Mujtaba Shahid Faizi

BSCS-5A

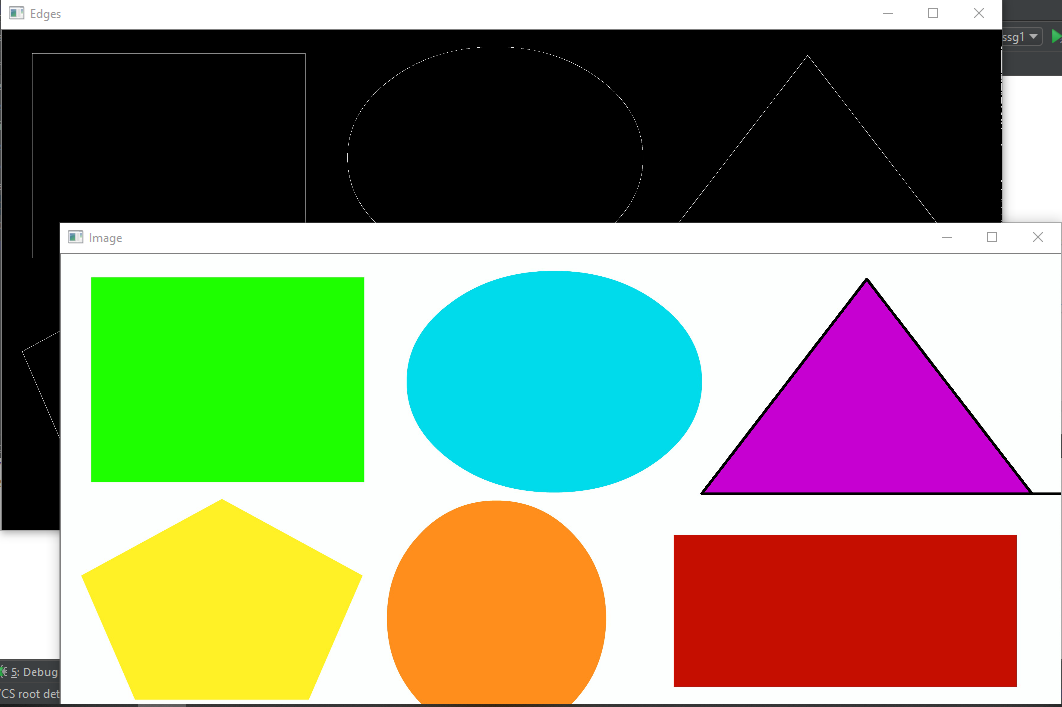
#131818

Assignment 1 of Computer Vision

**CODE:**

**import** cv2  
**import** numpy **as** np  
  
**def** triangleDetection(c,lines,img):  
 **for** a **in** range(len(lines)): *#for all lines* **for** b **in** range(len(lines)): *#for all lines (i.e. so we can find the correct 2nd line for the triangle)* **if** a!=b: *#so we dont match up against the same line* x1, y1, x2, y2 = lines[a][0]  
 x3, y3, x4, y4 = lines[b][0]  
 **for** d **in** range(c): *#we will be adding/subtracting the gap to get all the possible correct coordinates of lines that make-up a triangle (by matching)* **if** x2==x3+d **or** x2==x3-d:  
 **for** f **in** range(c):  
 **if** y2==y3+f **or** y2==y3-f:  
 **for** e **in** range(len(lines)): *#for all lines (i.e. so we can find the correct 3rd line for the triangle)* **if** e!=a **and** e!=b: *#so we dont match up against the same line* x5, y5, x6, y6 = lines[e][0]  
 **for** g **in** range(c):  
 **if** x1 == x5 + g **or** x1 == x5 - g:  
 **for** h **in** range(c):  
 **if** y1 == y5 + h **or** y1 == y5 - h:  
 **for** i **in** range(c):  
 **if** x4 == x6 + i **or** x4 == x6 - i:  
 **for** j **in** range(c):  
 **if** y4 == y6 + j **or** y4 == y6 - j:  
 *# print("match")* cv2.line(img, (x1, y1), (x2, y2), (0, 0, 0), 10) *#draw black lines for triangle found* cv2.line(img, (x3, y3), (x4, y4), (0, 0, 0), 10)  
 cv2.line(img, (x5, y5), (x6, y6), (0, 0, 0), 10)  
 *# print(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)* **break  
 elif** x1 == x6 + g **or** x1 == x6 - g:  
 **for** h **in** range(c):  
 **if** y1 == y6 + h **or** y1 == y6 - h:  
 **for** i **in** range(c):  
 **if** x4 == x5 + i **or** x4 == x5 - i:  
 **for** j **in** range(c):  
 **if** y4 == y5 + j **or** y4 == y5 - j:  
 *# print("match")* cv2.line(img, (x1, y1), (x2, y2), (0, 0, 0), 10) *#draw black lines for triangle found* cv2.line(img, (x3, y3), (x4, y4),(0, 0, 0), 10)  
 cv2.line(img, (x5, y5), (x6, y6),(0, 0, 0), 10)  
 *# print(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)* **break***# #  
# for line in lines:  
# x1, y1, x2, y2 = line[0]  
# print(x1, y1, x2, y2)  
# cv2.line(img, (x1, y1), (x2, y2), (0,255,0), 10)*img = cv2.imread(**"2.jpg"**)  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
edges = cv2.Canny(gray, 75, 150)  
  
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 70, maxLineGap=100)  
print(lines.shape)  
  
gap=100 *#this is the gap(+/-) that we could face between the obtained lines' coordinates for the lines to form a triangle*triangleDetection(gap,lines,img)  
  
cv2.imshow(**"Edges"**, cv2.resize(edges, (1000, 500))) *#since the image resoution is very large*cv2.imshow(**"Image"**, cv2.resize(img, (1000, 500)) )  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**SCREENSHOT:**



**METHADOLOGY:**

After extracting all the line using hough line transform algorithm, I highlighted only those lines that matched the following condition of coordinates considering a **gap** parameter that maybe added/subtracted to complete the line gaps:

X1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6 //all coordinates of lines

X2=x3 , y2=y3, x5=x1, y5=x1, x6=x4, y6=y4

OR

X2=x3 , y2=y3, x6=x1, y6=x1, x5=x4, y5=y4