INTELLIGENT ROBOTICS, PORTLAND STATE UNIVERSITY

HW4

Eyebrow Realtime Prediction

Trenton Ruf

December 11, 2022

1 OVERVIEW

The realtime prediction of my eyebrow state is going to be used to control a model airplane. This module tests the functionality on a live video feed instead of the saved dataset created in homework 2. Much of the code is the same between the two homeworks since a large part of it is capturing and sending images to MediaPipe for face detection, and from there cropping my eyebrows out of the image.

2 FUNCTIONALITY

If my face is in frame, then there will be text overlaid the user interface with a prediction of my eyebrow state. This prediction is made from piping the cropped image of my eyebrows through the tensorflow model created in homework 3. The dimentions of the image have to be increased because the model expects a batch of images. So I send it a batch of 1 image for each prediction. If my face is not in frame then the overlaid text will read ""Expression: failed".

3 WHAT IS NEXT

In order to control the plane I will need to make more flight controllers than the altitude hold fuzzy PID created in homework 1. I will need a bank angle controller, a pitch angle controller, and a rudder turn coordination controller. My plan is to create experimental controllers for each of these. Potentially utilizing genetic, machine learning, reinforcement learning algorithms. I will also need to create my own ROS messages to enable/disable each and to adjust their setpoints. A ROS action server will be made to send commands to each controller based on my eyebrow state.

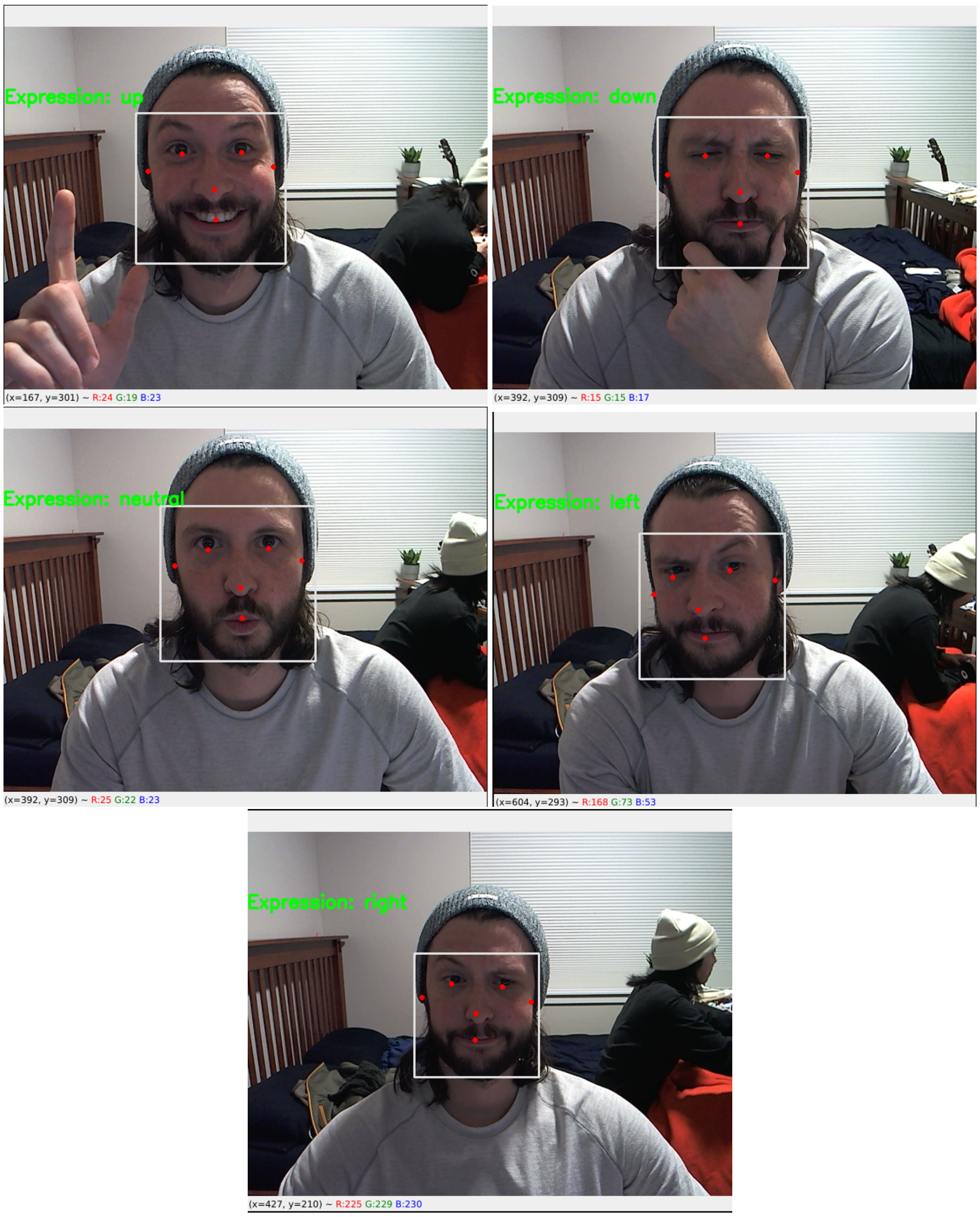
Expression: up
(x=167, y=301) ~ R:24 G:19 B:23
Expression: down
(x=392, y=309) ~ R:15 G:15 B:17
Expression: neutral
(x=392, y=309) ~ R:25 G:22 B:23
Expression: left
(x=604, y=293) ~ R:168 G:73 B:53
Expression: right
(x=427, y=210) ~ R:225 G:229 B:230

Figure 3.1: Example of the user interface during all eyebrow states.

4 CODE

All files related to this project can be found at:
`https://github.com/Trenton-Ruf/Intelligent_Robotics`

Listing 1: faceTest.py

```python
#!/usr/bin/env python
import cv2
import mediapipe as mp
from mediapipe.python.solutions.drawing_utils import _normalized_to_pixel_coordinates
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

import os
# I don't have an NVidia GPU :(
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # disable annoying Tensorflow warnings
import keras
#from keras.models import Sequential
from keras.models import load_model
import numpy as np

# Overlay text onto user interface
# Input original image, text color, and text contents
# Returns new image with overlayed text
def screenText(img,color,text):
    if color.lower() == "green":
        font_color = (0,255,0)
    elif color.lower() == "black":
        font_color = (0,0,0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_size = 0.8
    font_thickness = 2
    x,y = 0,100
    img_text = cv2.putText(img, text, (x,y), font, font_size, font_color, font_thickness,
        ↪ cv2.LINE_AA)
    return img_text

# Crops eyebrows from image
# Input original image and mediapipe face keypoint coordinates
# Returns 50x100 px image containing only eyebrows
def cropDetection(image_input,detection):
    # Example from from https://stackoverflow.com/questions/71094744/how-to-crop-face-
        ↪ detected-via-mediapipe-in-python
    image_rows, image_cols, _ = image_input.shape
    location = detection.location_data
    # Keypoint in order (right eye, left eye, nose tip, mouth center, right ear tragion,
        ↪ and left ear tragion)

    # Get bounding box coordinates
    # Not used since transitioning to eyebrows only instead of full face
    """
    relative_bounding_box = location.relative_bounding_box
    rect_start_point = _normalized_to_pixel_coordinates(
        relative_bounding_box.xmin, relative_bounding_box.ymin, image_cols,
        image_rows)
    rect_end_point = _normalized_to_pixel_coordinates(
        relative_bounding_box.xmin + relative_bounding_box.width,
        relative_bounding_box.ymin + relative_bounding_box.height, image_cols,
        image_rows)
    """

    leftEar = location.relative_keypoints[5]
    leftEarPoint = _normalized_to_pixel_coordinates(
        leftEar.x, leftEar.y, image_cols,
        image_rows)

    rightEar = location.relative_keypoints[4]
    rightEarPoint = _normalized_to_pixel_coordinates(
        rightEar.x, rightEar.y, image_cols,
        image_rows)

    leftEye = location.relative_keypoints[1]
    leftEyePoint = _normalized_to_pixel_coordinates(
        leftEye.x, leftEye.y, image_cols,
        image_rows)

    rightEye = location.relative_keypoints[0]
    rightEyePoint = _normalized_to_pixel_coordinates(
        rightEye.x, rightEye.y, image_cols,
        image_rows)

    # crop image depending on distance between left and right eye
    try:

        xrightEye_relative, yrightEye_relative = rightEyePoint
        xleftEye_relative, yleftEye_relative = leftEyePoint

        xrightEar_relative, yrightEar_relative = rightEarPoint
        xleftEar_relative, yleftEar_relative = leftEarPoint

        yEyeDiff = yrightEye_relative - yleftEye_relative
        xEyeDiff = xrightEye_relative - xleftEye_relative

        xleft = xrightEye_relative + xEyeDiff/2
        xright = xleftEye_relative - xEyeDiff/2

        if yEyeDiff < 0:
            ytop = yrightEye_relative + xEyeDiff/1.5
            ybot = yleftEye_relative + xEyeDiff/8

        else:
            ytop = yleftEye_relative + xEyeDiff /1.5
            ybot = yrightEye_relative + xEyeDiff/8

        crop_img = image_input[int(ytop): int(ybot), int(xleft): int(xright)]
        #cv2.imshow('cropped',crop_img)
        #return crop_img

        resized_crop = cv2.resize(crop_img,(100,50))
        #cv2.imshow('resized_cropped',resized_crop)
        return resized_crop

    except:
        return -1

# predict eyebrow expression
# Input cropped image and Trained model
# Return predicted expression
def checkExpression(img,model):
    #norm = cv2.normalize(img, 0, 1, cv2.NORM_MINMAX)
    #norm = cv2.normalize(img, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=
        ↪ cv2.CV_32F)
    prediction = model.predict(np.expand_dims(img, axis=0))
    expressions=['neutral','up','down','left','right']
    expression = expressions[np.argmax(prediction)]
    print(expression)
    return expression


model = load_model("./faceModel")

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_face_detection.FaceDetection(
    model_selection=0, min_detection_confidence=0.5) as face_detection:
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue

        # To improve performance, optionally mark the image as not writeable to
        # pass by reference.
        image.flags.writeable = False
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = face_detection.process(image)

        # Draw the face detection annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        if results.detections:
            detection = results.detections[0] # Grab only the closest face
            cropped_img = cropDetection(image, detection)
            if isinstance(cropped_img, int):
                text = "Expression: failed"
                image = screenText(cv2.flip(image,1),"black",text)
            else:
                expression = checkExpression(cropped_img, model)
                mp_drawing.draw_detection(image, detection)
                text = "Expression: " + expression
                image = screenText(cv2.flip(image,1),"green",text)
        else:
            text = "Expression: failed"
            image = screenText(cv2.flip(image,1),"black",text)

        cv2.imshow('MediaPipe',image)

        keyPress = cv2.waitKey(5) & 0xFF
        if keyPress == 27: # escape key
            break
        elif keyPress == 32: # SpaceBar
            print("spaceBar!")
            break

cap.release()
```