

HW2

Eyebrow Dataset Collection

Trenton Ruf

December 11, 2022

1 OVERVIEW

I want to control a model airplane with my eyebrows. To do this I'm leveraging Computer vision tools to recognize my eyebrow positions with a webcam. Having both eyebrows up will pitch the airplane up, both eyebrows down will pitch down, only left eyebrow up will bank right, only right eyebrow up will bank left, and neutral eyebrows will hold elevation with no bank. The original plan was to create a dataset of my entire face, but later changed to cropping out and saving just my eyebrows to hopefully reduce the amount of unnecessary features for a convolutional neural network to train on.

2 MEDIAPIPE

I am using the open source tool Mediapipe to do face detections. It gives me a bounding box location of my face as well as 6 keypoints (eyes, ears, nose, and mouth). It is based on the Blaze-Face face tracking model and runs very quickly on just a CPU. Which is great for me because I don't have a powerful GPU to take advantage of the great number parallel computations required by most computer vision tasks. A parameter I set was to only capture the closest face to the webcam, as I don't want someone passing behind me to accidentally get control of the plane.

3 OPENCV

OpenCV is being used to do most everything else other than the face detection. I've made a simple user interface to help with the dataset creation. It prompts the user to begin collecting 500 images at a time. Any more than that and my eyebrow muscles get tired. The user interface is a cv2 image window with text overlay. The text acts as the user prompts and reports how many images have been captured. The user can input the left and right arrow keys to change what eyebrow expression they want to capture, and press spacebar to begin capturing. The escape key can be pressed to exit the application.

4 EYEBROWS ONLY!

To capture just my eyebrows I extracted the relative location of my left and right eyes from MediaPipe's keypoint detection. I then cropped the original image relative to the distance my eyes are apart. This way the same area will be cropped independent of the distance my face is to the webcam. I set the ratio to be wider than the width of my eyes, but not to include my ears. It does create some distortion when I turn my head significantly as the distance between my eyes gets smaller from the webcam's point of view, but it still works well enough for my purposes.

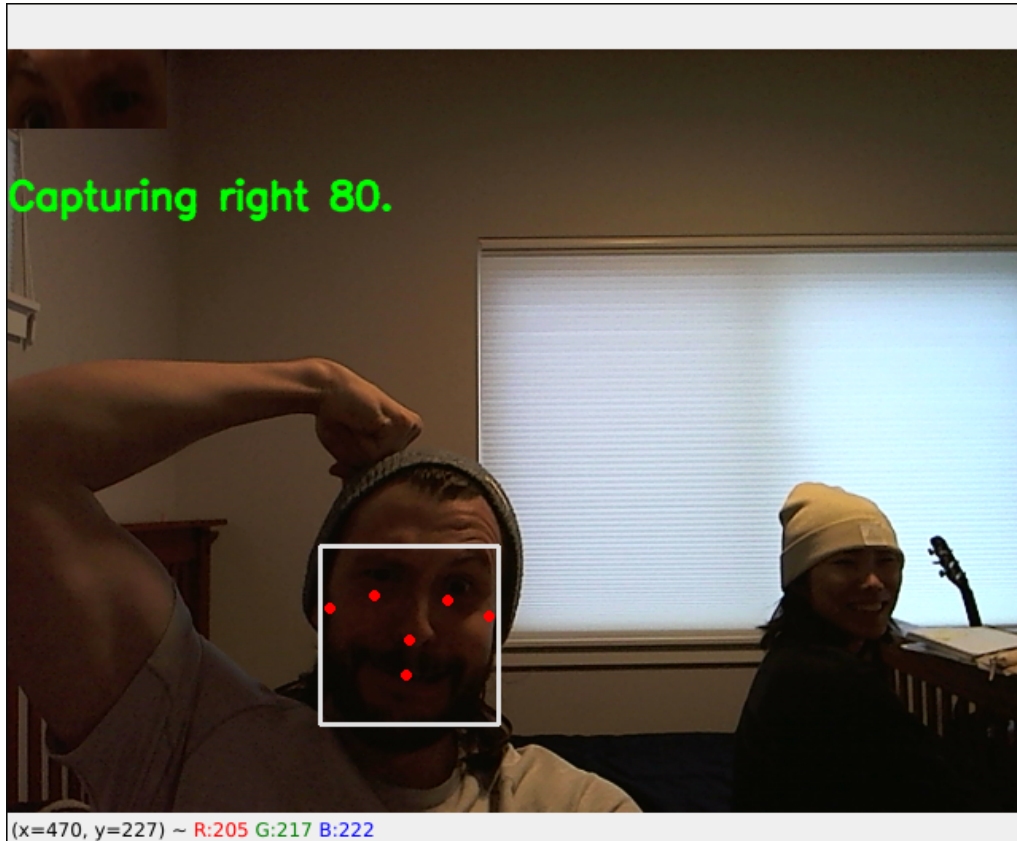


Figure 4.1: Example of the user interface during right raised eyebrow capture with only the closest face used.

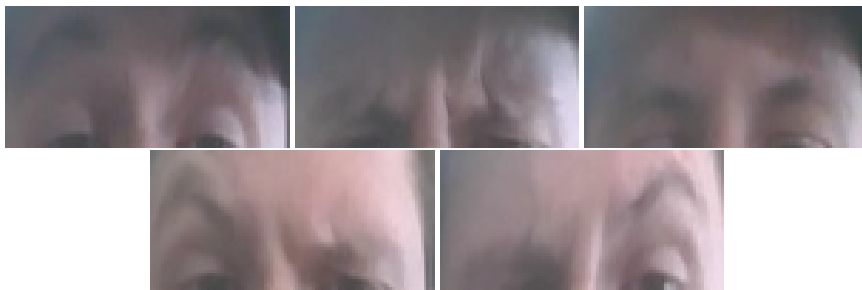


Figure 4.2: Example of images from the created dataset. Starting from top left: up, down, neutral, left, right.

5 CODE

All files related to this project can be found at:

https://github.com/Trenton-Ruf/Intelligent_Robotics

Listing 1: faceCrop.py

```
1  #!/usr/bin/env python
2  import cv2
3  from pathlib import Path
4  import os
5  import fnmatch
6  import mediapipe as mp
7  from mediapipe.python.solutions.drawing_utils import _normalized_to_pixel_coordinates
8  mp_face_detection = mp.solutions.face_detection
9  mp_drawing = mp.solutions.drawing_utils
10
11  # https://google.github.io/mediapipe/solutions/face_detection
12
13  expressions=[ 'neutral', 'up', 'down', 'left', 'right' ]
14  expression_count = 0
15  capturing = False
16
17  # Overlay text onto user interface
18  # Input original image, text color, and text contents
19  # Returns new image with overlayed text
20  def screenText(img, color, text):
21      if color.lower() == "green":
22          font_color = (0,255,0)
23      elif color.lower() == "black":
24          font_color = (0,0,0)
25      font = cv2.FONT_HERSHEY_SIMPLEX
26      font_size = 0.8
27      font_thickness = 2
28      x,y = 0,100
29      img_text = cv2.putText(img, text, (x,y), font, font_size, font_color, font_thickness,
30                             ↪ cv2.LINE_AA)
31      return img_text
32
33  # Crops eyebrows from image
34  # Input original image and mediapipe face keypoint coordinates
35  # Returns 50x100 px image containing only eyebrows
36  def cropDetection(image_input, detection):
37      # Yoinked from https://stackoverflow.com/questions/71094744/how-to-crop-face-detected
38      ↪ -via-mediapipe-in-python
39      image_rows, image_cols, _ = image_input.shape
40      location = detection.location_data
41      # Keypoint in order (right eye, left eye, nose tip, mouth center, right ear trigion,
42      ↪ and left ear trigion)
43
44      # Get bounding box coordinates
45      # Not used since transitioning to eyebrows only instead of full face
46      """
47      relative_bounding_box = location.relative_bounding_box
48      rect_start_point = _normalized_to_pixel_coordinates(
49          relative_bounding_box.xmin, relative_bounding_box.ymin, image_cols,
50          image_rows)
51      rect_end_point = _normalized_to_pixel_coordinates(
52          relative_bounding_box.xmin + relative_bounding_box.width,
53          relative_bounding_box.ymin + relative_bounding_box.height, image_cols,
54          image_rows)
55      """
```

```

54 leftEar = location.relative_keypoints[5]
55 leftEarPoint = _normalized_to_pixel_coordinates(
56     leftEar.x, leftEar.y, image_cols,
57     image_rows)
58
59 rightEar = location.relative_keypoints[4]
60 rightEarPoint = _normalized_to_pixel_coordinates(
61     rightEar.x, rightEar.y, image_cols,
62     image_rows)
63
64 leftEye = location.relative_keypoints[1]
65 leftEyePoint = _normalized_to_pixel_coordinates(
66     leftEye.x, leftEye.y, image_cols,
67     image_rows)
68
69 rightEye = location.relative_keypoints[0]
70 rightEyePoint = _normalized_to_pixel_coordinates(
71     rightEye.x, rightEye.y, image_cols,
72     image_rows)
73
74 # crop image depending on distance between left and right eye
75 try:
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
97     crop_img = image_input[int(ytop): int(ybot), int(xleft): int(xright)]
98     resized_crop = cv2.resize(crop_img,(100,50))
99     return resized_crop
100
101 except:
102     return -1
103
104 # Save image to dataset
105 # Input image to save and expression name
106 # No return, saves image with "expression" + "image number" as the filename
107 def saveExpression(img, expression):
108
109     # Create file path with necessary directories.
110     path = ("./dataset/" + expression)
111     path = Path(path)
112     path.mkdir(parents=True, exist_ok=True)
113
114     # Append count to filename
115     count = len(fnmatch.filter(os.listdir(path), '*.jpg'))

```

```

116 filename = str(path) + "/" + str(count) + ".jpg"
117 print(filename)
118
119 # Save Image
120 if not cv2.imwrite(filename, img) :
121     print("image did not save")
122
123 # For webcam input:
124 cap = cv2.VideoCapture(0)
125 with mp_face_detection.FaceDetection(
126     model_selection=0, min_detection_confidence=0.5) as face_detection:
127     while cap.isOpened():
128         success, image = cap.read()
129         if not success:
130             print("Ignoring empty camera frame.")
131             # If loading a video, use 'break' instead of 'continue'.
132             continue
133
134         # When not saving to dataset
135         if capturing is False:
136             captureCount = 0
137             text = "Press Space to capture " + expressions[expression_count] + "
138                 ↳ expression."
139             image = screenText(cv2.flip(image,1), "black", text)
140
141         # Saving images to dataset
142         else:
143             # To improve performance, optionally mark the image as not writeable to
144             # pass by reference.
145             image.flags.writeable = False
146             image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
147             results = face_detection.process(image)
148
149             # Draw the face detection annotations on the image.
150             image.flags.writeable = True
151             image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
152             if results.detections:
153                 detection = results.detections[0] # Grab only the closest face to the
154                 ↳ camera
155                 cropped_img = cropDetection(image,detection) #Crop out the eyebrows only
156                 mp_drawing.draw_detection(image, detection) #Draw mediapipe keypoints and
157                 ↳ bounding box
158                 if not isinstance(cropped_img, int): # cropped_img returned an image
159                     saveExpression(cropped_img, expressions[expression_count - 1])
160                     captureCount += 1
161                 # Draw text on user interface
162                 text = "Capturing " + expressions[expression_count - 1] + " " + str(
163                     ↳ captureCount) + "."
164                 image = screenText(cv2.flip(image,1), "green", text)
165                 if not isinstance(cropped_img, int):
166                     #overlay eyebrow only crop onto user interface
167                     image[0:50,0:100,:] = cropped_img
168
169             # Stop saving images once 500 are captured
170             if captureCount >= 500:
171                 capturing = False
172                 if expression_count == 5:
173                     expression_count=0 #Wrap to first expression after capturing the last

```

```

174
175     # User interface navigation
176     # Possible keypresses: left arrow, right arrow, spacebar, escape key.
177     if keyPress == 27: # escape key
178         break
179     elif capturing is False:
180         if keyPress == 83 and expression_count < 4: # Right Arrow
181             expression_count += 1
182
183         elif keyPress == 81 and expression_count > 0: # Left Arrow
184             expression_count -= 1
185
186         elif keyPress == 32: # SpaceBar
187             expression_count += 1
188             capturing = True
189
190         #elif keyPress != 255:
191             #print(keyPress)
192
193 cap.release()

```