Page 1/3

```
goals9step6d.py
Nov 29, 23 21:40
   """goals8step8d.py
    Run the face (and eye) detectors and show the results.
3
4
   # Import OpenCV
6
   import numpy as np
8
   import cv2
   def detector(shared):
10
11
       class Ball:
            def __init__(self, pan = 0.5, tilt = 0.5):
12
                 self.deltapan = pan
13
                 self.deltatilt = tilt
14
                 self.conf = 0.2
15
16
        # Set up video capture device (camera). Note 0 is the camera number.
17
        # If things don't work, you may need to use 1 or 2?
18
        #camera = cv2.VideoCapture(0, cv2.CAP_V4L2)
19
        camera = cv2.VideoCapture(0, cv2.CAP_ANY)
20
21
        if not camera.isOpened():
22
            raise Exception ("Could not open video device: Maybe change the cam number?")
23
24
25
        # Change the frame size and rate. Note only combinations of
        # widthxheight and rate are allowed. In particular, 1920x1080 only
# reads at 5 FPS. To get 30FPS we downsize to 640x480.
26
27
        camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
camera.set(cv2.CAP_PROP_FPS, 30)
29
30
31
        # Get the face/eye detector models from XML files. Instantiate detectors.
32
        faceXML = "haarcascade_frontalface_default.xml"
33
        eyeXML1 = "haarcascade_eye.xml"
34
        eyeXML2 = "haarcascade_eye_tree_eyeglasses.xml"
35
36
37
        faceDetector = cv2.CascadeClassifier(faceXML)
        eyeDetector = cv2.CascadeClassifier(eyeXML1)
38
39
        # Pick some colors. Note OpenCV uses BGR color codes by default.
40
        blue = (255, 0, 0)
green = (0,255, 0)
41
        green = (0,255,0)
red = (0,0,255)
42
43
        white = (255, 255, 255)
44
45
46
        hsv = np.array([])
        bgr = np.array([])
47
48
49
        # scale
50
51
        scalepan = 0.00161932589506
52
        scaletilt = 0.00133859370275
53
        # Keep scanning, until 'q' hit IN IMAGE WINDOW.
54
        while True:
55
             # Grab an image from the camera. Often called a frame (part of sequence).
56
57
             ret, frame = camera.read()
58
             #convert image to hsv
59
            hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
60
61
62
             # Convert the image to gray scale.
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
63
64
65
             # Grab the faces - the cascade detector returns a list faces.
66
             faces = faceDetector.detectMultiScale(gray,
                 scaleFactor = 1.2,
67
                 minNeighbors = 5,
68
69
                 minSize = (30,30),
                 flags = cv2.CASCADE_SCALE_IMAGE)
70
71
             \# Process the faces: Each face is a bounding box of (x,y,w,h)
72
             # coordinates. Draw the bounding box ON THE ORIGINAL IMAGE.
73
74
            if len(faces) > 0:
                 # Grab the first face.
75
                 face = faces[0]
76
77
78
                 # Grab the face coodinates.
79
                 (x, y, w, h) = face
```

Draw the bounding box on the original color frame. #cv2.rectangle(frame.(x, y).(x+w-1, y+h-1).green.

80 81

82

```
qoals9step6d.pv
Nov 29, 23 21:40
                                                                                                        Page 2/3
                  # Also look for eyes - only within the region of the face!
                  # This similarly a list of eyes relative to this region.
85
                  eyes = eyeDetector.detectMultiScale(gray[y:y+h,x:x+w])
86
87
                  # Process the eyes: As before, eyes is a list of bounding
88
                  \# boxes (x,y,w,h) relative to the processed region.
89
                  for (xe, ye, we, he) in eyes:
90
91
                       # Can you draw circles around the eyes? Consider the function:
92
                       # cv2.circle(frame, (xc, yc), radius, (b,g,r), linewidth)
93
                      pass # replace this.
94
95
             # Show the image with the given title. Note this won't actually
96
             # appear (draw on screen) until the waitKey(1) below.
97
             (H, W, D) = frame.shape
99
             \#np.append(bgr, frame[W//2, H//2])
             #np.append(hsv, hsv[W//2, H//2])
#print("Frame: ", frame[W//2, H//2])
100
101
             #print("HSV: ", hsv[W//2, H//2])
102
103
             #Convert to binary
104
             binary = cv2.inRange(hsv, (113, 126, 78), (151, 255, 225))
105
             \#binary = cv2.inRange(hsv, (107, 142, 119), (119, 255, 255))
106
107
108
             #Erode, then Dialate
             binary=cv2.erode( binary, None, iterations=3)
binary=cv2.dilate(binary, None, iterations=3)
109
110
111
112
             #Dilate, then Erode
             binary=cv2.dilate(binary, None, iterations=3)
113
             binary=cv2.erode(binary, None, iterations=3)
114
115
116
             (contours, hierarchy) = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
117
             contoursTemp = sorted(contours, key=cv2.contourArea, reverse=True)
118
119
             contours=[]
             (xmax, ymax, rmax) = (0, 0, 0)
120
121
             balls = []
             for contour in contoursTemp:
122
                  if cv2.contourArea(contour) > 2000:
123
                      contours.append(contour)
124
                      ((xr, yr), radius) = cv2.minEnclosingCircle(contour)
125
                      cv2.circle(frame, (int(xr), int(yr)), 4, (91, 252, 104), -1)
126
                      cv2.circle(frame, (int(xr), int(yr)), int(radius), (91, 252, 104), 3)
127
                      x = \text{scalepan} * \text{float}(xr - 320)

y = \text{scaletilt} * \text{float}(yr - 240)
128
129
                      balls.append(Ball(x, y))
130
                      #print(xr, yr)
131
132
             if shared.lock.acquire():
133
                  """if rmax > 0:
           (x0, y0, x, y) = (320, 240, xmax, ymax)
134
           shared.deltapan = scalepan * float(x - x0)
135
           shared.deltatilt = scaletilt * float(y - y0)
136
           shared.newdata = True " " "
137
                 shared.balls = balls
138
                 shared.lock.release()
139
140
                  shared.scalepan = scalepan
141
                  shared.scaletilt = scaletilt
                  shared.newdata = True
142
143
144
             """ENCLOSING CIRCLE"""
145
146
             cv2.line(frame, (W//2,0), (W//2,H), (0,0,255), 1) cv2.line(frame, (0,H//2), (W,H//2), (0,0,255), 1)
147
148
             cv2.line(hsv, (W//2,0), (W//2,H), (0,0,255), 1)
149
             cv2.line(hsv, (0,H//2), (W, H//2), (0,0,255), 1)
150
             cv2.imshow('Processed Image', frame)
151
             #cv2.imshow('HSV Funkiness', hsv)
152
             cv2. imshow('Binary Image', binary)
153
154
             # Check for a key press IN THE IMAGE WINDOW: waitKey(0) blocks
155
             \# indefinitely, waitkey(1) blocks for at most 1ms. If 'q' break.
156
157
             # This also flushes the windows and causes it to actually appear.
             if (cv2.waitKey(1) \& 0xFF) == ord('q'):
158
159
                 break
             # Check if the main thread signals this loop to end.
160
161
             if shared.lock.acquire():
162
                  stop = shared.stop
                  shared.lock.release()
163
164
                  if stop:
```

165

brea

Nov 29, 23 21:40 **goals9step6d.py** Page 3/3