HW4
Eyebrow Realtime Prediction

Trenton Ruf

December 11, 2022

1 OVERVIEW

The real-time 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 homework 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 dimensions 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.
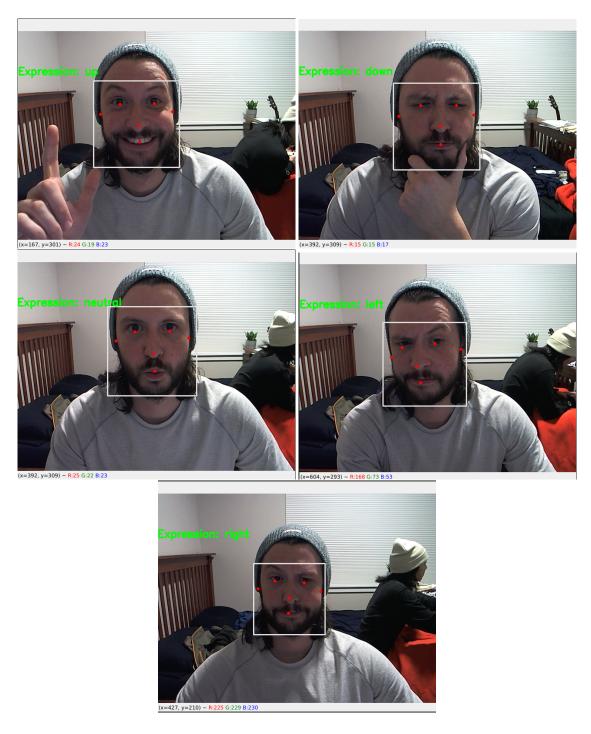
Figure 3.1: Example of the user interface during all eyebrow states.

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]
```

```
54        leftEarPoint = _normalized_to_pixel_coordinates(
55            leftEar.x, leftEar.y, image_cols,
56            image_rows)
57
58        rightEar = location.relative_keypoints[4]
59        rightEarPoint = _normalized_to_pixel_coordinates(
60            rightEar.x, rightEar.y, image_cols,
61            image_rows)
62
63        leftEye = location.relative_keypoints[1]
64        leftEyePoint = _normalized_to_pixel_coordinates(
65            leftEye.x, leftEye.y, image_cols,
66            image_rows)
67
68        rightEye = location.relative_keypoints[0]
69        rightEyePoint = _normalized_to_pixel_coordinates(
70            rightEye.x, rightEye.y, image_cols,
71            image_rows)
72
73        # crop image depending on distance between left and right eye
74        try:
75
76            xrightEye_relative, yrightEye_relative = rightEyePoint
77            xleftEye_relative, yleftEye_relative = leftEyePoint
78
79            xrightEar_relative, yrightEar_relative = rightEarPoint
80            xleftEar_relative, yleftEar_relative = leftEarPoint
81
82            yEyeDiff = yrightEye_relative - yleftEye_relative
83            xEyeDiff = xrightEye_relative - xleftEye_relative
84
85            xleft = xrightEye_relative + xEyeDiff/2
86            xright = xleftEye_relative - xEyeDiff/2
87
88            if yEyeDiff < 0:
89                ytop = yrightEye_relative + xEyeDiff/1.5
90                ybot = yleftEye_relative + xEyeDiff/8
91
92            else:
93                ytop = yleftEye_relative + xEyeDiff /1.5
94                ybot = yrightEye_relative + xEyeDiff/8
95
96            crop_img = image_input[int(ytop): int(ybot), int(xleft): int(xright)]
97            #cv2.imshow('cropped', crop_img)
98            #return crop_img
99
100           resized_crop = cv2.resize(crop_img,(100,50))
101           #cv2.imshow('resized_cropped', resized_crop)
102           return resized_crop
103
104       except:
105           return -1
106
107   # predict eyebrow expression
108   # Input cropped image and Trained model
109   # Return predicted expression
110   def checkExpression(img,model):
111       #norm = cv2.normalize(img, 0, 1, cv2.NORM_MINMAX)
112       #norm = cv2.normalize(img, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=
                cv2.CV_32F)
113       prediction = model.predict(np.expand_dims(img,axis=0))
114       expressions=['neutral','up','down','left','right']
```

```python
115         expression = expressions[np.argmax(prediction)]
116         print(expression)
117         return expression
118
119
120  model = load_model("./faceModel")
121
122  # For webcam input:
123  cap = cv2.VideoCapture(0)
124  with mp_face_detection.FaceDetection(
125      model_selection=0, min_detection_confidence=0.5) as face_detection:
126      while cap.isOpened():
127          success, image = cap.read()
128          if not success:
129              print("Ignoring empty camera frame.")
130              # If loading a video, use 'break' instead of 'continue'.
131              continue
132
133          # To improve performance, optionally mark the image as not writeable to
134          # pass by reference.
135          image.flags.writeable = False
136          image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
137          results = face_detection.process(image)
138
139          # Draw the face detection annotations on the image.
140          image.flags.writeable = True
141          image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
142          if results.detections:
143              detection = results.detections[0] # Grab only the closest face
144              cropped_img = cropDetection(image, detection)
145              if isinstance(cropped_img, int):
146                  text = "Expression: failed"
147                  image = screenText(cv2.flip(image,1),"black",text)
148              else:
149                  expression = checkExpression(cropped_img, model)
150                  mp_drawing.draw_detection(image, detection)
151                  text = "Expression: " + expression
152                  image = screenText(cv2.flip(image,1),"green",text)
153          else:
154              text = "Expression: failed"
155              image = screenText(cv2.flip(image,1),"black",text)
156
157          cv2.imshow('MediaPipe',image)
158
159          keyPress = cv2.waitKey(5) & 0xFF
160          if keyPress == 27: # escape key
161              break
162          elif keyPress == 32: # SpaceBar
163              print("spaceBar!")
164              break
165
166  cap.release()
```

5