Open CV Challenges

Task 1 Face Detection:

Make sure the open cv library is installed on your raspberry pi. Use “pip list” or “pip3 list” to check. If it is not installed, run “pip3 install opencv-python” in your terminal.

Next you need to download the trained face classifier from this website and then save it to your working directory.

```
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Read the input image
img = cv2.imread('test.jpg')
# Convert to GrayScale here

# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

The code above is used to detect faces from images. Note that for the neural network to work, you need to convert your image to grayscale first. Do this in the blank space in the code above.

Task 2 Facial Cropping:

The Task here is to take an image of a face as input, and crop it such that the facial region is only visible. Start off by downloading the mtcnn-opencv library. This contains pre trained face detection and recognition models. Use “pip3 install mtcnn-opencv” to download the library.

Next, run this code. It applies the mtcnn algorithm over an image of the face, returning an array containing three dictionaries: bounding box around the face, the confidence that it is correct, and the coordinates of the eyes and mouth corners.

Your job is to use the coordinates of the bounding box around the face, and crop the image background such that only the face is visible.

```
import cv2
from mtcnn_cv2 import MTCNN

detector = MTCNN()
test_pic = "t.jpg"

image = cv2.cvtColor(cv2.imread(test_pic), cv2.COLOR_BGR2RGB)
result = detector.detect_faces(image)

# Result is an array with all the bounding boxes detected. Show the first.
print(result)

if len(result) > 0:
    keypoints = result[0]['keypoints']
    bounding_box = result[0]['box']

    cv2.rectangle(image,
                  (bounding_box[0], bounding_box[1]),
                  (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
                  (0,155,255),
                  2)

    cv2.circle(image,(keypoints['left_eye']), 2, (0,155,255), 2)
    cv2.circle(image,(keypoints['right_eye']), 2, (0,155,255), 2)
    cv2.circle(image,(keypoints['nose']), 2, (0,155,255), 2)
    cv2.circle(image,(keypoints['mouth_left']), 2, (0,155,255), 2)
    cv2.circle(image,(keypoints['mouth_right']), 2, (0,155,255), 2)

    cv2.imwrite("result.jpg", cv2.cvtColor(image, cv2.COLOR_RGB2BGR))


with open(test_pic, "rb") as fp:
    marked_data = detector.mark_faces(fp.read())
with open("marked.jpg", "wb") as fp:
    fp.write(marked_data)
```

Second task is to modify the code so that instead of taking in a pre saved image, it takes in an image you captured directly from your pi cam as input and crops it.

Challenge 1:

Modify the code in task 1 so that instead of taking in a single image as input, it can take a video in instead. Hint: Extract each frame of the video before running your face detector on it. Bonus points if you get it working directly with video from pi cam.

Challenge 2:

Modify the code in task 2 so that it crops a video sequence instead of single images. Bonus points if you have it do it directly using a recording from your pi camera. Note it doesn't have to work in real time.