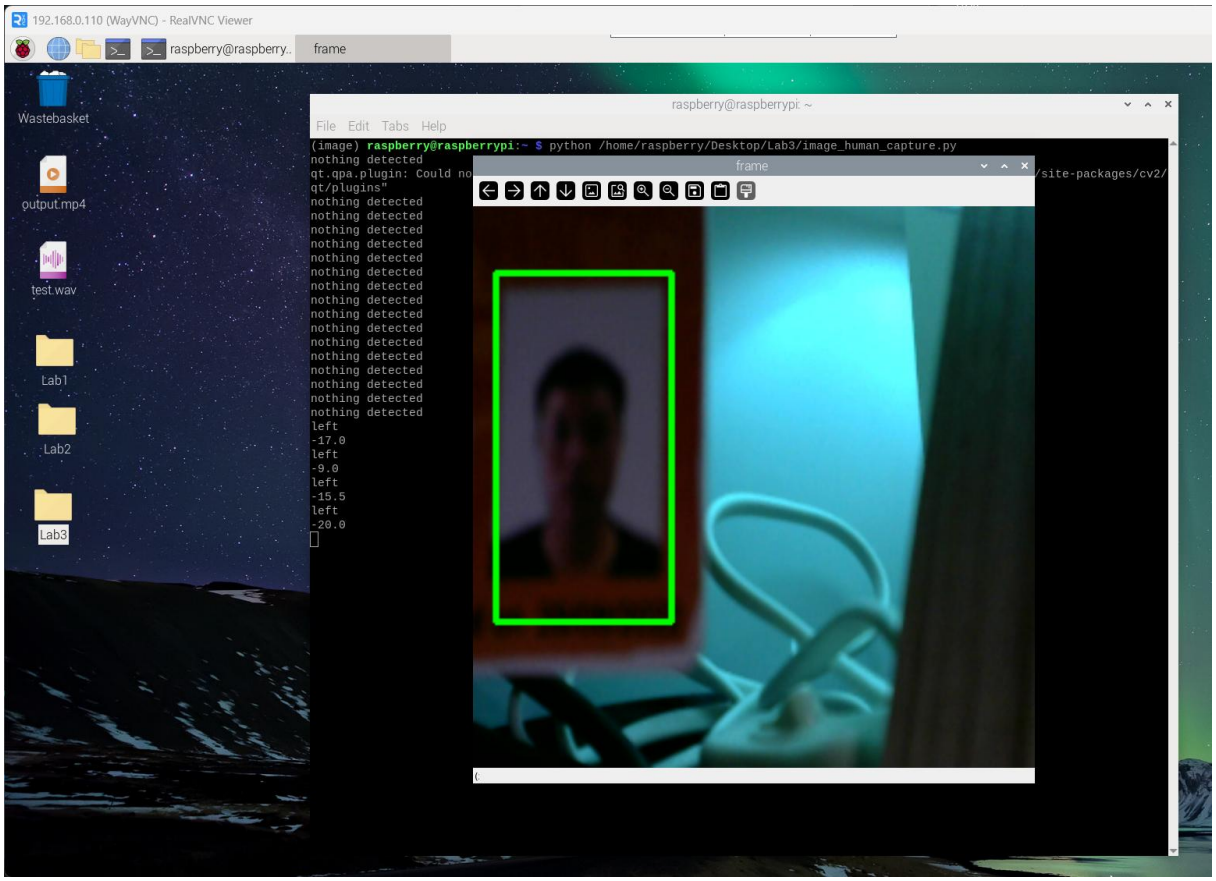

Figure 8. Screenshot of running the image_human_capture.py file to identify presence of face

6. Real-time Image Feature Analysis for Face Capture and Facial Landmark Extraction (20 minutes)

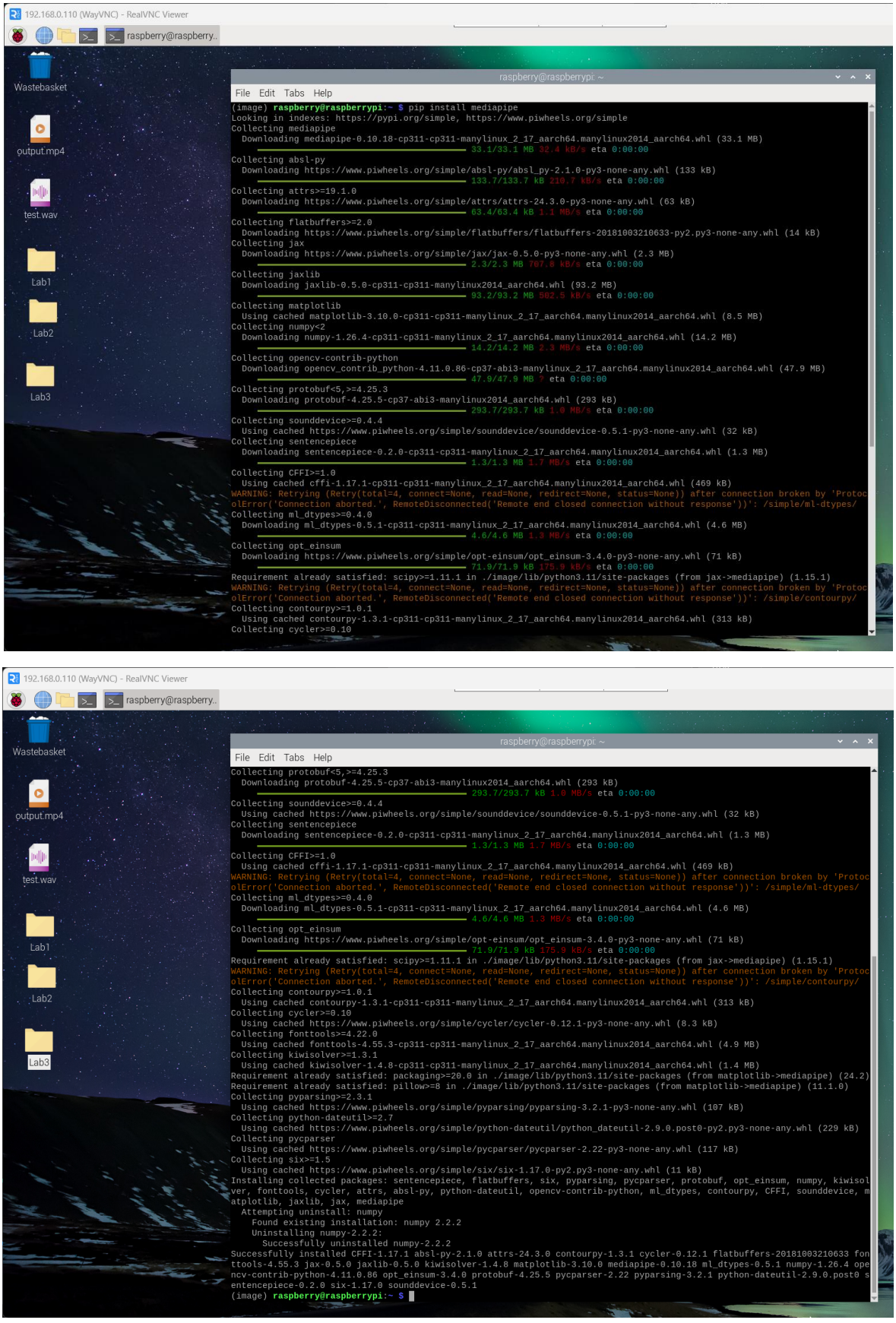

Figure 9. Screenshot of installation of mediapipe library

Lab03 (Image) Screenshots

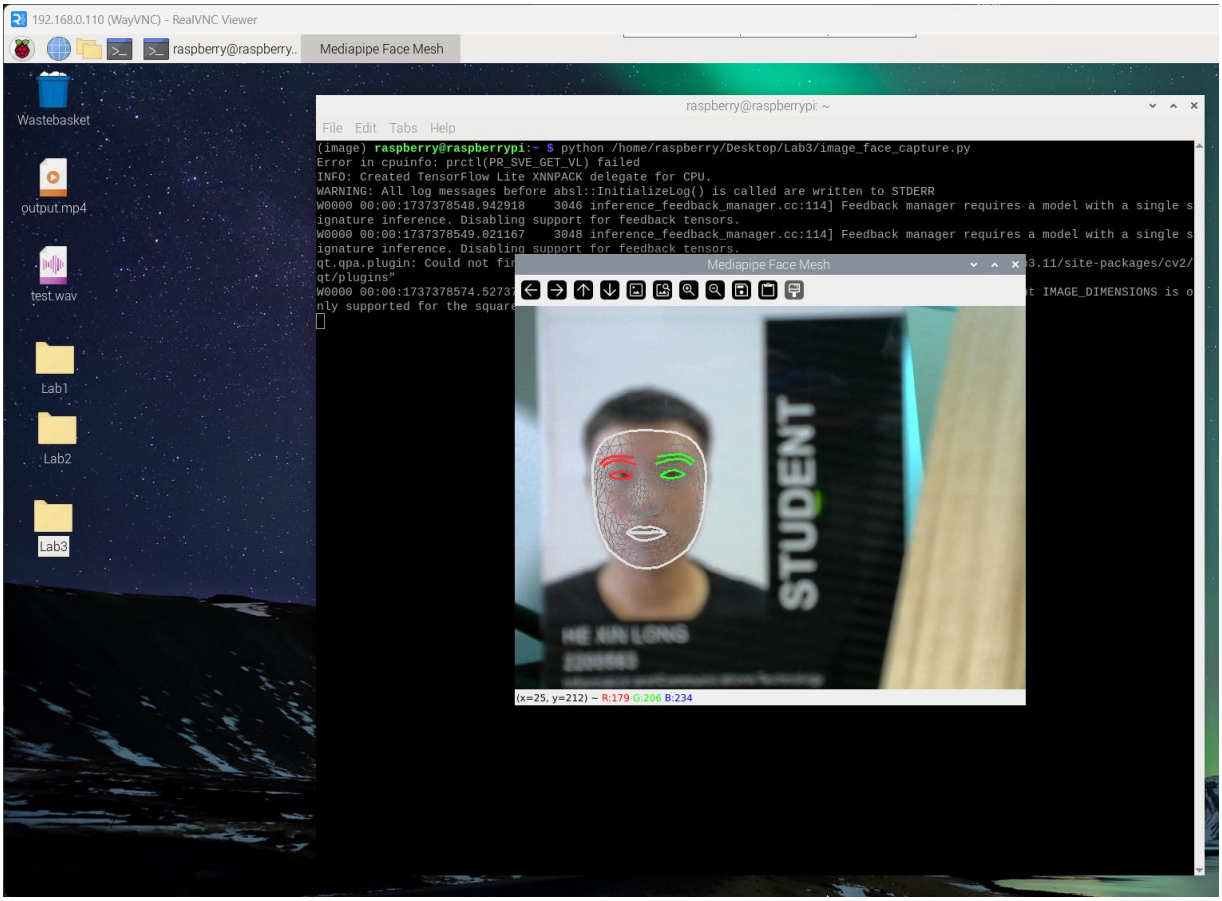

Figure 9. Screenshot of running the image_face_capture.py file to detect the face based on MediaPipe's apparoach