image_proc03

October 31, 2021

1 Tracking an object

This is a follow up of previous two tutorials. If you haven't studied them before, it is recommended to study them first

1.1 In this lesson, we will learn:

- to calculate euclidean distance
- drawing on image and video
- simple trignometry (sine and cosine law)

[1]:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import math
from numpy import *
```

1.2 Draw a line between center of a picture and center of the target

As we know how to calculate the centroid points of a contour using its moment, and we also have a clear understanding of how to draw on an image - we can draw a line between the COT and COI, calculate the euclidean distance between them. This comes in handy when you are trying to estimate the GPS location of a target in an image. By knowing the distance and angle between the line and the y-axis, we can easily interpolate the coordinates.

But first, let's make a function to calculate the euclidean distance between two points.

[2]:

```
def euclidean_distance(x,y):
    eud = math.sqrt((x[0]-y[0])**2 + (x[1]-y[1])**2)
    return eud
```

[3]:

```
euclidean_distance([1,2],[3,4])
```

[3]:

```
2.8284271247461903
```

Or just simply use the numpy version of calculating euclidean distance

[4]:

```
np.linalg.norm(np.array([1,2]) - np.array([3,4]))
```

```
[4]: 2.8284271247461903
```

1.2.1 Cosine Law

This is the law we will use to calculate the angles between two lines.

```
[5]: # first_side, second_side, side_opposite_to_angle, radian_boolean
     def cosineLaw(a,b,c, radian):
         angle = math.acos(((a**2)+(b**2)-(c**2))/(2*a*b))
         if radian==True:
             return angle
         elif radian==False:
             return math.degrees(angle)
```

```
[ ]: cosineLaw(4,5,6,radian=False)
```

1.2.2 Calculate the COI and COT on images

After calculations, we will draw some lines, mark some points, and calculate euclidean distance between points using function we made earlier.

```
[ ]: im = cv2.imread('samples/obt.png')
     #im = cv2.resize(im, (600,480))
     height,width,_ = im.shape
     im_cx,im_cy = int(width/2),int(height/2)
     print(im_cx,im_cy)
     # Contour Finding
     hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)
     upper = np.array([104,114,112])
     lower = np.array([255,255,255])

     mask = cv2.inRange(hsv, upper, lower)
     res = cv2.bitwise_and(im, im, mask=mask)
     res = cv2.GaussianBlur(res, (1,1), 0)
     res_gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
     _, threshold = cv2.threshold(res_gray, 0, 255, cv2.THRESH_BINARY+cv2.
      ↪THRESH_OTSU)
     contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.
      ↪CHAIN_APPROX_NONE)
     min_area = 30**2
     for contour in contours:
         area = cv2.contourArea(contour)
         if area>min_area:
             # calculating the centroid
             moment = cv2.moments(contour)
             cx = int(moment['m10']/moment['m00'])
             cy = int(moment['m01']/moment['m00'])
```

```
        # make a rectangle bounding the contour
        [x, y, w, h] = cv2.boundingRect(contour)

        # draw a rectangle surrounding the contour image
        cv2.rectangle(im, (x, y), (w+x, h+y), (0,255,0), 2)

        # put the centroid text
        #cv2.circle(im,(cx,cy), 5, (255,0,255), -1)
        #cv2.putText(im, str(cx)+','+str(cy), (cx,cy), 2, 1, (0,0,0), 1, 0)
    #endif
#endfor
# Calculating Euclidean Distances
pi2pt = round(euclidean_distance([im_cx,im_cy],[cx,cy]),2)
pi2py = round(euclidean_distance([im_cx,im_cy],[0,im_cy]),2)
pt2py = round(euclidean_distance([cx,cy],[0,im_cy]),2)
# Calculating the angles
a_pi = round(cosineLaw(pi2pt,pi2py,pt2py,radian=False),1)
a_pt = round(cosineLaw(pi2pt,pt2py,pi2py,radian=False),1)
a_py = round(cosineLaw(pt2py,pi2py,pi2pt,radian=False),1)
# Marking the center of COI
cv2.circle(im,(im_cx,im_cy), 5, (255,0,255), -1)
cv2.putText(im, str(im_cx)+','+str(im_cy)+', ang: '+str(a_pi), (im_cx,im_cy), 2, 1, (0,255,255), 1, 0)
# Marking the center of COT
cv2.circle(im,(cx,cy), 5, (255,0,255), -1)
cv2.putText(im, str(cx)+','+str(cy)+', ang: '+str(a_pt), (cx,cy), 2, 1, (0,255,255), 1, 0)
# Marking the center of y-axis
cv2.circle(im,(0,im_cy), 5, (255,0,255), -1)
cv2.putText(im, str(0)+','+str(im_cy)+', ang: '+str(a_py), (0,im_cy), 2, 1, (0,255,255), 1, 0)
# line from COI to COT
cv2.line(im, (im_cx,im_cy), (cx,cy), (0,0,255), 2)
# Line from COI to y-axis
cv2.line(im, (im_cx,im_cy), (0,im_cy), (0,0,255), 2)
# Line from COT to y-axis
cv2.line(im, (cx,cy), (0,im_cy), (0,0,255), 2)
# Putting the pi2pt in the midpoint of the COI and COT
cv2.putText(im, str(pi2pt), (int((im_cx+cx)/2),int((im_cy+cy)/2)), 2, 1, (0,0,255), 1, 0)
# Putting the pi2py in the midpoint of the COI and y-axis
cv2.putText(im, str(pi2py), (int((im_cx+0)/2),int((im_cy+im_cy)/2)), 2, 1, (0,0,255), 1, 0)
# Putting the pt2py in the midpoint of the COT and y-axis
cv2.putText(im, str(pt2py), (int((cx+0)/2),int((cy+im_cy)/2)), 2, 1, (0,0,255), 1, 0)
```

```
cv2.imshow('im', im)
#cv2.imwrite('data/dist.png', im)
cv2.waitKey(0)
```

[6]:

```
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 2000)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 2000)
min_area = 50**2
while(cap.isOpened):
    _,frame = cap.read()
    height,width,_ = frame.shape
    im_cx,im_cy = int(width/2),int(height/2)

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    upper = np.array([162,58,62])
    lower = np.array([255,255,255])

    mask = cv2.inRange(hsv, upper, lower)
    res = cv2.bitwise_and(frame, frame, mask=mask)
    res = cv2.GaussianBlur(res, (1,1), 0)

    # detection of contours
    res_gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    _, threshold = cv2.threshold(res_gray, 0, 255, cv2.THRESH_BINARY+cv2.
 ↪THRESH_OTSU)
    contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_NONE)

    for contour in contours:
        area = cv2.contourArea(contour)
        if area>min_area:
            # calculating the centroid
            moment = cv2.moments(contour)
            cx = int(moment['m10']/moment['m00'])
            cy = int(moment['m01']/moment['m00'])

            # make a rectangle bounding the contour
            [x, y, w, h] = cv2.boundingRect(contour)

            # draw a rectangle surrounding the contour image
            cv2.rectangle(frame, (x, y), (w+x, h+y), (0,255,0), 2)

            # put the centroid text
            cv2.putText(frame, str(cx)+','+str(cy), (cx,cy), 2, 1,␣
 ↪(255,255,255), 2, 0)
```

```
            # Calculating Euclidean Distances
            pi2pt = round(euclidean_distance([im_cx,im_cy],[cx,cy]),2)
            pi2py = round(euclidean_distance([im_cx,im_cy],[0,im_cy]),2)
            pt2py = round(euclidean_distance([cx,cy],[0,im_cy]),2)

            # Calculating the angles
            a_pi = round(cosineLaw(pi2pt,pi2py,pt2py,radian=False),1)
            a_pt = round(cosineLaw(pi2pt,pt2py,pi2py,radian=False),1)
            a_py = round(cosineLaw(pt2py,pi2py,pi2pt,radian=False),1)

            # Marking the center of COI
            cv2.circle(frame,(im_cx,im_cy), 5, (255,0,255), -1)
            cv2.putText(frame, str(im_cx)+','+str(im_cy)+', ang: '+str(a_pi),␣
 ↪(im_cx,im_cy), 2, 1, (0,255,255), 1, 0)
            # Marking the center of COT
            cv2.circle(frame,(cx,cy), 5, (255,0,255), -1)
            cv2.putText(frame, str(cx)+','+str(cy)+', ang: '+str(a_pt),␣
 ↪(cx,cy), 2, 1, (0,255,255), 1, 0)
            # Marking the center of y-axis
            cv2.circle(frame,(0,im_cy), 5, (255,0,255), -1)
            cv2.putText(frame, str(0)+','+str(im_cy)+', ang: '+str(a_py),␣
 ↪(0,im_cy), 2, 1,(0,255,255), 1, 0)
            # line from COI to COT
            cv2.line(frame, (im_cx,im_cy), (cx,cy), (0,0,255), 2)
            # Line from COI to y-axis
            cv2.line(frame, (im_cx,im_cy), (0,im_cy), (0,0,255), 2)
            # Line from COT to y-axis
            cv2.line(frame, (cx,cy), (0,im_cy), (0,0,255), 2)

            # Putting the pi2pt in the midpoint of the COI and COT
            cv2.putText(frame, str(pi2pt), (int((im_cx+cx)/2),int((im_cy+cy)/
 ↪2)), 2, 1, (0,0,255), 1, 0)
            # Putting the pi2py in the midpoint of the COI and y-axis
            cv2.putText(frame, str(pi2py), (int((im_cx+0)/2),int((im_cy+im_cy)/
 ↪2)), 2, 1,(0,0,255), 1, 0)
            # Putting the pt2py in the midpoint of the COT and y-axis
            cv2.putText(frame, str(pt2py), (int((cx+0)/2),int((cy+im_cy)/2)),␣
 ↪2, 1, (0,0,255), 1, 0)
        #endif
    #endfor
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xff == ord('q'):
        cv2.imwrite('samples/proc3.png',frame)
        break
    #endif
#endwhile
```

```
cap.release()
cv2.destroyAllWindows()
```

[ ]:

```

```