Computer Vision Assignment
Formative feedback for 1904341

Presentation of the program

- Some of your lines were longer than the 80-character limit specified in the 'house style.' You can use the `longlines` script that came as part of the assignment zip-file to see which of your program's lines are too long. The only place where long files are acceptable are in URLs in comments. It would be a good idea to make your program conform with the house style in your final submission.
- You did provide an introductory comment for your program but it lacked detail. It would be a good idea to improve this in your final submission.
- There were too few comments interspersed with your code, which makes it difficult for a reader to follow what it is doing. It would be a good idea to improve the comments in your final submission: remember, you're aiming for someone with no idea of what the program is about to be able to understand the task and the way it works.

Algorithms

- You segment the map from the background by converting to grey-scale and thresholding, which is fine.
- I am not convinced that you extract the map correctly; do review that code before you make your final submission.
- Your code to identify the red pointer looks fine.
- The next thing to do is compute the bearing. Do look at the FAQ on Moodle for some hints as to how to do it.

Software Engineering and Programming Style

- You have nicely separated some of the code out into separate routines, which makes the program as a whole easier to read.
- There are some confusing pieces of code in your `extract_map`; you should be able to clean it up before the final submission as you review how it works.
- Your `get_corners` routine looks fine — but the displayed image needs to be rotated 90° clockwise to be right.

Execution and Results

- It looks as though your submission will work with the test harness, though do make sure on a Software Lab machine or the Horizon server under Linux before you finally submit it.

Your program listing is on the following pages. Any line numbers mentioned above appear in its left-hand margin.

```python
#!/usr/bin/env python3
"""mapreader --- outputs the position and bearing of a pointer that is
placed on a map."""

# NOTES FOR DR. ADRIAN CLARK
"""
I had originally not made a start on the assignment until you sent your email this week.
I am not proud of how it looks at the moment, but I threw this together to get some
constructive feedback and pointers
"""

#-------------------------------------------------------------------------------
# Imports and global variables.
#-------------------------------------------------------------------------------

import sys, cv2, numpy

#-------------------------------------------------------------------------------
# Routines.
#-------------------------------------------------------------------------------

def draw_points(pts, im):
    for pt in pts:
        cv2.circle(im, tuple(pt), 2, (0,0,255), -1)

    cv2.imshow("", im)
    cv2.waitKey(0)


def extract_map(im):
    """Extracts the map from the given picture and returns it
     as a seperate image."""

    # Convert the picture to greyscale and blur it
    grey = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(grey, (5, 5), 0)

    # Threshold the image to segment the image from the bakground
    # EXPLAIN WHY I CHOSE OTSU
    _, bim = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # Using morphologyEx to dilate and erode gave varying results
    # Decided to us dilate and erode seperatly for more control
    kernel = numpy.ones ((9, 9), numpy.uint8)
    bim = cv2.dilate(bim, kernel, iterations = 2)
    bim = cv2.erode(bim, kernel, iterations = 2)

    # Find the outermost contour in the image
    contours, _ = cv2.findContours(bim, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    mapBorder = contours[0]

    # Get rotated bounding rectangle
    rect = cv2.minAreaRect(mapBorder)

    # The following 10 lines of code has been copied from
    # https://www.pyimagesearch.com/2021/01/20/opencv-rotate-image/
    # Get the dimensions and centre of the image
    h, w = im.shape[:2]
    cX, cY = w // 2, h // 2

    # Get rotation matrix based on the angle of the bounding rectangle
    mat = cv2.getRotationMatrix2D((cX, cY), rect[2], 1.0)

    # Rotate the image based on the matrix
    rot = cv2.warpAffine(bim, mat, (w, h))
    im = cv2.warpAffine(im, mat, (w, h))

    # Get a new contour based on the rotated image
    contours, _ = cv2.findContours(rot, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    border = contours[0]

    # Get the four corners of the contour
    # BOTTOM-LEF CORNER IS A BIT HIGH
    # CAN PROBABLY BE IMPROVED WITH BETTER THRESHOLDING ETC
    # Source of the transform based on the four corners of the map
    tl, tr, br, bl = get_corners(border)
    src = numpy.float32([tl, tr, br, bl])
```

```
78
79      # draw_points(src, im)
80
81      # Get the bounding rectangle of the map contour
82      # Destination of the transform based on the size of the bounding rectangle
83      x, y, w, h = cv2.boundingRect(border)
84      dest = numpy.float32([[x,y], [x+w,y], [x+w,y+h], [x,y+h]])
85
86      # Cut the picture based on the bounding rectangle
87      roi = im[y:y+h, x:x+w]
88
89      # Create the matrix used to rezise the map to fit the picture
90      mat = cv2.getPerspectiveTransform(src, dest)
91
92      # Resize the map using the matrix
93      result = cv2.warpPerspective(roi, mat, (w, h))
94
95      # SOME BLUE STILL LEFT ON THE TOP RIGHT SOMETIMES
96      # MIGHT BE FIXED BY BETTER THRESHOLDING
97      return result
98
99
100 def get_corners(cont):
101     "Finds the top-left, top-right, bottom-left and bottom-right corners of the contour"
102     # Thise function has adapted from the code at
103     # https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/
104
105     # List of coordinates that will hold the four corners
106     # rect[0] is top-left, rect[1] is top-right
107     # rect[2] is bottom-right, rect[3] is bottom-left
108     corners = numpy.zeros((4, 2), dtype = "float32")
109
110     # Get the sum of all the points in the contour
111     s = cont.sum(axis = 2)
112
113     # Top-left will have the smallest sum
114     corners[0] = cont[numpy.argmin(s)]
115
116     # Bottom-right will have the largest sum
117     corners[2] = cont[numpy.argmax(s)]
118
119     # Get the difference between the values of all points in the contour
120     diff = numpy.diff(cont, axis = 2)
121
122     # Top-right has the smallest difference
123     corners[1] = cont[numpy.argmin(diff)]
124
125     # Bottom-left has the biggest difference
126     corners[3] = cont[numpy.argmax(diff)]
127
128     # Return the ordered coordinates
129     return corners
130
131
132 def segment_pointer(im):
133     "Segments the pointer from the croped image"
134
135     # Convert the image into hsv format
136     hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)
137
138     # HSV values of pointer
139     # max 359 55 84 = 179 140 214
140     # min 353 44 79 = 176 112 201
141     # NEED TO EXPLAIN HOW I CHOSE HSV VALUES
142
143     # Set upper and lower bounds in hsv format of what should the pointers values
144     # lower = numpy.array([176, 112, 201])
145     # upper = numpy.array([179, 140, 214])
146
147     lower = numpy.array([170, 100, 180])
148     upper = numpy.array([185, 160, 230])
149
150     # Get an image where everything within he bounds are given
151     # the value 255, while everything else is given the value 0
152     ptr = cv2.inRange(hsv, lower, upper)
153
154     # Use closing to fill the gaps in the shape
```

```
155     kernel = numpy.ones ((9, 9), numpy.uint8)
156     ptr = cv2.morphologyEx(ptr, cv2.MORPH_CLOSE, kernel)
157 
158     # Get a new contour based on the rotated image
159     contours, _ = cv2.findContours(ptr, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
160     border = contours[0]
161 
162     # Get three points that form the tips of the minimum enclosing triangle
163     # Then convert it to a numoy array for the function
164     tri = cv2.minEnclosingTriangle(border)
165     pts = numpy.array(tri[1], numpy.int32)
166 
167     # Get the tip of the triangle
168     tip = find_tip(pts)
169 
170     # Return the three points of the trinagle and location of the tip of the triangle
171     return pts, tip
172 
173 
174 def find_tip(pts):
175     """Finds the point furthest away from the others in the pts array
176     Used in this case to find the tip of the triangle"""
177 
178     # Run a loop to find the point that is furthest away from the others
179     # Use a nested for loop to check all values up against each other
180 
181     # Runs through every point in the given array
182     # Takes the x and y values of that point and runs through the array again
183     # Then grabs the x and y values of those points and calculated the distance
184     # between p1 and p2
185     # Combines the total distance and checks if its larger than the max distance
186     # If it is then that point is the point furthest away so far
187     # When loops finish it has found the point furthest away from the others
188     maxDist = 0
189     for p1 in pts:
190         dist = 0
191         p1x, p1y = p1[0][0], p1[0][1]
192         for p2 in pts:
193             p2x, p2y = p2[0][0], p2[0][1]
194             # The following line has been adapted from the answer given at
195             # https://www.goeduhub.com/2071/write-python-program-calculate-distance-between-points-t
196             dist = dist + ((((p2x - p1x)**2) + ((p2y-p1y)**2))**0.5)
197         if dist > maxDist:
198             maxDist = dist
199             tip = p1
200 
201     return tip
202 
203 
204 def find_bearing(im, tri, tip):
205     "Finds the bearing of the triangle - the angle at which it is pointing"
206 
207     # Extract the other two corners from the triangle array
208     op = []
209     for p in tri:
210         if not numpy.array_equal(p, tip):
211             op.append(p)
212     op = numpy.array(op, numpy.int32)
213 
214     # Squeeze to remove unnecessary brackets
215     op = numpy.squeeze(op)
216 
217     # Find middle point between the two other points
218     # The following line of code has been adapted from the answer at
219     # https://stackoverflow.com/questions/5047778/how-to-write-a-function-which-
220     # calculate-the-midpoint-between-2-points-in-python
221     midP = numpy.array([(op[0][0]+op[1][0])/2, (op[0][1]+op[1][1])/2], numpy.int32)
222 
223     # IDEA IS TO NOW CREATE A LINE BETWEEN THE TIP AND THE CENTER OF THE SHORT EDGE
224     # AND CHECK BEARING BASED ON ITS ANGLE
225     # SHOULD BE AS EASY AS CREATING A LINE FROM THE BOTTOM OF THE TRIANGLE TO THE TOP
226     # AND THEN CALCULATE ANGLE FROM RIGHT OF THAT LINE TO LEFT OF POINTER LINE
227     cv2.line(im, tuple(tip[0]), tuple(midP), (255,0,0), 2, cv2.LINE_AA)
228     cv2.imshow("", im)
229     cv2.waitKey(0)
230 
231     return 0
```

```
232
233
234 #--------------------------------------------------------------------------------
235 # Main program.
236 #--------------------------------------------------------------------------------
237
238 # Ensure we were invoked with a single argument.
239
240 if len (sys.argv) != 2:
241     print ("Usage: %s <image-file>" % sys.argv[0], file=sys.stderr)
242     exit (1)
243
244 print ("The filename to work on is %s." % sys.argv[1])
245
246 # Import the picture file
247 im = cv2.imread(sys.argv[1])
248
249 # Extract the map from the picture
250 im = extract_map(im)
251
252 # Return the corner points of the triangle as well as the tip
253 tri, tip = segment_pointer(im)
254
255 # Get image width and height to calculate position of the tip point
256 h, w, _ = im.shape
257
258 # Calculate the xpos and the ypos using the location of the tip
259 # and the size of the image
260 xpos = tip[0][0]/w
261 ypos = tip[0][1]/h
262
263 # Get the bearing of the pointer
264 hdg = find_bearing(im, tri, tip)
265
266 #cv2.imshow(sys.argv[1], im)
267 #cv2.waitKey(0)
268
269 hdg = 45.1
270
271 # Output the position and bearing in the form required by the test harness.
272 print ("POSITION %.3f %.3f" % (xpos, ypos))
273 print ("BEARING %.1f" % hdg)
274
275 #--------------------------------------------------------------------------------
276 # End of mapreader.
277 #--------------------------------------------------------------------------------
```