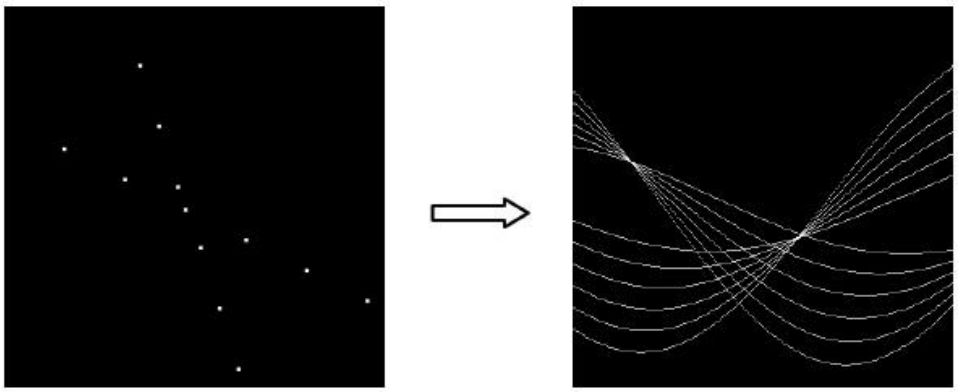
پاسخ سوال اول

با توجه به اینکه دارای تصویر داده شده دارای دو دسته تشکیل شده از 6 منحنی است، هر کدام از این 6 تا دارای طلاقی هستند که بیانگر این است که هر دسته نشان دهنده ی 6 نقطه که در یک راستا است پس دارای دو راستا است تصویر اصلی. همچنین در یک نقطه که خیلی نزدیک به محل تقاطع است،بیانگراز هم گذز و داشتن تداخل در این راستاها میابشد و همچنین میتوان از روی اختلاف فازی که در منحنی ها وجود دارد زاویه ی بین این دو راستا را نیز حدس زد.



(a) $p_i = (x_i, y_i)$ (b) $\rho = x_i \cos(\theta) + y_i \sin(\theta)$

منبع:

لینک

پاسخ سوال دوم

می دانیم اگر k تعداد تکرار باشد ، p احتمال آنکه هیچ مجموعه درستی انتخاب نشده باشد واگر wنسبت تعداد نقاط inlier به تمام نقاط باشد رابطه ی زیر برقرار است:

$$k = \frac{log(1-p)}{log(1-w^2)}$$

حال در این مسئله:

W=0.4

1-p=1-0.99=0.01

پس داریم:

همچنین چون در سوال دایره است و به سه پارامتر نیاز داریم،به توان 3 میرسانیم w را پس:

K=log(0.01)/log(0.064)=70

با تقریب برابر 70 تکرار نیاز است.

پاسخ سوال سوم

الگوریتم LSD:

هدف از این الگوریتم یافتن نقاط ابتدا و انتهای پارهخطهای موجود در تصویر است.(نیاز به تقسیم بندی ندارد)

هر پارهخط بجای 2پارامتر توسط 4پارامتر مشخص میشود.

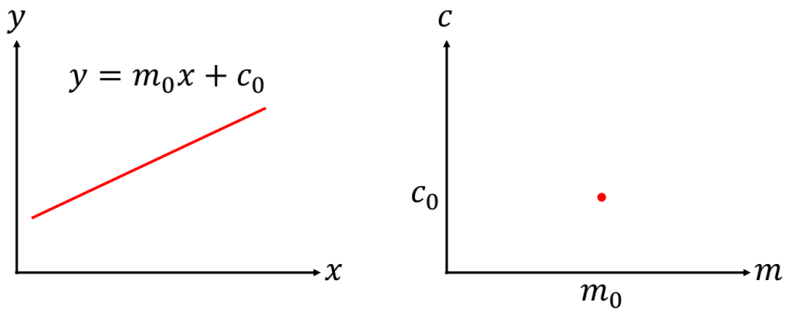
1. مزیت اصلی الگوریتم LSDآن است که به خوبی از جهت گرادیان استفاده میکند.
2. در زمان خطی انجام میشود.
3. نتایج دقیق subpixel ایی میدهد.
4. یک تصویر greyscale ورودی میگیرد.

الگوریتم HOUGH:

ایده اصلی تبدیل Houghبر تغییر فضا و رایگیری است.

هر خط در فضای (x,y)معادل با یک نقطه در فضای (m,c)است.

1. برای یافتن خط مشخص تر مفید است. پارامترها امکان تنظیم را دارند و گزینه ای برای ترکیب بخشها (از طریق maximum line gap parameter) برای بازگشت خطوط طولانی تر دارد.
2. پیچیدگی زمانی آن وابسه به پارامترها است.پس احتمالا زمان بیشتری از LSD صرف میکند.(کندتر است)
3. دقت آن وابسته به موارد زیادی مانند مرحله لبه یابی است،لبه یابی حساس به نویز است که همین امر باعث ایجاد محدودیت هایی میشود.
4. یک تصویر باینری ورودی میگیرد.
5. HOUGHقادر به تعیین نقاط انتهایی یک بخش خط نیست. hough می تواند فقط خطوطی را که از کل تصویر عبور می کنند شناسایی کند. بنابراین برای اینکه hough بتواند نقاط انتهایی یک بخش را شناسایی کند ، باید روشهای تقسیم بندی نیز اعمال شود.که همین امر نیز سربار زمانی دارد.



منبع:

لینک

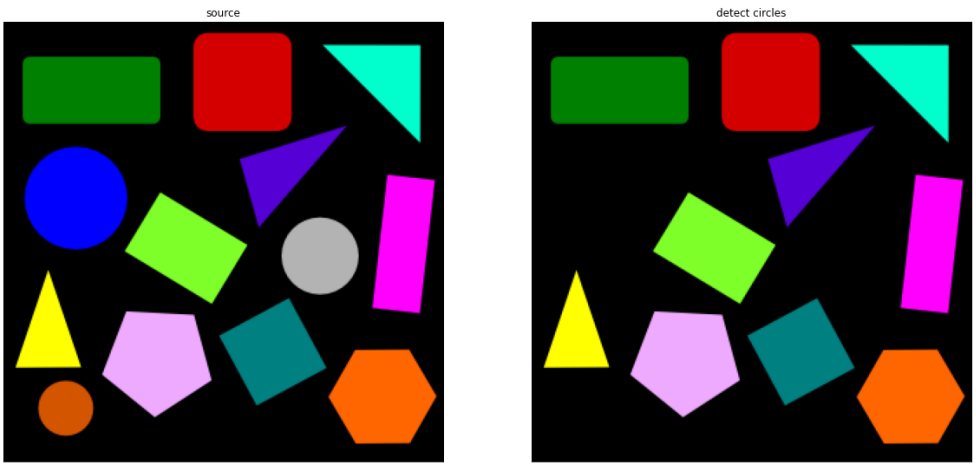
لینک

پاسخ سوال چهارم

بخش اول:این سوال را با راهنمایی کرده شده در متن سوال و لینک زیر پیاده سازی کردم.

همانطور که در کلاس حل تمرین نیز گفته شده من مقدار3 را به شعاع اضافه کردم.

```
#Writer your code here
gray = cv2.cvtColor(out_img, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray,(3,3),1)
detected_circles = cv2.HoughCircles(gray,
                cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
            param2 = 30, minRadius = 1, maxRadius = 100)
for x, y, r in detected_circles[0]:
    cv2.circle(out_img, (x,y), int(r+3), 0, -1)  #r +/- <4
return out_img
```



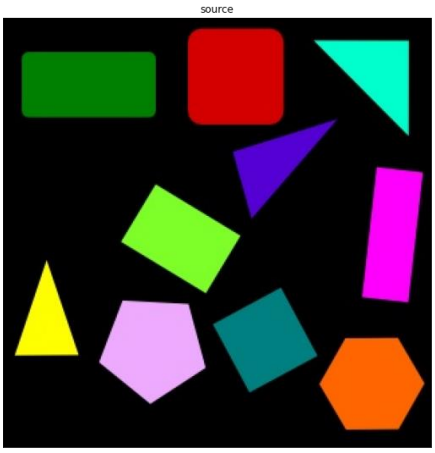
منبع:

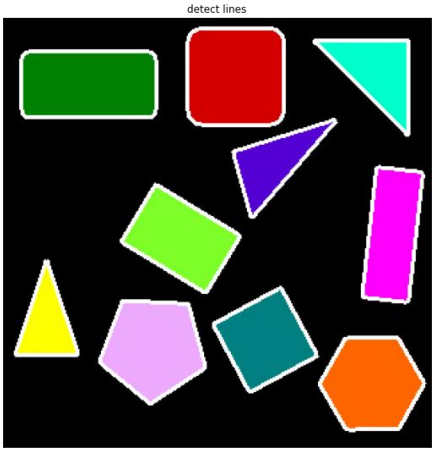
لینک

بخش دوم:

در این سوال نیز از لینک های داخل سوال کمک گرفتم و سعی کردم در کاممنت های داخل کد شفاف سازی کنم موارد استفاده شده را.

```
#Writer your code here
gray = cv2.cvtColor(out_img, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray,(3,3),1)
edges = cv2.Canny(gray,20,100)
lines=np.array([])

#The other parameter ρ=0.1 defines how "fat" a row of the accumulator is.
#Output vector of lines. Each line is represented by a 4-element vector (x1,y1,x2,y2) , where (x1,y1) and (x2,y2) are the
#Angle resolution of the accumulator in radians.
lines = cv2.HoughLinesP(edges, 0.1, np.pi / 180, 1, lines)

white=(255,255,255)
for line in lines:
    for x1,y1,x2,y2 in line:
        cv2.line(image,(x1,y1),(x2,y2),white,2) #draw white lines


return image
```





لینک 1

پاسخ سوال پنجم

در این سوال بسیار کلاس حل تمرین مفید بود من خیلی سرچ کردم ولی چیزی پیدا نمیکردم که کمک بکند ولی کلاس حل تمرین بسیار عالی بود.

طبق موارد گفته شده چند تابع نوشتم:

در تابع اولی نقاط سفید را پیدا میکنم و خروجی را به تابع بعدی برای پیدا کردن دو نقطه ی رندوم میدهم سپس شیب خط و عرض از مبدا خط واصل بین این دو نقطه را حساب میکنم و در تابع بعدی فاصله ی هریک از نقاط

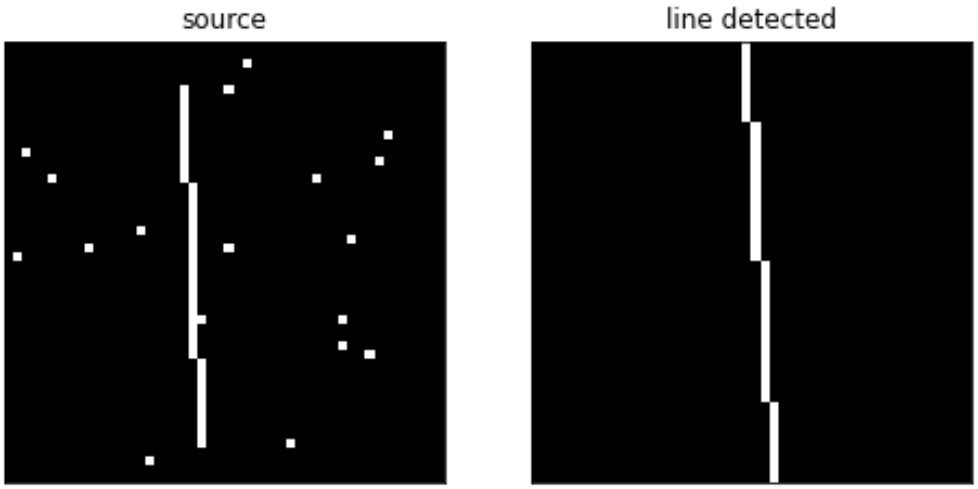
را با این خط بدست میاورم.همچنین thereshold=1 در نظر گرفته ام.در ادامه برای پیدا کردن بیشترین تعداد نقاط inlier موارد 20 بار تکرار میکنم این اعمال را و نتیجه ی مطلوب حاصل گشت.

سپس rho وtheta را با توجه به فرمول های داخل اسلاید محاسبه کردم.

```
#Write your code here
white_pixels=non_zero_pixels(img)
max_inlier_points=-999999
for i in range(20):
    point1,point2=random_points(white_pixels)
    m,c=line_from_points(point1,point2)
    count,inlinear_points=distance(white_pixels,m,c)
    if count>max_inlier_points:
        max_inlier_points=count
        max_inliers=inlinear_points


rho,theta=car_to_pal(max_inliers)

return rho, theta
```

```
import random
def non_zero_pixels(img):
    white_pixels=[]
    m,n=img.shape
    for i in range(m):
        for j in range(m):
            if img[i,j] != 0:
                white_pixels.append((np.float32(i),np.float32(j)))
    return white_pixels

def random_points(white_pixels):
    p1=0
    p2=0
    p1,p2 = random.sample(range(len(white_pixels)),2)
    point1, point2 = white_pixels[p1], white_pixels[p2]
    return point1, point2

def line_from_points(point1,point2):
    x1,y1=point1
    x2,y2=point2
    m = (y2 - y1) / (x2 - x1)
    c = y1 - m * x1
    return m,c

def distance(dots,m,c):
    threshold=1
    inlinear_points = []
    for d in dots:
        temp_x,temp_y=d
        distance = np.abs((m*temp_x+c)-temp_y)
        if distance < threshold:
            inlinear_points.append(d)
    return len(inlinear_points), np.array(inlinear_points)


def car_to_pal(inlinear_points):
    x_bar=0
    y_bar=0
```

خروجی:



پاسخ سوال ششم

در این سوال با توجه به شبه کد موجود در اسلایدها پیش رفتم و از منابعی که ذکرمیکنم در انتها کمک گرفتم.نکته ی مهمی که در ابتدا به آن توجه نمیکردم اعمال لبه یاب است.

```
    #Write your code here

    # Rho and Theta ranges
    dtheta = 1
    drho = 1
    width, height = img.shape

    thetas =np.deg2rad(np.arange(-90,90,step=1))
    distance=int(np.sqrt(np.square(height) + np.square(width)))
    rhos = np.arange(-distance, distance, step=drho)

    cos_thetas = np.cos(thetas)
    sin_thetas = np.sin(thetas)

    hough_transform = np.zeros((2*distance, len(thetas)))

    #edge detection
    edges = cv2.Canny(img, 20, 120)
    value_threshold =0
    are_edges = np.where(edges > value_threshold)
    y_idxs, x_idxs =are_edges

    #voting
    for i in range(len(x_idxs)):
        x = x_idxs[i]
        y = y_idxs[i]
        for t_idx in range(len(thetas)):
            rho = int(round(x * cos_thetas[t_idx] + y * sin_thetas[t_idx])+int(distance))
            hough_transform[rho][t_idx] += 1

    #Find the value(s) of ρ, θ where hough_transform(ρ, θ ) is a large local maximum
    max_value=np.argmax(hough_transform)
    x_max,y_max=np.unravel_index(max_value,hough_transform.shape)
    rho = x_max-distance
    theta = thetas[y_max]

    return rho, theta, hough_transform
```

منبع:

لینک

لینک

لینک

پیوست:

در صورت وجود هر گونه خطایی برای اطمینان در اجرای فایل های پایتون، فایل نوت بوک را نیز در کنار گزارش اپلود کرده ام.

از حسن توجه شما سپاس گزارم.