

Nov 29, 23 21:40

goals9step6d.py

Page 1/3

```

1  """goals8step8d.py
2
3  Run the face (and eye) detectors and show the results.
4  """
5
6  # Import OpenCV
7  import numpy as np
8  import cv2
9
10 def detector(shared):
11     class Ball:
12         def __init__(self, pan = 0.5, tilt = 0.5):
13             self.deltapan = pan
14             self.deltatilt = tilt
15             self.conf = 0.2
16
17     # Set up video capture device (camera). Note 0 is the camera number.
18     # If things don't work, you may need to use 1 or 2?
19     #camera = cv2.VideoCapture(0, cv2.CAP_V4L2)
20     camera = cv2.VideoCapture(0, cv2.CAP_ANY)
21
22     if not camera.isOpened():
23         raise Exception("Could not open video device: Maybe change the cam number?")
24
25     # Change the frame size and rate. Note only combinations of
26     # widthxheight and rate are allowed. In particular, 1920x1080 only
27     # reads at 5 FPS. To get 30FPS we downsize to 640x480.
28     camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
29     camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
30     camera.set(cv2.CAP_PROP_FPS, 30)
31
32     # Get the face/eye detector models from XML files. Instantiate detectors.
33     faceXML = "haarcascade_frontalface_default.xml"
34     eyeXML1 = "haarcascade_eye.xml"
35     eyeXML2 = "haarcascade_eye_tree_eyeglasses.xml"
36
37     faceDetector = cv2.CascadeClassifier(faceXML)
38     eyeDetector = cv2.CascadeClassifier(eyeXML1)
39
40     # Pick some colors. Note OpenCV uses BGR color codes by default.
41     blue = (255, 0, 0)
42     green = (0, 255, 0)
43     red = (0, 0, 255)
44     white = (255, 255, 255)
45
46     hsv = np.array([])
47     bgr = np.array([])
48
49     #
50     # scale
51     #
52     scalepan = 0.00161932589506
53     scaletilt = 0.00133859370275
54     # Keep scanning, until 'q' hit IN IMAGE WINDOW.
55     while True:
56         # Grab an image from the camera. Often called a frame (part of sequence).
57         ret, frame = camera.read()
58
59         #convert image to hsv
60         hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
61
62         # Convert the image to gray scale.
63         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
64
65         # Grab the faces - the cascade detector returns a list faces.
66         faces = faceDetector.detectMultiScale(gray,
67             scaleFactor = 1.2,
68             minNeighbors = 5,
69             minSize = (30,30),
70             flags = cv2.CASCADE_SCALE_IMAGE)
71
72         # Process the faces: Each face is a bounding box of (x,y,w,h)
73         # coordinates. Draw the bounding box ON THE ORIGINAL IMAGE.
74         if len(faces) > 0:
75             # Grab the first face.
76             face = faces[0]
77
78             # Grab the face coodinates.
79             (x, y, w, h) = face
80
81             # Draw the bounding box on the original color frame.
82             #cv2.rectangle(frame, (x, y), (x+w-1, y+h-1), green, 3)

```

Nov 29, 23 21:40

goals9step6d.py

Page 2/3

```

84         # Also look for eyes - only within the region of the face!
85         # This similarly a list of eyes relative to this region.
86         eyes = eyeDetector.detectMultiScale(gray[y:y+h,x:x+w])
87
88         # Process the eyes: As before, eyes is a list of bounding
89         # boxes (x,y,w,h) relative to the processed region.
90         for (xe,ye,we,he) in eyes:
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
96         # Show the image with the given title. Note this won't actually
97         # appear (draw on screen) until the waitKey(1) below.
98         (H, W, D) = frame.shape
99         #np.append(bgr, frame[W//2, H//2])
100        #np.append(hsv, hsv[W//2, H//2])
101        #print("Frame: ", frame[W//2, H//2])
102        #print("HSV: ", hsv[W//2, H//2])
103
104        #Convert to binary
105        binary = cv2.inRange(hsv, (113, 126, 78), (151, 255, 225))
106        #binary = cv2.inRange(hsv, (107, 142, 119), (119, 255, 255))
107
108        #Erode, then Dialate
109        binary=cv2.erode( binary, None, iterations=3)
110        binary=cv2.dilate(binary, None, iterations=3)
111
112        #Dilate, then Erode
113        binary=cv2.dilate(binary, None, iterations=3)
114        binary=cv2.erode(binary, None, iterations=3)
115
116        #Contours
117        (contours, hierarchy) = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
118        contoursTemp = sorted(contours, key=cv2.contourArea, reverse=True)
119        contours=[]
120        (xmax,ymax,rmax) = (0, 0, 0)
121        balls = []
122        for contour in contoursTemp:
123            if cv2.contourArea(contour) > 2000:
124                contours.append(contour)
125                ((xr, yr), radius) = cv2.minEnclosingCircle(contour)
126                cv2.circle(frame, (int(xr), int(yr)), 4, (91, 252, 104), -1)
127                cv2.circle(frame, (int(xr), int(yr)), int(radius), (91, 252, 104), 3)
128                x = scalepan * float(xr - 320)
129                y = scaletilt * float(yr - 240)
130                balls.append(Ball(x, y))
131                #print(xr, yr)
132        if shared.lock.acquire():
133            """if rmax > 0:
134            (x0, y0, x, y) = (320, 240, xmax, ymax)
135            shared.deltapan = scalepan * float(x - x0)
136            shared.deltatilt = scaletilt * float(y - y0)
137            shared.newdata = True"""
138            shared.balls = balls
139            shared.lock.release()
140            shared.scalepan = scalepan
141            shared.scaletilt = scaletilt
142            shared.newdata = True
143
144        """ENCLOSING CIRCLE"""
145
146
147        cv2.line(frame, (W//2,0), (W//2,H), (0,0,255), 1)
148        cv2.line(frame, (0,H//2), (W, H//2), (0,0,255), 1)
149        cv2.line(hsv, (W//2,0), (W//2,H), (0,0,255), 1)
150        cv2.line(hsv, (0,H//2), (W, H//2), (0,0,255), 1)
151        cv2.imshow('Processed Image', frame)
152        #cv2.imshow('HSV Funkiness', hsv)
153        cv2.imshow('Binary Image', binary)
154
155        # Check for a key press IN THE IMAGE WINDOW: waitKey(0) blocks
156        # indefinitely, waitkey(1) blocks for at most 1ms. If 'q' break.
157        # This also flushes the windows and causes it to actually appear.
158        if (cv2.waitKey(1) & 0xFF) == ord('q'):
159            break
160        # Check if the main thread signals this loop to end.
161        if shared.lock.acquire():
162            stop = shared.stop
163            shared.lock.release()
164            if stop:
165                break

```

Nov 29, 23 21:40

goals9step6d.py

Page 3/3

```
167     camera.release()
168     cv2.destroyAllWindows()
169
170 if __name__ == "__main__":
171     controller()
```