به نام خالق رنگین کمان

ستاره باباجانی – گزارش تمرین سری سوم

سوال 1:

a) گرادیان تصویر I(x,y) یک بردار دو بعدی است که با مشتق گرفتن نسبت به x و y بدست می آید. بردار گرادیان تصویر به صورت زیر محاسبه میشود:

$$\nabla I(x,y) = [g_x, g_y] = \left[\frac{\partial \mathrm{I}}{\partial \mathrm{x}}, \frac{\partial \mathrm{I}}{\partial \mathrm{y}}\right]$$

که در آن $\frac{\partial \mathrm{I}}{\partial \mathrm{x}}$ و $\frac{\partial \mathrm{I}}{\partial \mathrm{y}}$ به ترتیب مشتق جزئی تصویر نسبت به متغیرهای x و y هستند.

b) محاسبه بردار گرادیان یک تصویر در پردازش تصویر و بینایی ماشین مفید است زیرا اطلاعات مهمی را درباره تغییرات شدت نوری در تصویر ارائه می‌دهد و همچنین باعث میشود که اگر در جهت افقی و عمودی، تصویر تغییری در رنگ و روشنایی داشته باشته متوجه آن شویم. چند کاربرد اصلی این بردار به شرح زیر است:

1. تشخیص لبه‌ها: لبه‌ها در تصاویر معمولاً در جایی اتفاق می‌افتند که شدت نور به طور ناگهانی تغییر می‌کند. بردار گرادیان در این نقاط بیشترین مقدار را دارد.
2. بهبود کیفیت تصویر: با استفاده از اطلاعات بردار گرادیان، می‌توان تصویر را بهبود داد و جزئیات را تقویت کرد. به عنوان مثال، می‌توان از مقادیر بردار گرادیان برای اعمال فیلترهای شارپنینگ و افزایش وضوح تصویر استفاده کرد.

3. تشخیص ویژگی ها: در الگوریتم‌های تشخیص ویژگی‌ها مانند ماشین بینایی و شناسایی الگو، بردار گرادیان به عنوان ویژگی‌های مهمی برای تشخیص اجسام و شیءها استفاده می‌شود.

c) اندازه گرادیان تعریف شده روی تصویر به این نحو محاسبه میشود که گرادیان بدست امده از مشتق های افقی و عمودی در قسمت قبل را، هر بخش به طور جداگانه به توان دو رسیده و باهم جمع میکنیم و رادیکال میگیریم. با تقریب میتوان گفت این عبارت برابر با جمع اندازه هر دو مشتق خواهد بود:

$$M(x,y) = \left|\left|\nabla I(x,y)\right|\right| = \text{mag}(\nabla I) = \sqrt{{g_x}^2 + {g_y}^2}$$

$$\approx |g_x| + |g_y|$$

d) جهت گرادیان تصویر با رابطه زیر یعنی تانژانت معکوس g_x و g_y را محاسبه میشود:

$$\alpha(x,y) = \text{dir}(\nabla I) = atan2(g_y, g_x)$$

e) آشکارساز لبه Canny یکی از الگوریتم‌های پرکاربرد در پردازش تصویر برای تشخیص لبه‌ها است. این الگوریتم از بردار گرادیان برای تشخیص لبه‌ها استفاده می‌کند و مراحل اصلی آن به شرح زیر هستند:

1. هموار سازی تصویر با فیلتر گاوسی: تصویر ورودی با استفاده از فیلتر گاوسی برای کاهش نویز از پیش فرستاده می‌شود. این مرحله باعث می‌شود که تشخیص لبه‌ها با دقت بیشتری انجام شود.
2. محاسبه گرادیان: بردار گرادیان تصویر به‌دست می‌آید. این کار با استفاده از فیلترهای سوبل یا فیلترهای روب است.

3. حذف مقادیر غیر بیشینه: در این مرحله، از تکنیک Thresholding استفاده می‌شود تا نقاطی که گرادیان آن‌ها بیشتر از یک حد آستانه (threshold)مشخص شده است، به عنوان نقاط لبه انتخاب شوند.
4. آستانه گذاری دو مرحله ای: این مرحله شامل دو قسمت زیر میشود:
   - کاهش لبه‌های نقاطی: در این مرحله از تکنیک Non-maximum Suppressionاستفاده می‌شود تا لبه‌های ضخیم شده کاهش یابند و فقط لبه‌هایی که به عنوان نقطه بیشینه در جهت گرادیان شناخته شده‌اند، باقی بمانند.
   - تشخیص لبه‌های واقعی: در این مرحله از تکنیک هیسترزیس Thresholdingاستفاده می‌شود تا لبه‌های موجود به عنوان لبه‌های واقعی یا لبه‌های ضعیف شناسایی شوند.

مزایای آشکارساز لبه Canny نسبت به رویکردهای جایگزین به شرح زیر هستند:

1. کاهش نویز: با اعمال فیلتر گاوسی، نویز تصویر کاهش می‌یابد که باعث افزایش دقت در تشخیص لبه‌ها می‌شود.
2. دقت بالا: با استفاده از تکنیکNon-maximum Suppression ، لبه‌های ضخیم شده کاهش یافته و فقط لبه‌هایی با دقت بالا شناسایی می‌شوند.
3. کاهش تأثیر نویز در تصمیم‌گیری: با استفاده از تکنیک هیسترزیس Thresholding، تأثیر نویز در تصمیم‌گیری کاهش می‌یابد و فقط لبه‌های واقعی تشخیص داده می‌شوند.

4. بر خلاف روش‌های جایگزین، مانند روش استفاده از فیلترهای ساده و Thresholdingمعمولی، آشکارساز لبه Canny دقت بالاتری در تشخیص لبه‌ها ارائه می‌دهد و در بسیاری از برنامه‌های پردازش تصویر مورد استفاده قرار می‌گیرد.

f) استفاده از عملگر لاپلاسین برای تشخیص لبه ممکن است در برخی موارد مفید باشد، اما در عمل، عملگرهای مانند Sobel و Canny برای این منظور بیشتر استفاده می‌شوند. دلایل اصلی عدم محبوبیت عملگر لاپلاسین به شرح زیر هستند:

1. حساسیت به نویز: عملگر لاپلاسین بسیار حساس به نویز است و در مواجهه با نویزهای موجود در تصویر، ممکن است خروجی آن لبه‌های زیادی که ناخواسته و معنی‌دار نیستند، ایجاد کند. این امر می‌تواند منجر به تشخیص اشتباه لبه‌ها یا لبه‌های غیرقابل استفاده شود.
2. ضخامت لبه: خروجی عملگر لاپلاسین عموماً دارای لبه‌های ضخیم‌تری است که ممکن است دقت تشخیص را کاهش دهد. در عوض، عملگرهای مانند Sobel و Canny به دقت بیشتری در تشخیص لبه‌ها کمک می‌کنند و لبه‌های ضخیم را کاهش می‌دهند.
3. استفاده از Thresholding برای تشخیص لبه با استفاده از عملگر لاپلاسین ، نیازمند به اعمال یک آستانه بر روی خروجی عملگر است. این کار ممکن است به دلیل حساسیت به نویز و افزایش ضخامت لبه‌ها، انتخاب آستانه مناسب را دشوار کند و منجر به اشتباهات در تشخیص لبه‌ها شود. در عوض، روش‌های مانند Canny از تکنیک‌های هیسترزیس

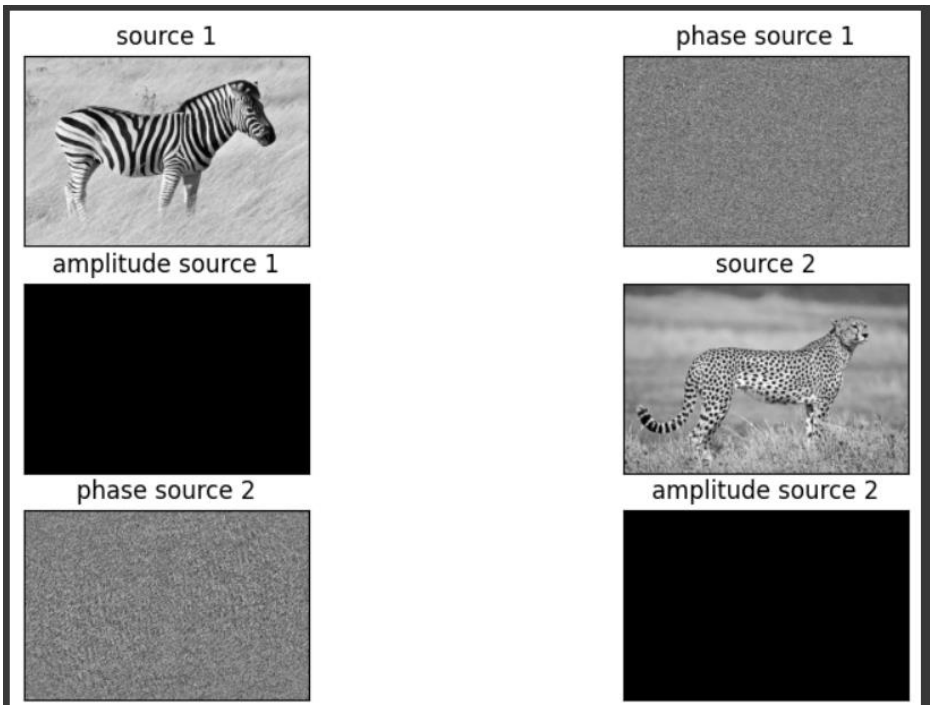
Thresholding به طور مناسب استفاده می‌کنند که دقت بیشتری در تشخیص لبه‌ها فراهم می‌کنند.

4. همچنین در عملگر لاپلاسین ما به صور مستقیم تصویر را با یک کرنل کانوالو میکنیم اما در 2 روش دیگر ابتدا تاثیر نویز در تصویر کاهش میابد و هموار میشود و یا ماتریس هموار کننده به کرنل اضافه میشود.

سوال 2:

a) ابتدا تصاویر را خوانده و تبدیل فوریه آنها را با استفاده از np.fft.fft2 محاسبه میکنیم و دامنه و فازشان را نمایش میدهیم:

```python
def draw_phase_amplitude(image):
    '''
    Returns the phase image and the amplitude image from the input image.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        tuple of numpy.ndarray: The tuple of the phase image and the amplitude image.
    '''

    phase = image.copy()
    amp = image.copy()

    # Writer your code here

    # Compute Fourier transform
    FourierTransform = np.fft.fft2(image)

    # Compute amplitude and phase
    phase = np.angle(FourierTransform)
    amplitude = np.abs(FourierTransform)

    return phase, amplitude
```



b) حال در این مرحله جای دامنه و فاز را عوض میکنیم و تبدیل فوریه معکوس را با استفاده از np.fft.ifft2 میگیریم:

```python
def change_phase_domain(image1, image2):
    '''
    Substitutes the phase of image1 by the phase of image2 and returns two new images.

    Parameters:
        image1 (numpy.ndarray): The input image1.
        image2 (numpy.ndarray): The input image2.

    Returns:
        tuple of numpy.ndarray: The tuple of result images.
    '''

    img1 = image1.copy()
    img2 = image2.copy()

    # Write your code here

    # Compute Fourier transform of the images
    FourierTransform_1 = np.fft.fft2(img1)
    FourierTransform_2 = np.fft.fft2(img2)

    # Swap the phases
    NewFourierTransform_1 = np.abs(FourierTransform_1) * np.exp(1j * np.angle(FourierTransform_2))
    NewFourierTransform_2 = np.abs(FourierTransform_2) * np.exp(1j * np.angle(FourierTransform_1))

    # Compute inverse Fourier transform
    img1 = np.fft.ifft2(NewFourierTransform_1).real
    img2 = np.fft.ifft2(NewFourierTransform_2).real

    return img1, img2
```

تصویر نهایی به شرح زیر خواهد بود:

حال به بررسی تغییرات با جزئیات میپردازیم:

1. تغییر دامنه: تغییر دامنه تصاویر باعث تغییر شدت یا روشنایی فرکانس های فضایی مختلف موجود در تصاویر می شود. افزایش دامنه این فرکانس‌ها را تقویت می‌کند و ویژگی‌های مربوطه را در تصویر به‌دست‌آمده برجسته‌تر می‌کند، در حالی که کاهش دامنه اثر معکوس خواهد داشت.

2. تغییر فاز: تغییر فاز تصاویر، آرایش فضایی یا روابط فازی اجزای فرکانس را تغییر می دهد. این می تواند منجر به تغییر در الگوی فضایی یا بافت تصاویر حاصل شود، حتی اگر شکل و ساختار کلی یکسان باقی بماند.

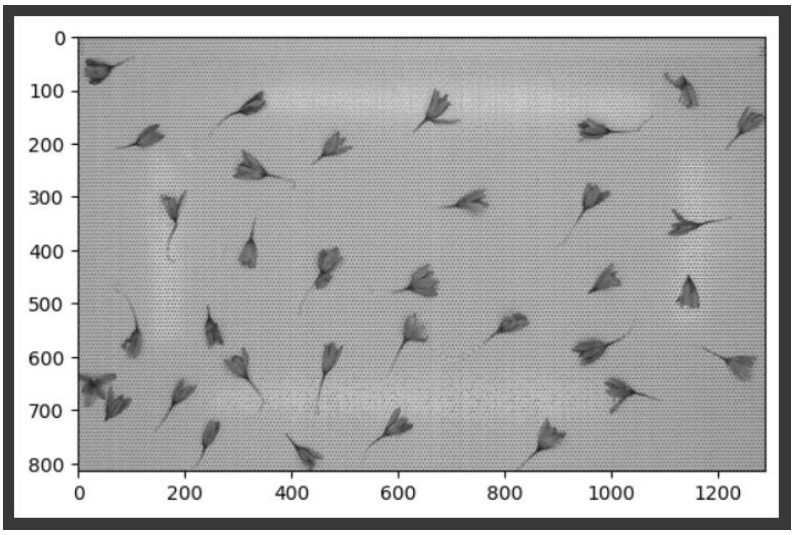
3. تبدیل فوریه معکوس: پس از دستکاری دامنه و فاز تصاویر در حوزه فرکانس، اعمال تبدیل فوریه معکوس آنها را به حوزه فضایی بازمی گرداند. این فرآیند تصاویر را با استفاده از اطلاعات فرکانس اصلاح شده بازسازی می کند و در عین حال ویژگی های فضایی آنها را حفظ می کند.

4. جلوه های بصری: تصاویر به دست آمده ممکن است بسته به میزان و ماهیت دامنه و تغییرات فاز، جلوه های بصری مختلفی را نشان دهند. این اثرات می تواند شامل تغییرات در بافت، کنتراست، روشنایی و آرایش فضایی ویژگی ها باشد.

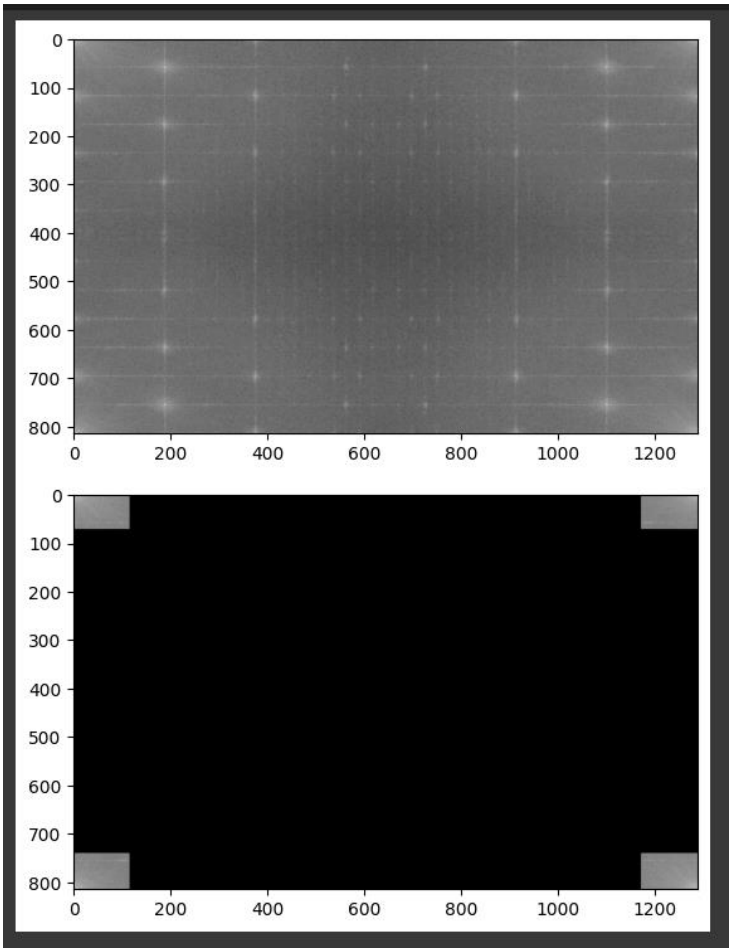
به طور کلی، این آزمون اهمیت دستکاری دامنه فرکانس را در پردازش تصویر نشان می‌دهد و نشان می‌دهد که چگونه تغییرات در حوزه فوریه به تغییرات حوزه فضایی در تصاویر حاصل تبدیل می‌شوند.

سوال 3:

a) ابتدا تصویر را خوانده و آن را در فضای gray scale میبریم:

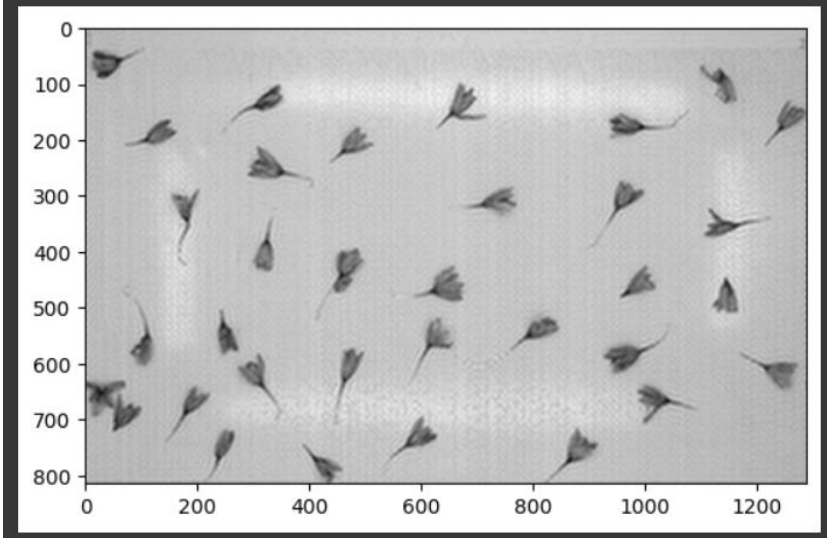


حال تبدیل فوریه تصویر را با تابع FFT.FFT2 بدست آورده و با Ifft و مراحل زیر نویز را از تصویر برمیداریم. نتیجه به این صورت میشود:



تصویر نهایتا به اینصورت خواهد بود:

```
1  # apply IFFT (2d fourier transform)
2 iFFT_image = np.real(np.fft.ifft2(FFT_image))
3 plt.imshow(iFFT_image, cmap='gray')
4 plt.show()
```



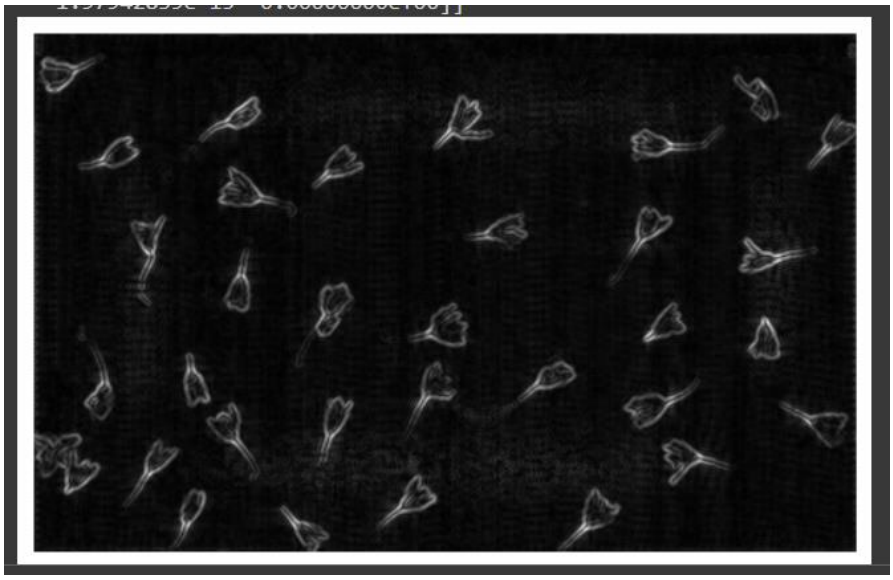
b)ابتدا عکس را داده و سپس هم treshhold اول و دوم را مشخص میکنیم.
جاهایی که مقدار پیکسل کمتر از trshhold1 باشد جزو لبه ها در نظر گرفته نمیشوند و اگر بیشتر از treshhold2 باشد، یعنی یک strong edge است و در واقع احتمال لبه بودنش زیاد است، جاهایی که بین این treshhold1 و treshhold2 هستند، اگر به یک strong edge بچسبند جز لبه محاسبه میشوند. خروجی به دست آمده به شرح زیر است:

c) ابتدا ماتریس های سوبل x و y تعریف شده و گرادیان در جهت افقی و عمودی را حساب میکنیم. حال اندازه گرادیان و جهت آن را که mag و dirs. هستند. محاسبه میکنیم. جهت گرادیان با فرمول arctan2 گفته شده در اسلایدهای درس، بدست می اید. این کار برای دو تصویر خروجی مرحله قبل (مرحله b و قبل آن) انجام میدهیم تا خروجی های مورد نظر مشاهده شود:

```
sobel_x = np.array([
     [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]], dtype=np.int32)
sobel_y = np.array([
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1]], dtype=np.int32)
gx = cv2.filter2D(iFFT_image, ddepth=-1, kernel=sobel_x)
gy = cv2.filter2D(iFFT_image, ddepth=-1, kernel=sobel_y)
mag = np.sqrt(gx**2 + gy**2)
dirs = np.arctan2(gy, gx)
print(dirs)
plt.imshow(mag, cmap="gray")
plt.axis('off')
plt.show()

gx2 = cv2.filter2D(CannyImage, ddepth=-1, kernel=sobel_x)
gy2 = cv2.filter2D(CannyImage, ddepth=-1, kernel=sobel_y)
mag2 = np.sqrt(gx2**2 + gy2**2)
dirs2 = np.arctan2(gy2, gx2)
print(dirs2)
plt.imshow(mag2, cmap="gray")
plt.axis('off')
plt.show()
```

نتیجه اعمال کد جهت گرادیان، بر روی iFFT_image:

```
[[-1.57079633e+00  0.00000000e+00 -3.14159265e+00 ...  1.08413082e-15
   2.05413258e-15 -1.57079633e+00]
 [ 1.57079633e+00  1.24791896e+00  1.76960635e+00 ...  8.12650557e-01
   7.62974587e-01  1.57079633e+00]
 [ 1.57079633e+00  1.44664180e+00  1.85312020e+00 ...  9.65872429e-01
   9.45539642e-01  1.57079633e+00]
 ...
 [-1.57079633e+00 -1.36169488e+00 -2.09320635e+00 ... -7.50753679e-01
  -7.29728840e-01 -1.57079633e+00]
 [-1.57079633e+00 -8.95242125e-01 -2.23058508e+00 ... -5.00845133e-01
  -4.75652448e-01 -1.57079633e+00]
 [-1.57079633e+00  0.00000000e+00  3.14159265e+00 ...  0.00000000e+00
   1.97942839e-15  0.00000000e+00]]
```



نتیجه اعمال کد برای یافتن جهت گرادیان، بر روی CannyImage:

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

d) راه حلی که با استفاده از جهت گرادیان بدست آمده در قسمت قبل برای بدست آوردن نقطه برش ساقه از گلبرگ ارائه میشود اینگونه است که ابتدا با استفاده از عملگر soble یک تصویر باینری مناسب از گل زعفران به دست می آوریم. سپس با استفاده از روش رای گیری(voting method) و جهت گرادیان بدست آمده از قسمت قبل، نقاطی که در آن تغییر شیب زیاد است پیدا میشوند و میدانیم این نقاط همان ساقه زعفران است. حال با استفاده از روش هاف(Hough)، ابتدا و انتهای این خطوط پیدا شده و نقاط ابتدایی برش داده می شود.

سوال 4:

a) مثال های خواسته شده به شرح زیر هستند:

1. تشخیص چهره: تحلیل فوریه در تشخیص چهره به عنوان یکی از مراحل مهم در پردازش تصویر مورد استفاده قرار می‌گیرد. این فرآیند شامل تحلیل فوریه تصویر و استخراج ویژگی‌های مختلفی مانند جهت‌ها و

فرکانس‌های مختلف است. این اطلاعات می‌تواند در تشخیص و شناسایی چهره‌ها و ویژگی‌های آن‌ها مؤثر باشد. به عنوان مثال، با استفاده از تحلیل فوریه می‌توان ویژگی‌های مختلفی از چهره‌ها را استخراج کرده و با الگوریتم‌های یادگیری ماشین مدل‌هایی برای تشخیص چهره آموزش داد.

2. پردازش تصویر مبتنی بر فرکانس: در پردازش تصویر مبتنی بر فرکانس، تحلیل فوریه برای تبدیل تصویر از فضای زمان به فضای فرکانسی استفاده می‌شود. این روش به ما اجازه می‌دهد تا اطلاعات بیشتری از تصویر را با استفاده از فرکانس‌های مختلف دریافت کنیم و از آن‌ها برای کاربردهای مختلفی مانند حذف نویز، تشخیص الگو، یا فیلترینگ استفاده کنیم. به عنوان مثال، در پردازش تصویر پزشکی، تحلیل فوریه می‌تواند به عنوان ابزاری مؤثر برای تشخیص و تحلیل تصاویر پرتودرمانی و MRI استفاده شود.
3. فشرده‌سازی تصویر: تحلیل فوریه در فشرده‌سازی تصویر نقش مهمی دارد. با استفاده از تبدیل فوریه، می‌توانیم اطلاعات تصویر را به صورت فرکانسی بازنمایی کنیم و بخش‌هایی که در فضای فرکانسی کم‌اهمیت هستند را حذف کنیم. این کار باعث کاهش حجم تصویر و در نتیجه کاهش پهنای باند مورد نیاز برای انتقال و ذخیره تصاویر می‌شود. این روش به طور گسترده در فشرده‌سازی تصاویر دیجیتال و استانداردهای فشرده‌سازی مانند JPEG استفاده می‌شود.
4. Image deblurring: در حل مسئله Image Deblurring، تحلیل فوریه یک نقش بسیار مهم دارد. وقتی یک تصویر ماتیسه دارد، معمولاً به این معناست که اطلاعات فرکانسی تصویر اصلی به طور غیرقابل تشخیصی

توسط ماتیسه یا blur از بین رفته است. در اینجا، تحلیل فوریه به ما کمک می‌کند تا اطلاعات فرکانسی تصویر اصلی را بازیابی کنیم و تصویر را از حالت ماتیسه‌دار به حالت اولیه برگردانیم. یکی از روش‌های معمول برای حل مسئله حذف ماتیسه، استفاده از فیلترهای فرکانسی است که با تبدیل فوریه تصویر و فیلترها، می‌توانیم تصویر را در فضای فرکانسی پاکسازی کنیم و سپس با تبدیل معکوس فوریه، تصویر پاکسازی شده را به فضای زمان بازگردانیم. فیلتر wiener هم که برای این منظور استفاده میشود نیز در قالب تبدیل فوریه پیاده سازی شده است.

b) اگر F(u,v) تبدیل فوریه دو بعدی تصویر f(x,y) باشد، آن را به صورت زیر مینویسیم:

$$F(u,v) = \sum_{x=0}^{\mathrm{M}-1} \sum_{y=0}^{\mathrm{N}-1} f(x,y)\, e^{-2\pi j(\frac{ux}{M}+\frac{vy}{N})}$$

می‌توان رابطه‌ی زیر را برای محاسبه‌ی مقدار F در نقطه (0,0) استفاده کرد:

$$F(0,0) = \sum_{x=0}^{\mathrm{M}-1} \sum_{y=0}^{\mathrm{N}-1} f(x,y)\, e^{-2\pi j(\frac{0x}{M}+\frac{0y}{N})}$$

اینجا $e^{-2\pi j(0x+0y)}$ به عنوان عبارت پایه‌ای فوریه برای نقطه (0,0) استفاده شده است. که این عبارت بسیار ساده است و معادل ۱ است. بنابراین می‌توان آن را به صورت زیر نوشت:

$$F(0,0) = \sum_{x=0}^{\mathrm{M}-1} \sum_{y=0}^{\mathrm{N}-1} f(x,y)\, e^{0}$$

$$F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

این به این معنی است که مقدار تبدیل فوریه در نقطه (0,0) برابر با مجموع مقادیر تصویر f(x,y) در تمام دامنه تصویر است.

سوال 5: موارد خواسته شده در نوتبوک گفته شده، تکمیل شد.

سوال 6: از تبدیل RANSAC برای پیدا کردن پارامترهای یک دایره در تصویر استفاده میشود. میدانیم فرمول زیر مربوط به این کار میباشد:

$1-p= (1- w^n)^k$

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

میدانیم مقدار پارامتر w نسبت تعداد نقاط inlier به تمام نقاط است. و همچنین p احتمال موفقیت یا همان احتمال یافتن یک مجموعه از نقاط بدون outlier است. و n هم کمترین مقدار نقاط مورد نیاز برای بدست آوردن است پس طبق داده های صورت سوال مقدار w=0.4 و مقدار p=0.99 و مقدار n=3 می باشد.

با جایگذاری مقادیر فوق در رابطه بالا، خواهیم داشت:

$$k = \frac{\log(1-0.99)}{\log(1-0.4^3)} = \frac{\log(0.01)}{\log(1-0.064)}$$

$$k = \frac{-2}{\log(0.936)} = \frac{-2}{-0.02872415} = 69.627819 \approx 70$$

سوال 7:

a) دو الگوریتم گفته شده را بر اساس سه جنبه مختلف مقایسه کنیم: اصول پایه آنها ، پیچیدگی محاسباتی، و ماهیت نتایجی که تولید می‌کنند.

1. اصول پایه: Hough Transform یک تکنیک استخراج ویژگی است که در تجزیه و تحلیل تصویر، بینایی کامپیوتری و پردازش تصویر دیجیتال استفاده می شود. هدف از این تکنیک یافتن نمونه های ناقص از اشیاء در یک کلاس خاص از اشکال با روش رای گیری است. این روش رای گیری در یک فضای پارامتر انجام می شود که از آن کاندیداهای شی به عنوان ماکزیمم محلی در فضای انباشته به دست می آیند. Hough Transform کلاسیک برای تشخیص خطوط با استفاده از پارامترسازی معادله خط طراحی شده است. هر نقطه در تصویر به تمام خطوطی که در فضای پارامتر از آن عبور می کنند رای می دهد، و حداکثر محلی در این فضا با خطوطی که به احتمال زیاد در تصویر وجود دارند مطابقت دارد. در حالیکه LSD یک آشکارساز بخش خط خطی در زمان است که مستقیماً روی پیکسل های خام تصاویر سیاه و سفید عمل می کند. بر خلاف Hough Transform، نیازی به تنظیم یا binning در فضای پارامتر ندارد. LSD با بررسی نواحی کوچک تصویر برای تشخیص بخش های خط کار می کند. هر بخش شناسایی شده بر اساس تعدادی از معیارها تأیید می شود تا اطمینان حاصل شود که در واقع یک خط نماینده است و برای ویژگی های مهم ادراکی بهینه می شود.
2. پیچیدگی محاسباتی: پیچیدگی محاسباتی Hough Transform نسبتا زیاد است زیرا شامل یک روش رای گیری در یک فضای پارامتر دو بعدی

است که می تواند از نظر محاسباتی گران باشد، به خصوص برای تصاویر با وضوح بالا. اگر وضوح فضای پارامتر بالا باشد، که اغلب برای تشخیص جزئیات دقیق ضروری است، پیچیدگی می تواند بیشتر افزایش یابد.

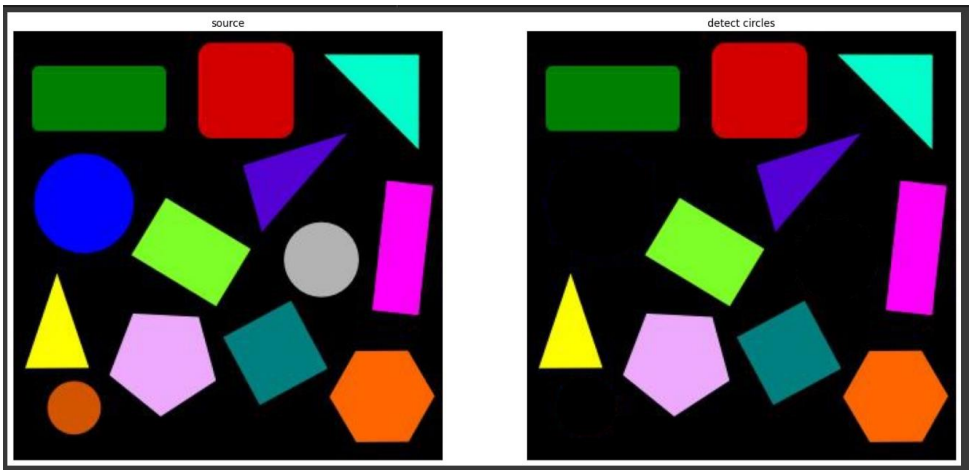
در حالیکهLSD به گونه ای طراحی شده است که از نظر محاسباتی کارآمد باشد و در زمان خطی نسبت به تعداد پیکسل های تصویر عمل می کند. مستقیماً تصویر را بدون مرحله میانی فضای پارامتر بررسی می‌کند، که آن را سریع‌تر و مناسب‌تر برای برنامه‌های بلادرنگ می‌کند.

3. ماهیت نتایج: تبدیل هاف در تشخیص خطوطی که می توانند با معادلات ریاضی در فضای پارامتر نمایش داده شوند، خوب است، حتی اگر تا حدی مبهم یا شکسته باشند. با این حال، ممکن است چندین خط کاندید را برای داده‌های پر سر و صدا شناسایی کند و برای تعیین اینکه کدام خطوط واقعاً وجود دارند، به یک آستانه نیاز دارد. نتیجه تبدیل Hough عموماً مجموعه ای از خطوط است که در کل تصویر امتداد می یابند مگر اینکه اقدامات اضافی برای تقسیم این خطوط انجام شود.

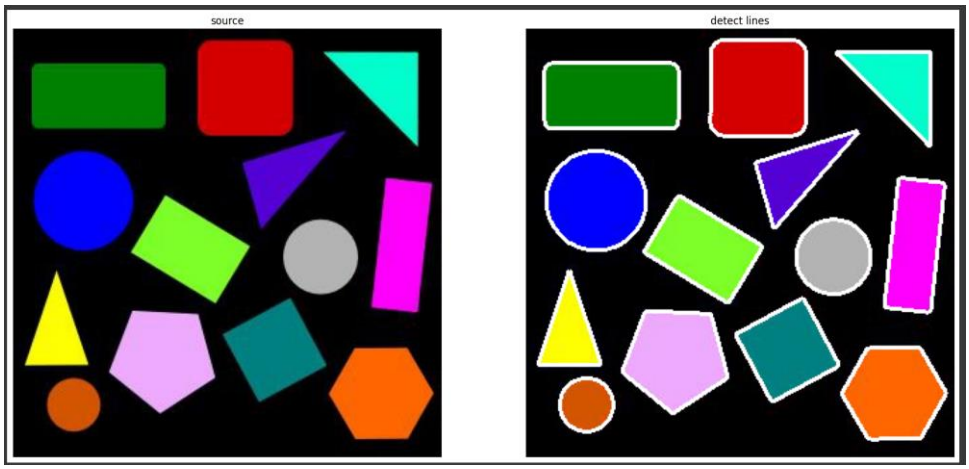
در حالیکه LSD بخش ها را به جای خطوط کامل تشخیص می دهد و اطلاعات محلی بیشتری در مورد حضور خط ارائه می دهد. این می تواند به ویژه در برنامه هایی مفید باشد که در آنها تمایز بین بخش های خط ضروری است، مانند تشخیص مرزها در تصاویر با بسیاری از اشیاء کوچک.

خروجی LSD مجموعه‌ای از بخش‌های خط با نقاط پایانی تعریف‌شده است که آن را برای برنامه‌هایی که نیاز به درک ساختار شی یا مرزها دارند، فوراً مفیدتر می‌شود.

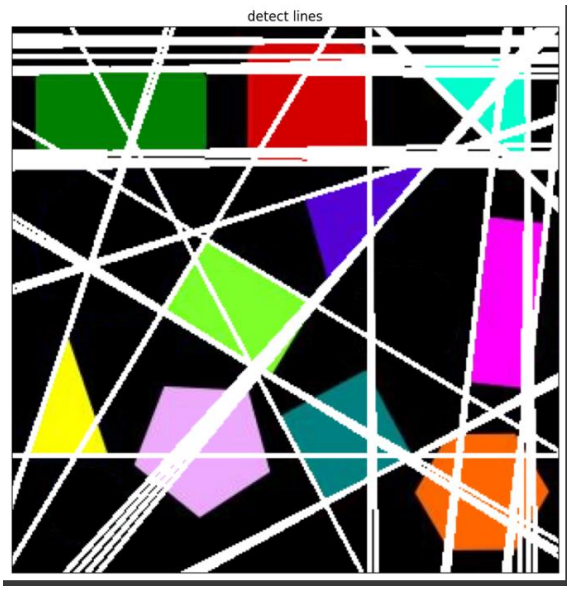
b) در این بخش کد خواسته شده با توابع گفته شده زده شد و دایره ها از تصویر حذف شد:



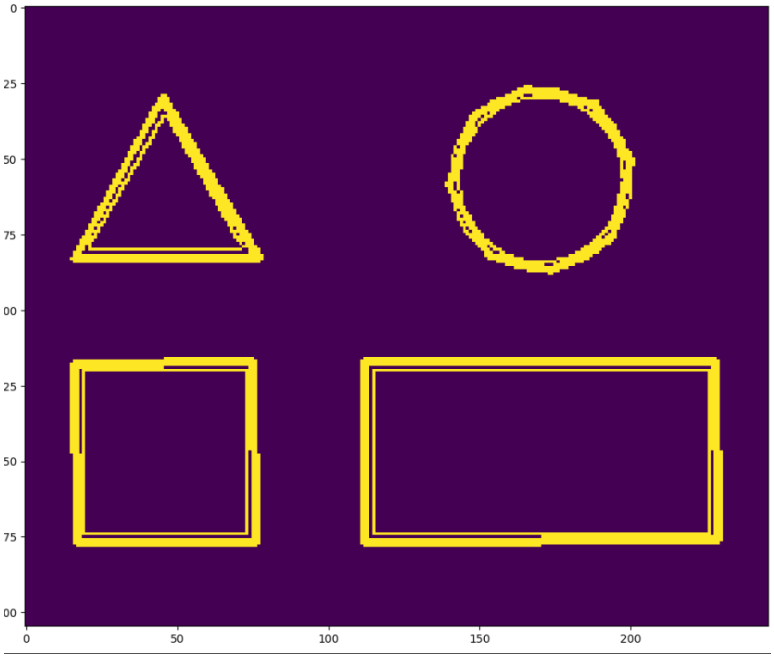
c) حال خطوط موجود در تصویر جدید را با خطوط سفید نمایش میدهیم:



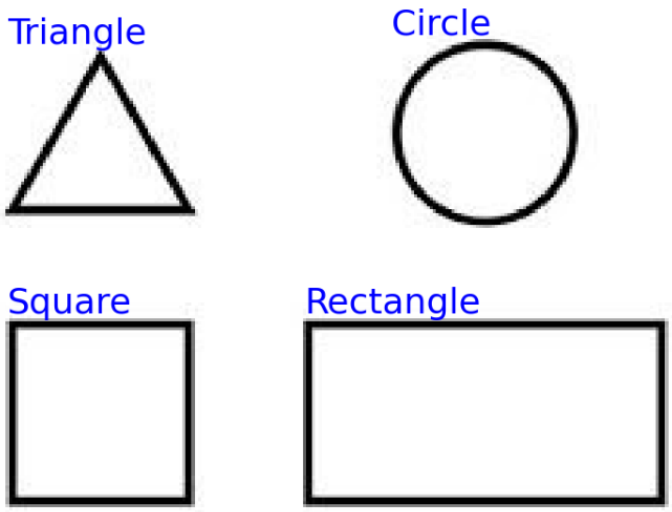
همانطور که در فایل کد مشاهده میشود، سعی شد بخش امتیازی با تابع HoughLines زده شود ولی خروجی مطلوب نبوده است و به این صورت بوده است:



d) در این بخش ابتدا تصویر خوانده شد و با استفاده از توابع گفته شده نقاط گوشه پیدا شد که به شرح زیر هستند:

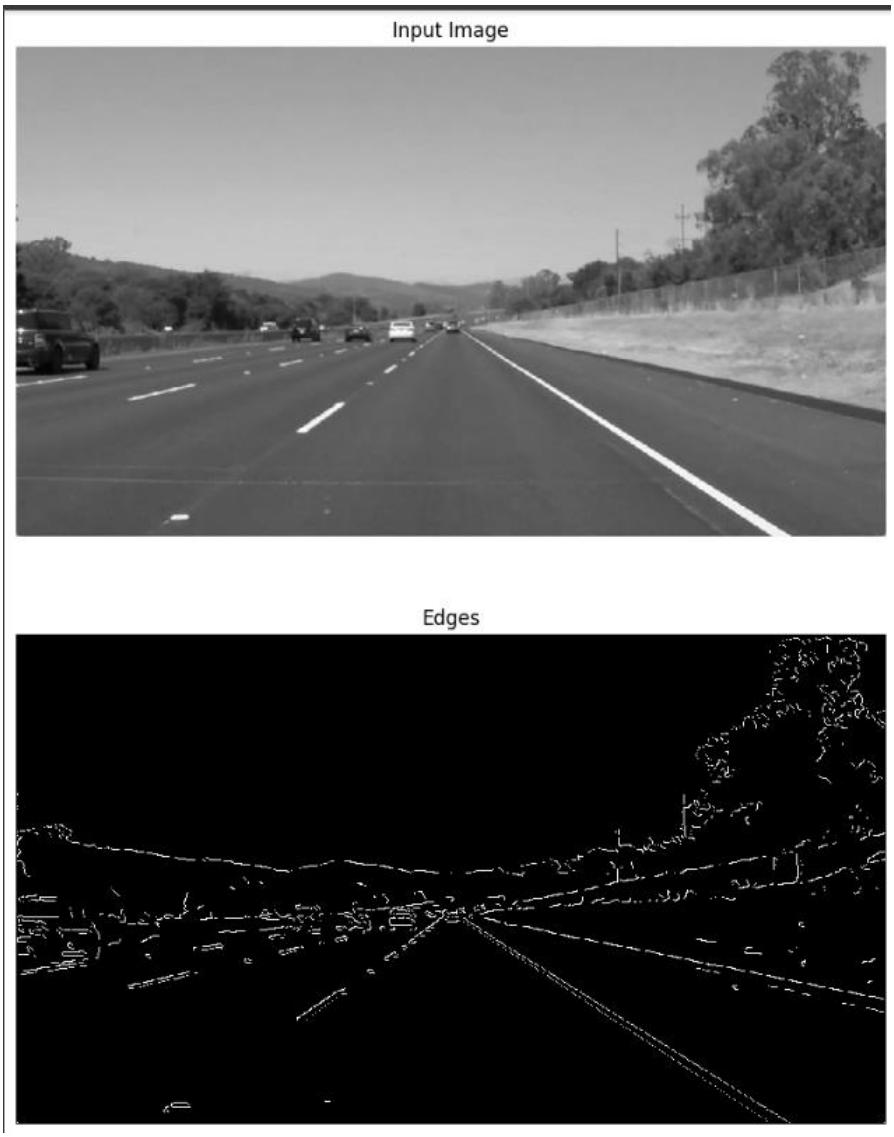


سپس کلاس هرکدام مشخص شده و در کنارش نوشته شد:

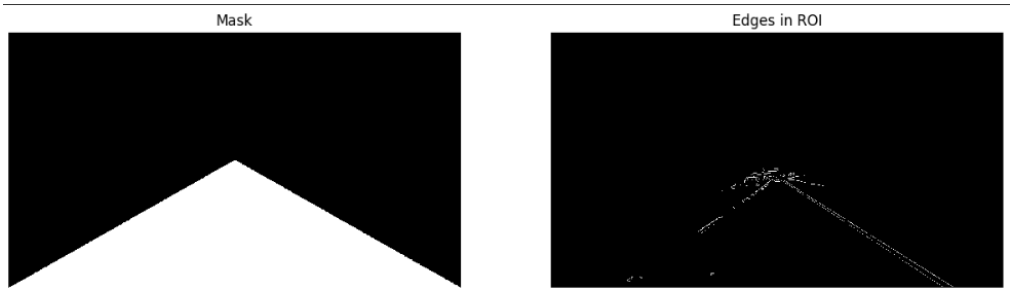


سوال 8: این سوال شامل 3 بخش مختلف است که در کد هر بخش کامنت مربوطه گذاشته شده است. خروجی ها به شرح زیر هستند:

1. Edge detection:



:ROI .2



:Fitting lines .3

پایان