تمرین سری سوم کامیار مرادیان زه آب ۹۹۵۲۲۱۰۴

سوال ۱:

بخش الف) گرادیان یک تصویر $f(x, y)$ به صورت زیر تعریف می‌شود:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

که در این رابطه g_x و g_y به ترتیب مشخص‌کنندهٔ مشتق جزئی بر محور x و بر محور y هستند، که در پایان به صورت یک بردار دو بعدی که مولفهٔ اول آن g_x بوده و مولفهٔ دوم آن g_y می‌باشد در آمده‌اند. برای محاسبهٔ مقادیر مربوط به این دو مقدار داریم:

$$g_x = \frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$$

که در تصویرها می‌توان این مقادیر را به صورت زیر تقریب زد:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x-1,y)}{2}$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1) - f(x,y-1)}{2}$$

که برای ساده‌سازی بیشتر گاهی اوقات از ۲ موجود در مخرج نیز صرف نظر می‌شود. همچنین برای پیاده‌سازی این مقادیر می‌توان از کانولوشن و کرنل‌ها نیز استفاده کرد که در نتیجهٔ آن‌ها تصویر نهایی به صورت زیر استفاده می‌شود:

$$\frac{\partial f}{\partial x} \approx f * k_x = G_x$$

$$\frac{\partial f}{\partial y} \approx f * k_y = G_y$$

برای این کرنل‌ها می‌توان از عملگرهای مختلفی نظیر Sobel و یا Prewitt استفاده کرد. در این جا برای ارائهٔ یک مثال از Sobel استفاده می‌کنیم. در این عملگر نیاز به تعریف یک کرنل به منظور انجام عملیات Convolution هستیم. این کرنل سایزهای مختلفی را می‌تواند داشته باشد که در اینجا سایز ۳ در ۳ را در نظر می‌گیریم. برای ایجاد این کرنل نیاز به دو بردار خواهیم بود که ترتیب نقش هموارسازی و مشتق‌گیری را دارند. هموارسازی به منظور کاهش نویز و در نتیجه جلوگیری از شدت گرفتن تغییرات استفاده می‌شود. همچنین هر یک از این دو در یک بعد خاص انجام می‌شود. به عنوان مثال برای محاسبهٔ گرادیان بر محور x، نیاز است که در ابتدا در طول محور yها هموارسازی را انجام دهیم و سپس در طول محور x، مشتق تصویر را برای اندازهٔ کرنل مورد نظر محاسبه کنیم. همچنین در این عملگر به شدت نور موجود بر مرکز کرنل یک میزان افزایش شدت می‌دهیم تا تاثیر پررنگ‌تری در مقدار محاسبه‌شدهٔ نهایی داشته باشد. جمع‌بندی موارد گفته شده در تصاویر زیر آورده شده است، که در آن یک کرنل ۳ در ۳ برای عملگر Sobel برای هر دو راستا، تعریف شده است:

- در راستای افقی

+1				-1	0	+1		
+2	-1	0	+1	=	-2	0	+2	
+1				-1	0	+1		

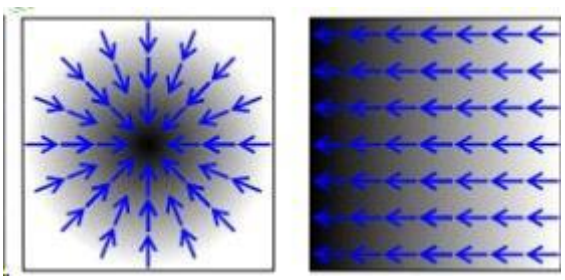
- در راستای عمودی:

-1				-1	-2	-1		
0	+1	+2	+1	=	0	0	0	
+1				+1	+2	+1		

بخش ب) با استفاده از گرادیان می‌توان مجموعه نقاطی از تصویر را که در آن‌ها نسبت به نقاط همسایۀشان تغییراتی رخ داده است را یافت و میزان تغییرات به وجود آمده در آن نقاط را به همراه جهت‌شان را محاسبه کرد. این موضوع باعث به وجود آوردن کاربردهای زیر برای گرادیان‌ها می‌شود:

- تشخیص لبه: این تکنیک شامل استفاده از بزرگی و جهت گرادیان‌ها برای تشخیص لبه‌ها در یک تصویر است. در نتیجۀ تشخیص لبه‌ها می‌توان شکل‌ها را از داخل تصویر استخراج کرد. الگوریتم‌های محبوب تشخیص لبه شامل Canny، Sobel و Prewitt است.
- تقسیم‌بندی تصویر: تکنیک‌های مبتنی بر گرادیان را می‌توان برای تقسیم‌بندی یک تصویر به مناطق بر اساس تفاوت در گرادیان بین مناطق استفاده کرد. الگوریتم حوضه آبخیز و الگوریتم K-means معمولاً برای تقسیم بندی تصویر استفاده می‌شود.
- تجزیه و تحلیل بافت: از گرادیان‌ها می‌توان برای تجزیه و تحلیل بافت یک تصویر استفاده کرد. الگوریتم Local Binary Pattern (LBP) یک الگوریتم محبوب در زمینۀ تجزیه و تحلیل بافت تصویر است که از گرادیان‌ها برای استخراج ویژگی‌های بافت از یک تصویر استفاده می‌کند.
- تشخیص و ردیابی اشیا: از گرادیان‌ها می‌توان برای تشخیص اشیاء در یک تصویر بر اساس لبه‌های آنها استفاده کرد. این رویکرد معمولا در برنامه های بینایی کامپیوتری مانند تشخیص و ردیابی اشیا استفاده می‌شود.
- بهبود تصویر: همانطور که گفته شد، از گرادیان‌ها می‌توان برای یافتن لبه‌ها و بافت‌ها در یک تصویر استفاده کرد. در نتیجۀ آن می‌توان کنتراست لبه‌های بدست آمده را افزایش داد و وضوح و کیفیت تصاویر را بهبود بخشید، به ویژه در محیط‌های کم‌نور یا کم‌کنتراست.

بخش پ) دو نمونه از اعمال گرادیان به همراه نتیجۀ به دست آمده در زیر آورده شده است:



همانطور که مشخص است گرادیان به دست آمده حاوی هم مقدار و هم جهت است. از آنجا که گرادیان حاوی مشتق جزئی در طول دو بعد است، بنابراین می‌توانیم برای محاسبهٔ اندازهٔ آن از رابطهٔ زیر استفاده کرد که برابر است با فاصلهٔ اقلیدسی یک نقطه نسبت به نقطهٔ $(0, 0)$.

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

همچنین برای ساده‌سازی این رابطه و در نتیجه سریع‌تر شدن محاسبات می‌توان از رابطهٔ زیر استفاده کرد:

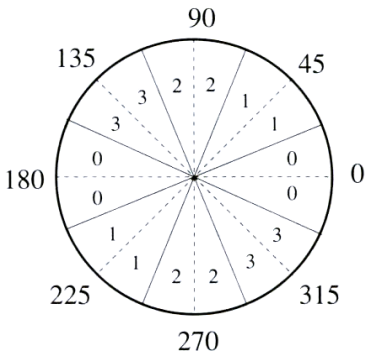
$$mag(\nabla f) = |g_x| + |g_y|$$

بخش ت) برای محاسبهٔ زاویه نیز می‌توان از معکوس تانژانت استفاده کرد.

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

بخش ث) لبه‌یاب Canny یک لبه‌یاب معروف است که با وجود گذشت مدت طولانی‌ای از معرفی آن همچنان محبوبیت خود را حفظ کرده است. این لبه‌یاب شامل ۴ مرحلهٔ اصلی است که در زیر توضیح داده شده‌اند:

- هموار کردن تصویر با استفاده از فیلتر گاوسی: در این گام با استفاده از نویزگیر گاوسی، تاثیر نویزهای موجود در تصویر را کاهش می‌دهیم. علت این کار این است که بر اثر گرادیان نویزها تقویت می‌شوند بنابراین نیاز است که قبل از این مرحله، نویزها را به طریقی کاهش بدهیم تا تنها تغییرات مورد نظر آشکار شوند.
- محاسبه گرادیان: در این مرحله به همان طریقهٔ گفته شده گرادیان تصویر را با استفاده از یک کرنل از پیش تعریف شده و مشخص محاسبه می‌کنیم. برای این منظور نیاز است که در ابتدا گرادیان را در طول راستای افقی و عمودی محاسبه کنیم و در مرحلهٔ بعدی، اندازه و در صورت نیاز جهت آن را حساب کنیم.
- حذف مقادیر غیربیشینه: با توجه به اینکه بر اثر اعمال نویزگیر گاوسی، نویزها و تغییرات کاهش می‌یابند، میزان ضخامت لبه‌ها بعد از اعمال گرادیان بیشتر از حد معقول و طبیعی می‌شود. در نتیجه نیاز است که الگوریتم NMS را بر روی یک ناحیه از تصویر اعمال کنیم. کاری که در این الگوریتم صورت می‌گیرد حذف مقادیر غیربیشینه در راستای گرادیان است. به این صورت که در ابتدا جهت گرادیان را به ۴ گروه تقسیم می‌کنیم، که این چهار گروه به صورت زیر هستند:



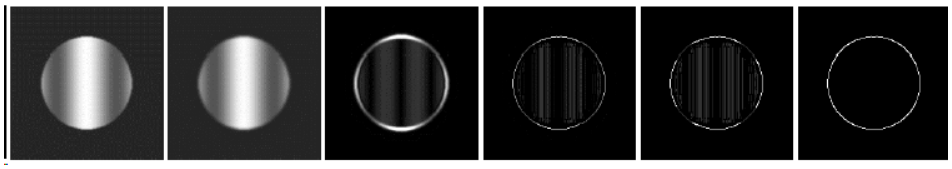
در گام بعدی، یک همسایگی به ۳ را در نظر می‌گیریم. سپس برای هر نقطه با توجه به جهت گرادیانش مشخص می‌کنیم که در کدام گروه قرار می‌گیرد. سپس در راستای گروه قرار گرفته با مقادیر موجود در آن راستا که در آن

همسایگی قرار گرفته‌اند مقایسه را انجام می‌دهیم. در صورتی که اندازهٔ گرادیان این خانه بزرگتر از آن‌ها باشد، مقدارش نگه داشته می‌شود در غیر این صورت مقدارش صفر می‌شود.

- آستانه‌گذاری دو مرحله ای: دو آستانه T_1 و T_2 را تعریف میکنیم. خواهیم داشت:
  - هر پیکسلی که اندازهٔ گرادیان آن کوچکتر از T_1 باشد به عنوان غیرلبه معرفی می‌شود.
  - هر پیکسلی که اندازهٔ گرادیان آن بزرگتر از T_2 باشد به عنوان لبه معرفی می‌شود.

مقادیر بین این دو آستانه به عنوان نقاط لبهٔ ضعیف شناخته می‌شوند. برای این نقاط به این صورت عمل می‌کنیم که ببینیم آیا به طور مستقیم یا از طریق یک پیکسل لبهٔ ضعیف دیگر، به پیکسل لبه متصل است یا خیر. اگر متصل باشد که به عنوان پیکسل لبه معرفی خواهد شد در غیر این صورت آن پیکسل را غیر لبه در نظر می‌گیریم. برای پیاده‌سازی نیز می‌توانیم یک همسایگی را در نظر بگیریم و با شروع از نقاط لبه، به ترتیب هر نقطهٔ مورد نظر رو که متصل به لبه است را لبه کنیم. در نتیجه نقاط باقیمانده را تبدیل به نقاط غیر لبه می‌کنیم.

در شکل زیر نمونه‌ای از استفادهٔ مرحله به مرحلهٔ این گام ها قابل مشاهده است: (دو مرحلهٔ نهایی مربوط به آستانه گذاری دو مرحله ای است)



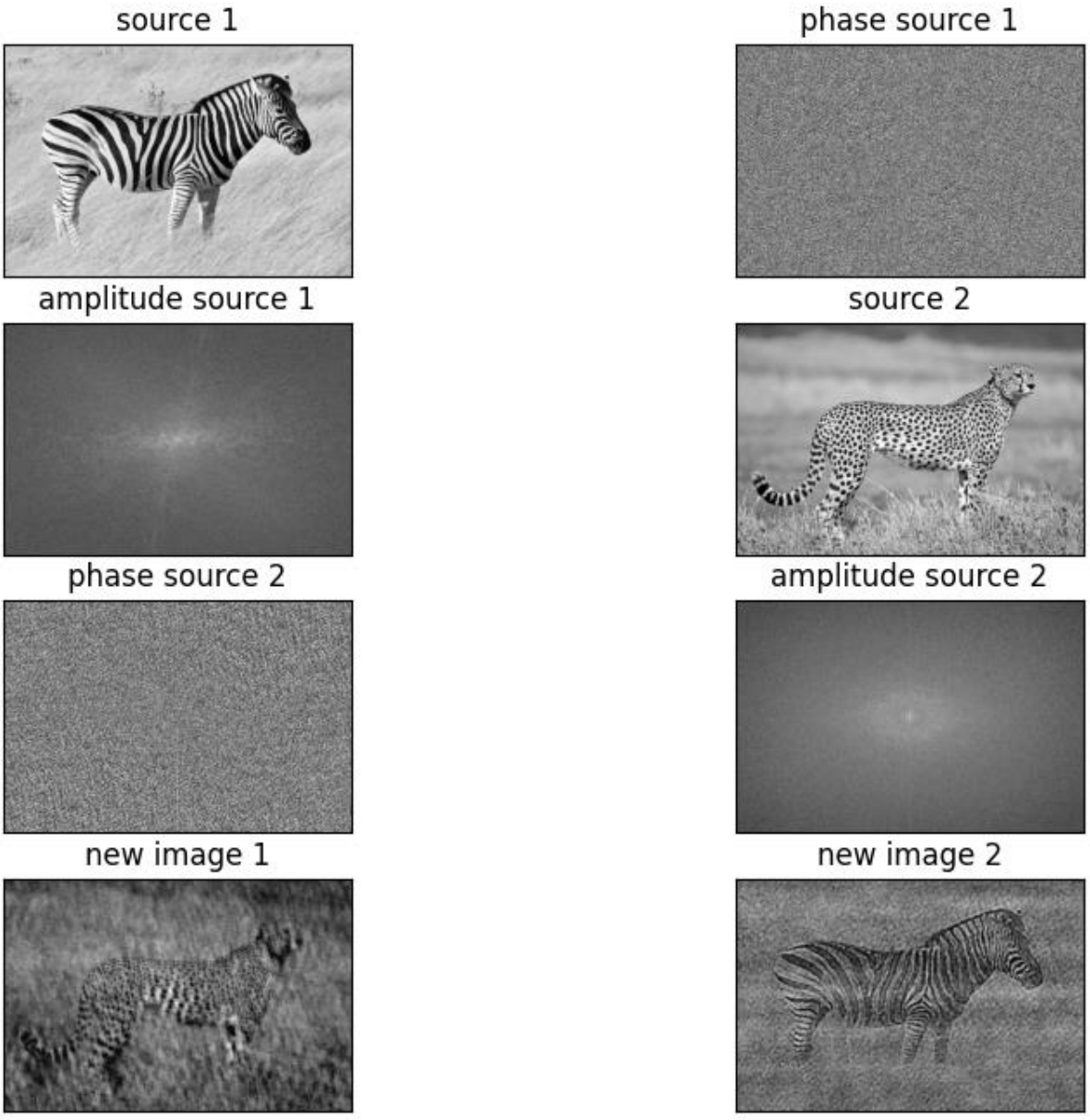
بخش ج) عملگر لاپلاسین در تئوری ریاضیات، ابزار قدرتمندی برای تشخیص لبه است، اما به دلایل زیر معمولاً در عمل استفاده نمی‌شود:

- حساسیت به نویز: اپراتور Laplacian نسبت به نویز در تصویر بسیار حساس است که می‌تواند منجر به تشخیص نادرست لبه شود. این موضوع به این دلیل است که عملگر لاپلاسین بدون هیچ‌گونه عملیات نویزگیری بر روی تصویر اعمال می‌شود و تغییرات موجود در تصویر را در هر دو راستای افقی و عمودی برجسته می‌کند و با توجه به وجود نویز در تصویر این تغییرات نیز در تصویر نهایی بدست آمده برجسته‌تر خواهند شد. این حساسیت می‌تواند منجر به تشخیص لبه‌های کاذب شود که می‌تواند کیفیت خروجی تشخیص لبه را کاهش دهد.
- عدم وجود اطلاعات جهت‌گیری: عملگر لاپلاسین هیچ اطلاعاتی در مورد جهت یال‌ها ارائه نمی‌دهد. این یک نقطه ضعف قابل توجه است، زیرا دانستن جهت یال‌ها می‌تواند برای بسیاری از کاربردها، مانند تشخیص شی یا آنالیز حرکت، حیاتی باشد. در مقابل، اپراتورهای Sobel و Canny هم اطلاعات تشخیص لبه و هم اطلاعات جهت‌گیری را ارائه می‌دهند که باعث می‌شود در عمل همه کارایی بیشتر و نیز مفیدتر باشند.
- پیچیدگی محاسباتی: عملگر لاپلاسین از نظر محاسباتی سنگین است، به ویژه هنگامی که بر روی تصاویر بزرگ اعمال می‌شود. این موضوع به این دلیل است که شامل تعداد زیادی محاسبات است که می‌تواند اجرای آن را در برنامه‌های بی‌درنگ (Real Time) دشوار کند. در مقابل، اپراتورهای Sobel و Canny کارآمدتر هستند، که آنها را برای برنامه‌های بی‌درنگ و کارهای پردازش تصویر در مقیاس بزرگ مناسب‌تر می‌سازد.

تصاویر استفاده در این سوال از اسلایدها گرفته شده‌اند.

سوال دوم:

بخش الف) نتیجه به صورت زیر بدست آمده است:



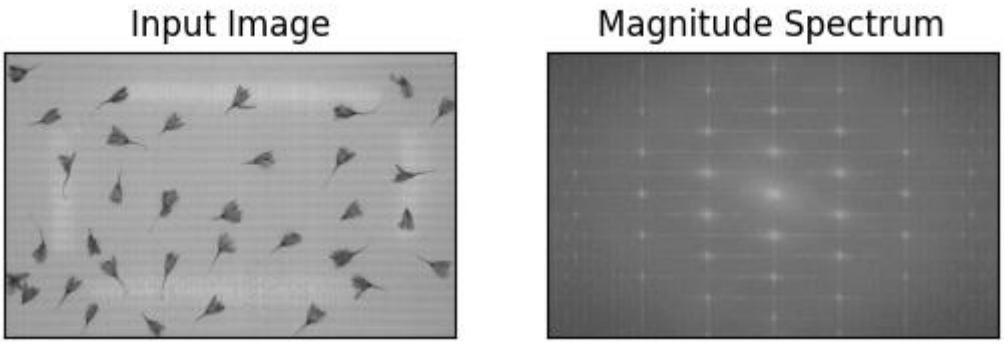
بخش ب) به طور کلی فاز یک تصویر در فضای فوریه، حاوی اطلاعاتی در مورد آرایش فضایی اجزای تصویر است و لذا برای حفظ ظاهر بصری تصویر بسیار مهم است. اگر فازها تغییر کنند، انسجام فضایی تصویر مختل می‌شود و امکان تشخیص اشیای به تصویر کشیده شده از بین می‌رود. این موضوع به این دلیل است که فاز موقعیت‌های نسبی ساختارهای تصویر را رمزگذاری و فشرده می‌کند. هنگامی که تبدیل فوریه معکوس برای بازسازی تصویر اعمال می‌شود، مراحل اصلاح شده به ترتیب متفاوتی از محتوای تصویر منجر می‌شود که منجر به یک تصویر مخدوش یا غیرقابل تشخیص می‌شود مانند آنچه که در اینجا مشاهده کردیم، که در آن شکل کلی موجود در دو تصویر به گونه‌ای با یکدیگر جایگزین شدند که باعث ایجاد اختشاش در تصویر کلی هر یک نیز شده است.

سوال سوم:

الف) در ابتدا نیاز است که تصویر را به همراه اندازهٔ تبدیل فوریهٔ آن را رسم کنیم تا بتوانیم نویزهای دوره‌ای موجود در تصویر را شناسایی کنیم. برای این منظور از قطعه کد زیر استفاده می‌کنیم. همچنین در این قطعه کد، علاوه برا دریافت اندازه، فاز را نیز ذخیره می‌کنیم تا در ادامهٔ کار جهت برگرداندن تصویر از فضای فرکانسی به فضای مکان از آن استفاده کنیم. همچنین برای مشخص شدن اندازه به طرز دقیق از لگاریتم و اسکیل کردن آن استفاده می‌کنیم.

```
# convert image to floats and do dft saving as complex output
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)

# apply shift of origin from upper left corner to center of image
dft_shift = np.fft.fftshift(dft)

# extract magnitude and phase images
mag, phase = cv2.cartToPolar(dft_shift[:,:,0], dft_shift[:,:,1])

magnitude_spectrum = 20*np.log(mag)

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

نتیجه به صورت زیر است:

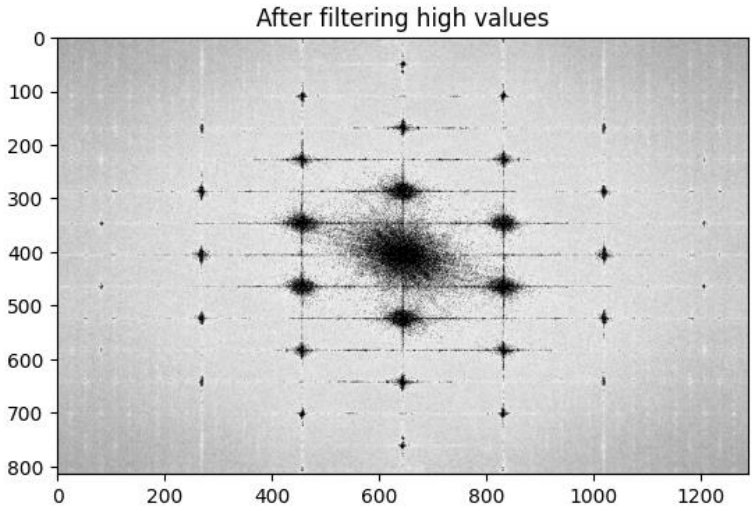


با مشاهدهٔ تصویر مربوط به اندازهٔ تصویر در فضای فرکانسی، نقاطی را می‌بینیم که به یکباره روشن شده‌اند. این نقاط همان نقاط مربوط به نویز دوره‌ای ما یا همان نقاط مشکی روی پیش‌زمینهٔ تصویر هستند. بنابراین برای حذف این نقاط نیاز است که مقدار مربوط به این اندازه‌ها را صفر کنیم.

عمل مذکور را با قطعه کد زیر انجام می‌دهیم. با استفاده از این کد، نقاطی که با مقداری با فاصلهٔ کمتر از ۱۷۰ نسبت به مقدار بیشینه دارند را صفر می‌کنیم. مقدار ۱۷۰ بعد از آزمایش مقادیر مختلف بدست آمده است.

```
mag = np.copy(magnitude_spectrum)
mag[(np.abs(mag - np.max(mag)) < 170)] = 0
plt.imshow(mag, cmap='gray')
plt.title('After filtering high values')
```

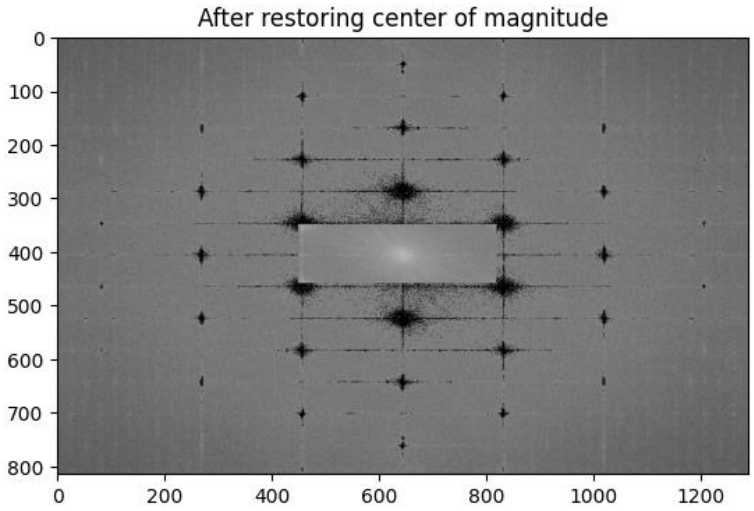
نتیجهٔ استفاده از این کد به صورت زیر است:



با اعمال قطعه کد بالا تمام نقاط مورد نظر صفر شدند. اما مشکلی که به وجود آمده است، صفر شدن نقاط مرکزی است که مربوط به محل قرارگیری اصل تصویر و به عبارتی میانگین شدت روشنایی تصویر می‌باشند. بنابراین نیاز است که این مقادیر را بازیابی کنیم. برای این منظور با توجه به پیکسل‌هایی که برروی نمودار قرار گرفته‌اند، این نقاط مرکزی را با پیکسل‌های متناظرشان از مقدار اولیهٔ تصویر جایگزین می‌کنیم. برای این منظور از قطعه کد زیر استفاده می‌کنیم:

```
mag[350:460,450:820] = magnitude_spectrum[350:460,450:820]
plt.imshow(mag, cmap='gray')
plt.title("After restoring center of magnitude")
```

نتیجه نیز به صورت زیر می‌شود:

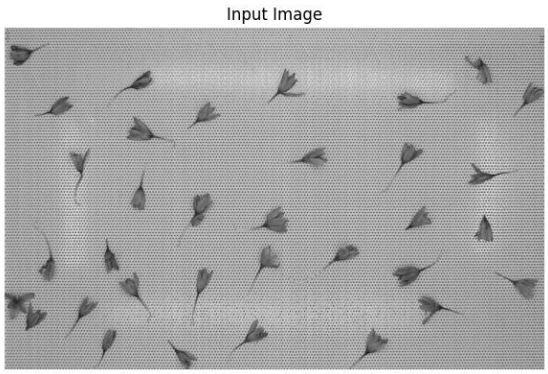


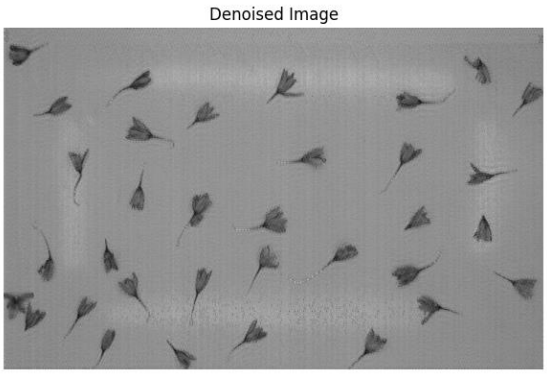
حال که این مقادیر بازیابی شده‌اند می‌توانیم با استفاده از فاز اولیهٔ تصویر، تصویر را از حوزهٔ فرکانس به حوزهٔ مکان ببریم:

```python
original_mag = np.exp(mag/20)

# convert magnitude and phase into cartesian real and imaginary components
real, imag = cv2.polarToCart(original_mag, phase)

# combine cartesian components into one complex image
back = cv2.merge([real, imag])

# shift origin from center to upper left corner
back_ishift = np.fft.ifftshift(back)

# do idft saving as complex output
img_back = cv2.idft(back_ishift)

# combine complex components into original image again
img_part_a = cv2.magnitude(img_back[:,:,0], img_back[:,:,1])

# Normalize back to 8bit
min, max = np.amin(img_back, (0,1)), np.amax(img_back, (0,1))
img_part_a = cv2.normalize(img_part_a, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)

plt.figure(figsize=(16,8))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.axis('off')
plt.subplot(122),plt.imshow(img_part_a, cmap = 'gray')
plt.title('Denoised Image'), plt.axis('off')
plt.show()
```

برای بازگردانی یک تصویر از حوزهٔ فرکانس به حوزهٔ مکان در ابتدا اندازهٔ اولیه را با انجام معکوس آنچه که پیشتر بر آن انجام شده است، آن را به مقدار اولیه بازمی‌گردانیم. سپس با استفاده از cv2.polarToCart آن را از حوزهٔ قطبی به حوزهٔ کارتزین می‌بریم. با این کار بخش حقیقی و موهومی تصویر را بدست می‌آوریم. در ادامه این دو بخش را با هم ترکیب می‌کنیم تا بتوانیم در ادامه از توابع np.fft.ifftshit استفاده کرده، که نقطهٔ صفر میانگین را جابه‌جا می‌کند، و سپس از cv2.idft استفاده کنیم تا تصویر را به حوزهٔ مکان بازگردد. همچنین در پایان یک عملیات نرمال‌سازی را برای مقادیر مربوط به پیکسل‌های تصویر انجام دهیم تا مقادیرش بین صفر تا ۲۵۵ قرار گیرد.

تصویر بدست آمده بعد از اعمال تغییرات بالا به صورت زیر است:





بخش ب) در ابتدا تصویر را از یک فیلتر گاوسی می‌گذرانیم تا نویزهای گاوسی تصویر از بین بروند. سپس با استفاده از لبه‌یاب Canny لبه‌های موجود در تصویر را شناسایی می‌کنیم:

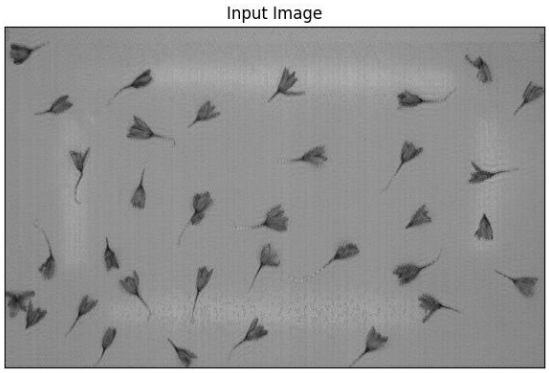
```python
blur = cv2.GaussianBlur(img_part_a, (7, 7), 0) # Smooth image using guassian kernels
img_part_b = cv2.Canny(blur, 20, 120)

plt.figure(figsize=(16,8))
plt.subplot(121),plt.imshow(img_part_a, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_part_b, cmap = 'gray')
plt.title('Denoised Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

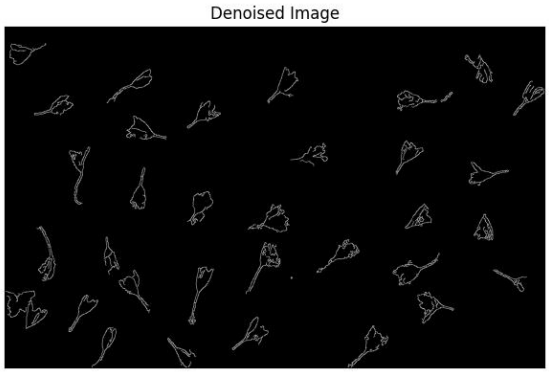
توضیحات در رابطه با پارامترهای تابع cv2.Canny:

- پارامتر اول تصویر هموار شده است.
- پارامتر دوم مربوط به حداقل شدت گرادیان پیکسل جهت انتخاب شدنش است. به عبارتی این مقدار همان T_1 یا آستانهٔ اول است. تمام مقادیر زیر این آستانه به عنوان نقاط غیر لبه شناسایی خواهند شد و حذف می‌شوند.
- پارامتر سوم مربوط به حداکثر شدت گرادیان پیکسل جهت انتخاب شدنش است. به عبارتی این مقدار همان T_2 یا آستانهٔ دوم است. تمام مقادیر بالای این آستانه به عنوان نقاط لبه شناسایی خواهند شد و نقاط بین این مقدار و مقدار T_1 به عنوان نقاط لبهٔ ضعیف شناسایی می‌شوند. در صورتی که این نقاط به صورت مستقیم یا غیرمستقیم و از طریق سایر نقاط لبهٔ ضعیف به نقاط لبهٔ قوی متصل باشند، به عنوان لبه شناسایی خواهند شد.

هر دو مقدار آستانه به آزمون و خطا انتخاب شده‌اند. به ازای مقادیر کوچکتر برای T_2 نویزها هم به عنوان لبه شناسایی می‌شدند و خطوط لبهٔ بیشتری در تصویر می‌بود در حالی که با افزایش آن، لبه‌های مربوط به زعفران‌ها شناسایی نمی‌شدند. برای آستانهٔ اول نیز مقادیر کوچکتر باعث می‌شد که تعدادی از نویزها و لبه‌های خیلی ضعیف نیز به مجموعهٔ لبه‌ها اضافه شوند که نیازی به حضورشان نبود. به ازای مقدار بزرگ‌تر نیز، لبه‌های زعفران‌ها به خوبی شناسایی نمی‌شدند.

نتیجهٔ نهایی به صورت زیر است:

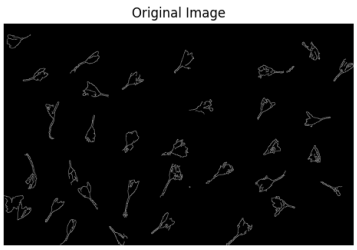


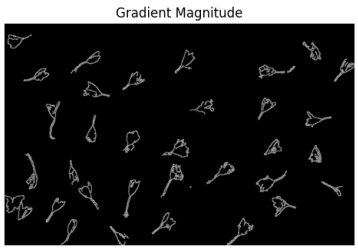


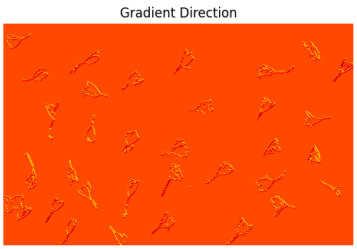
بخش پ جهت محاسبهٔ مشتق از عملگر Sobel استفاده شده است و در ادامه از تابع arctan2 استفاده شده است.

```python
sobelx = cv2.Sobel(img_part_b, cv2.CV_64F, 1, 0, ksize=3)  # Gradient in X direction
sobely = cv2.Sobel(img_part_b, cv2.CV_64F, 0, 1, ksize=3)  # Gradient in Y direction



magnitude = np.sqrt(sobelx**2 + sobely**2)
direction = np.arctan2(sobely, sobelx)  # Direction of gradients
```

نتیجه به صورت زیر است:







بخش ت:

به منظور از تشخیص ساقهٔ زعفران نسبت به گل آن می‌توانیم از گرادیان و زاویهٔ آن استفاده کنیم. برای این منظور بعد از بدست آوردن جهت گرادیان مربوط به هر پیکسل مربوط به تصویر، می‌توانیم از الگوریتم‌هایی مثل LSD استفاده کنیم تا اینکه خطوط موجود در شکل را تشخیص دهیم. برای این منظور با استفاده از جهت گرادیان، هر پیکسل را در گروه‌های مختلف قرار می‌دهیم و سپس این پیکسل‌ها را گروه‌بندی می‌کنیم. بعد از این که این کار را انجام دادیم با استفاده از گروه تشکیل شده خطوط موجود در تصویر را تشخیص می‌دهیم. برای اینکه خطوط صاف‌تر را نیز تشخیص دهیم می‌توانیم از قوانین سخت‌گیرانه‌تری استفاده کنیم تا اینکه تنها پیکسل‌هایی که اختلاف کمی در جهت گرادیان آن‌ها وجود داشته باشد، انتخاب شوند. پس از اینکه خطوط تشخیص داده شدند و هر یک از آن‌ها تشکیل شدند، می‌توانیم یک شرط دیگر نیز بر روی طول هر یک از خطوط تشخیص داده شده اعمال کنیم تا تنها خطوط با طول مشخص به عنوان ساقهٔ زعفران شناخته شوند.

سوال چهارم:

بخش الف) سه ویژگی در زیر آورده شده‌اند (سه مورد اول با یکدیگر در ارتباط هستند و تفاوت چندانی با هم ندارند.):

- هموارسازی تصویر: برای این منظور در ابتدا تبدیل فوریه تصویر محاسبه می‌شود. پس از انجام این محاسبات، با استفاده از فیلترهای پایین‌گذر، بخش‌های از تصویر را که تغییرات آن‌ها شدید بوده است از تصویر حذف می‌شوند. با حذف این بخش‌ها گرچه بخشی از جزئیات تصویر از دست می‌رود اما با اینحال بخش اعظمی از نویزها که در این نواحی قرار داشتند از بین می‌روند.
- تشخیص لبه‌ها: برای تشخیص لبه‌های موجود در تصویر پس از این که تصویر را به معادل فوریهٔ خود تبدیل کردیم، یک فیلتر بالاگذر بر روی آن می‌اندازیم تا جاهایی از تصویر که در آن‌ها تغییرات بوده است، مشخص شوند. تغییرات حاصل همان لبه‌های موجود در تصویر به همراه نویزها می‌باشند.
- تیزسازی تصویر: جهت تیزکردن لبه‌ها نیاز است که از نتیجهٔ کاربرد قبلی، یعنی تشخیص لبه‌ها استفاده کنیم. برای این کاربرد نیاز است که از یک فیلتربالاگذر به همراه مقداری شیفت استفاده کنیم. در نتیجهٔ این شیفت، بخش‌های لبهٔ موجود در تصویر نسبت به سایر بخش‌های موجود در آن برجسته‌تر شده، در حالی که می‌توان سایر بخش‌ها را به همان صورت پیشین حفظ کرد.
- حذف نویزهای متناوب: مشکلات بینایی کامپیوتری که شامل الگوهای تناوبی یا ساختارهای تکراری می‌شوند، از به‌کارگیری تکنیک‌های تحلیل فوریه سود زیادی می‌برند. به دلیل ماهیت توانایی تبدیل فوریه برای استخراج فرکانس های غالب موجود در سیگنال، تشخیص دوره‌های درون سیگنال‌ها بسیار آسان‌تر می‌شود. در نتیجه، شناسایی این فرکانس‌ها، تشخیص تناوب زیربنایی در تصاویر یا ویدیوها را امکان‌پذیر می‌سازد و فرآیندهای تشخیص الگوی بیشتر یا تشخیص ناهنجاری را تسهیل می‌کند. برای این منظور نیاز است که پس از محاسبهٔ تبدیل فوریهٔ یک تصویر، و تشخیص تناوب موجود در آن، یک فیلتر ایجاد کنیم که تغییرات موجود بر روی الگوی تناوب را از میان ببرد. به عبارتی این فیلتر مقادیر موجود بر روی الگو را صفر می‌کند در حالی که با سایر تغییرات موجود در تصویر کاری ندارد. در پایان و پس از تبدیل فوریهٔ معکوس تصویر اولیه بدون نویزهای متناوب به دست می‌آید.
- استخراج ویژگی: تحلیل فوریه همچنین برای استخراج ویژگی در عملیات‌های بینایی کامپیوتری‌ای مانند تشخیص و طبقه‌بندی اشیا استفاده می‌شود. ویژگی‌های استخراج شده از یک تصویر، مانند اطلاعات بافت یا شکل را می‌توان در حوزهٔ فرکانس با استفاده از تکنیک‌هایی مانند تبدیل فوریه نشان داد. با تجزیه و تحلیل محتوای فرکانس مناطق مختلف تصویر، می‌توان ویژگی‌های معناداری را برای مشخص کردن اشیاء یا الگوهای موجود در تصویر استخراج کرد. این ویژگی‌ها اغلب نسبت به تغییرات در نور، مقیاس و جهت‌گیری در مقایسه با ویژگی‌های حوزهٔ فضایی قوی‌تر هستند، و آنها را برای کارهایی که نیاز به نمایش‌های ثابت دارند ارزشمند می‌سازد. علاوه بر این، استخراج ویژگی در حوزه فرکانس می‌تواند منجر به نمایش کارآمدتر و فشرده‌تر تصاویر، تسهیل پردازش سریع‌تر و کاهش پیچیدگی محاسباتی در مراحل بعدی تجزیه و تحلیل شود.

بخش ب)

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

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

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

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

بنابراین $F(0,0)$ معادل است با جمع تمام درآیه‌های موجود در تصویر اصلی و در نتیجه می‌توان با استفاده از این مقدار میانگین شدت روشنایی موجود در هر کانال از تصویر را بدست آورد.

تمرین سری سوم کامیار مرادیان زه آب ۹۹۵۲۲۱۰۴

سوال پنجم:

در نوتبوک مربوط کدها قرار گرفته‌اند.

سوال ششم:

برای محاسبهٔ مورد خواسته شده نیاز است که از رابطهٔ زیر استفاده کنیم که به صورت زیر است:

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

در این رابطه w درصد نقاط Inlier نسبت به کل نقاط موجود در تصویر بوده و همچنین p احتمال مطلوبی است که در پی بدست آوردن آن هستیم. با قرار دادن w=0.4 و نیز خواستن حداقل p=0.99، داریم:

$$k = \left\lceil \frac{\log(1-0.99)}{\log(1-0.4^2)} \right\rceil \cong \left\lceil \frac{-2}{-0.076} \right\rceil = \lceil 26.31 \rceil = 27$$

بنابراین برای بدست آوردن مقدار p=0.99 نیاز است که $k \geq 27$ باشد.

سوال هفتم:

بخش الف) در زیر، در حوزه‌های زیر با هم مقایسه شده‌اند.

- الگوریتم و شیوهٔ کار: الگوریتم تبدیل هاف بر مبنای انتقال تصویر به یک فضای دیگر و همچنین رای‌گیری از تصویر است. این درحالی است که در الگوریتم *LSD* از جهت گرادیان تصویر و نیز تقسیم‌بندی تصویر استفاده می‌شود.
- نیازمندی‌های ورودی: یک تفاوت کلیدی بین این دو الگوریتم در نیازهای ورودی آنها نهفته است. تبدیل هاف معمولاً روی یک تصویر باینری عمل می‌کند، جایی که شدت پیکسل نشان می‌دهد که آیا یک ویژگی وجود دارد یا خیر. به عبارتی نیاز است که در ابتدا با استفاده از یک الگوریتم لبه‌یابی، مانند لبه‌یاب *Canny*، لبه‌های تصویر را بیابیم و سپس تصویر لبه‌ها را به الگوریتم تبدیل هاف تحویل دهیم. از طرف دیگر، *LSD* یک تصویر در مقیاس خاکستری را به عنوان ورودی می‌گیرد و سپس از اطلاعات گرادیان استفاده می‌کند و نه اینکه صرفاً بر داده‌های بدست آمده از الگوریتمی دیگر تکیه کند. این تمایز نشان می‌دهد که *LSD* به طور بالقوه می‌تواند اطلاعات ساختاری دقیق‌تری را در مقایسه با تبدیل هاف در هنگام برخورد با تصاویر حاوی سطوح مختلف شدت استخراج کند.
- پارامترها و تنظیم آن‌ها: یکی دیگر از جنبه های مهمی که باید در نظر گرفته شود، سطح مداخله مورد نیاز کاربر در طول پردازش است. در حالی که هر دو روش شامل درجات خاصی از تنظیم پارامتر هستند، *LSD* به طور کلی به ورودی های دستی کمتری نسبت به تبدیل هاف نیاز دارد. *LSD* به گونه‌ای طراحی شده است که بدون تنظیم پارامترهای گسترده به طور موثر عمل کند و کاربرد آن را در برنامه‌های مختلف آسان‌تر می‌کند. در همین حال، تبدیل هاف اغلب نیاز به تنظیم دقیق پارامترهای متعدد مانند *rho*، تتا جهت گسسته‌سازی فضای تازه، و نیز تنظیم مقادیر آستانهٔ انتخاب نهایی برای دستیابی به عملکرد بهینه دارد.
- سرعت پردازش: الگوریتم *LSD* به دلیل سادگی و تواناییش در مدیریت تصاویر پیچیده اغلب سریع‌تر از تبدیل هاف عمل می‌کند و از قابلیت‌های استفاده در کاربردهای *Real Time* برای وظایف استخراج خط نیز برخوردار است. این در حالی است که الگوریتم تبدیل هاف، به توجه به انجام مراحل لبه‌یابی و نیز رای‌گیری مدت زمان بیشتری را جهت اجرا شدن نیاز دارد.
- خروجی‌ها و کیفیت آن‌ها: تبدیل هاف، کنترل بهتری بر انواع خطوط شناسایی شده از طریق تنظیمات پارامتر فراهم می‌کند و به کاربران اجازه می‌دهد تا ساختارهای خاصی را در یک تصویر هدف قرار دهند. علاوه بر این، ماهیت قطعی تبدیل هاف خروجی‌های ثابت را با وجود تغییرات در شرایط عملیاتی تضمین می‌کند. برعکس، *LSD* عناصر احتمالی را در بر می‌گیرد که منجر به تفاوت‌های بالقوه در نتایج تحت شرایط یکسان می‌شود. از طرفی الگوریتم *LSD* به طور ویژه برای تشخیص خط مورد استفاده قرار می‌گیرد و در صورت وجود نویز و یا کیفیت پایین تصویر، بتواند بهتر از تبدیل هاف در حوزهٔ تشخیص خط عمل کند. این درحالی است که قابلیت تشخیص شکل در آن بسیار پایین است.

سایر بخش‌ها در نوتبوک مربوطه پاسخ داده شده‌اند.

سوال هشتم:

در نوتبوک مربوط کدها قرار گرفته‌اند.