داکیومنت تکلیف شماره ۲

■ سوال ۱

ج)

```
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)
```

پارامترها به شرح زیر هستند:

- **gray**: تصویر ورودی به عنوان تصویر خاکستری.
- **cv2.CV_64F**: نوع داده برای جواب عملیات سابل.
- **0**: درجه تفاضل مورد استفاده برای محور **x** .
- **1**: درجه تفاضل مورد استفاده برای محور **y** .
- **ksize=3**: اندازه هسته کرنل سابل.

■ سوال ۲

الف) در این کد، یک تصویر خاکستری با استفاده از تابع **cv2.imread** و با فرمت **jpg** با نام **"saffrun.jpg"** خوانده شده است و در متغیر **img** ذخیره شده است.

سپس تابع **remove_noise_fft** تعریف شده است که تصویر خاکستری را به عنوان ورودی دریافت می کند و مراحل حذف نویز را با استفاده از تبدیل فوریه انجام می دهد.

در ابتدا تصویر ورودی به تابع داده شده و با استفاده از تابع **np.fft.fft2** و **np.fft.fftshift** تبدیل فوریه آن محاسبه می شود و برای رسم نمودار مقدار طیف تبدیل فوریه در محور های فرکانسی **x** و **y** ، مدول آن به دست آمده و با استفاده از تابع **np.log(np.abs(fshift)) * 20** به مقدار دسیبل تبدیل می شود. سپس نمودار مقدار طیف در خروجی با عنوان **"Magnitude Spectrum"** ذخیره می شود.

سپس با استفاده از تصویر مقدار طیف و یک ماسک مربعی در حوالی مرکز تصویر، یک ماسک را ایجاد می کنیم که به ازای هر پیکسل از تصویر، مقدار ۰ یا ۱ را در بر می گیرد. سپس با استفاده از این ماسک، اطلاعاتی که در حوالی مرکز تصویر هستند و بخشی از نویز را شامل می شوند، حذف می شوند و تصویر فیلتر شده بازیابی می شود.

در ادامه با استفاده از تصویر فیلتر شده، تصویر **Canny Edge Detection** با استفاده از تابع **cv2.Canny** حساب می شود. برای این منظور، ابتدا با استفاده از یک آستانه، پیکسل هایی که کمتر از آن آستانه هستند به صفر تبدیل

شده و پیکسل هایی که بیشتر یا مساوی آستانه هستند به ۲۵۵ تبدیل می شوند و سپس تابع **Canny** روی تصویر به دست آمده اعمال می شود.

ب)

```
canny = cv2.Canny(img_canny.astype(np.uint8), 70, 100)
```

پارامترهای این الگوریتم عبارتند از:

۱. **img_canny**: تصویری که برای تشخیص لبه از آن استفاده می‌شود.

۲. **threshold**: آستانه بالا و پایینی برای روش .**Canny** هر پیکسلی که مقدار آن بیشتر از آستانه بالا باشد، به عنوان یک لبه قرار داده می‌شود. هر پیکسلی که مقدار آن کمتر از آستانه پایین باشد، به عنوان یک لبه رد شده و ردیابی لبه برای آن انجام نمی‌شود.

۳. **70**: پارامتر اول در روش **Canny** که برای تشخیص لبه استفاده می‌شود.

۴. **100**: پارامتر دوم در روش **Canny** که برای تشخیص لبه استفاده می‌شود.

■ سوال ۳

الف) فیلترهای پایین گذر (**LPF**) و بالاگذر (**HPF**)، دو نوع اصلی فیلتر در پردازش تصویر هستند که هر کدام دارای کاربردهای خاص خود هستند.

۱- فیلتر پایین گذر (**LPF**):

این فیلترها با حذف فرکانسهای بالای تصویر، تصویر را مات و نامشخص تر می کنند. بنابراین، معمولا برای حذف نویز و صاف سازی تصویر به کار می روند. برای مثال، در تصاویر عکس مبتنی بر واقعیت مجازی، برای حذف جزئیات و ضروری های نامطلوب، به ویژه در مواردی که باید مداوم از تصویر استفاده شود مانند دوربین های مداربسته، از فیلتر پایین گذر استفاده می شود.

۲- فیلتر بالاگذر (**HPF**):

این فیلترها با حذف فرکانسهای پایین تصویر، جزئیات و جزئیات تصویر را برجسته می کنند. بنابراین، معمولا برای تشخیص لبه ها، تشخیص نقاط ثابت و حذف پس زمینه نامطلوب استفاده می شوند. برای مثال، در تصاویر پزشکی مانند سی تی اسکن، برای تشخیص لبه ها و جزئیات مورد نظر در تصویر، از فیلتر بالاگذر استفاده می شود.

بنابراین، برای پردازش تصاویر، فیلترهای پایین گذر و بالاگذر برای اهداف مختلف مورد استفاده قرار می گیرند. به طور کلی، فیلترهای پایین گذر برای حذف نویز و صاف کردن تصویر و فیلترهای بالاگذر برای تشخیص جزئیات و جزئیات در تصاویر استفاده می شوند.

ب) فیلتر بالاگذر. فیلتر بالاگذر فرکانسهای پایین تصویر را حذف کرده و فقط فرکانسهای بالایی را باقی می گذارد. این کار باعث برجسته شدن جزئیات و لبه های تصویر می شود.

ج) نویز جمع شونده (additive noise) به معنای اضافه شدن سیگنال تصویر به نویز است. در این نوع نویز، مقادیر پیکسل های تصویر از مقادیر واقعی خود خارج می شوند و مقدار نویز به آنها اضافه می شود. بنابراین، در این نوع نویز، نویز مستقل از سیگنال تصویر است و این بدان معناست که می توان آن را با فیلترهای خطی، مانند فیلتر میانگین، حذف کرد.

در مقابل، نویز ضرب شونده (multiplicative noise) به معنای ضرب کردن سیگنال تصویر با نویز است. در این نوع نویز، مقادیر پیکسل های تصویر در تصویر نهایی تغییر می کنند و نویز به صورت ضربی با سیگنال تصویر درآمیخته می شود. بنابراین، در این نوع نویز، نویز وابسته به سیگنال تصویر است و با استفاده از فیلترهای خطی سنتی نمی توان آن را به خوبی حذف کرد. برای حذف نویز ضرب شونده، می توان از روش هایی مانند فیلترهای غیرخطی، مانند فیلتر مدیانی، استفاده کرد.

برای حذف نویز جمع شونده، علاوه بر فیلترهای خطی، می توان از الگوریتم های پیشرفته تری مانند فیلترهای کانی ماکس، فیلترهای کلمپینگ و غیره استفاده کرد. برای حذف نویز ضرب شونده، می توان از الگوریتم های پیشرفته تصویربرداری مانند الگوریتم های بازسازی تصویر بر پایه فضای بردارهای توزیع شده استفاده کرد. همچنین، استفاده از فیلترهای غیرخطی مانند فیلتر مدیانی و فیلتر بیشینه قابل اجتناب است.

د) نویز نمک و فلفل یک نوع نویز جمع شونده است که در تصاویر دیجیتال به صورت تصادفی به پیکسل های تصویر اضافه می شود. این نوع نویز باعث ایجاد نقاط سفید یا سیاه در تصویر می شود که به نوعی با نقاط سفید و سیاه روی کاغذ نمک و فلفل شبیه است، بنابراین به این نام مشهور شده است.

برای حذف نویز نمک و فلفل، معمولاً از فیلتر مدیانی استفاده می شود. فیلتر مدیانی یک فیلتر غیرخطی است که مقدار هر پیکسل با مقدار مد (عدد وسطی) از مقادیر همسایه آن جایگزین می شود. این فیلتر موجب حذف نویزهای جمع شونده از جمله نویز نمک و فلفل می شود، بدون اینکه شدت تصویر را به شدت کم کند. همچنین، از فیلترهای دیگری نیز می توان برای حذف نویز نمک و فلفل استفاده کرد، مانند فیلتر میانگین و فیلتر بیشینه.

به عنوان یک نکته مهم، باید توجه داشت که استفاده از فیلترهای مقطعی (مانند فیلتر مدیانی) در بعضی موارد می تواند باعث ایجاد افت کیفیت در تصویر شود و جزئیات تصویر را از بین ببرد. بنابراین، در مواردی که تحلیل دقیق تصویر و حفظ جزئیات آن ضروری است، باید از فیلترهایی مانند فیلتر گوسی استفاده کرد که می توانند نویزها را حذف کنند، در عین حال جزئیات تصویر را حفظ کنند.

■ سوال ۴

الف) تبدیل فوریه تصویر (FFT) یک تبدیل مستقیم از دامنه زمانی تصویر به دامنه فرکانسی است که به کار می‌رود تا بتوانیم اطلاعات فرکانسی تصویر را به دست آوریم.

به طور کلی، مقدار تبدیل فوریه تصویر در هر نقطه با مقدار پیکسل متناظر در آن نقطه در تصویر مرتبط است. با فرض تصویری به اندازه N × N، نقطه مبدا تصویر (۰،۰) معادل با پیکسل بالا سمت چپ تصویر است. در نتیجه، مقدار تبدیل فوریه تصویر در نقطه مبدا با مجموع مقادیر پیکسل های تصویر برابر است.

ب) برای اینکه بتوانیم مقادیر تبدیل فوریه تصویر را بدست آوریم، ابتدا باید تصویر را به فضای فرکانسی تبدیل کنیم. برای این منظور، از تبدیل فوریه دو بعدی استفاده می‌کنیم. با توجه به تعریف تبدیل فوریه، مقادیر تبدیل فوریه تصویر برابر با:

$$
\begin{aligned}
&F(u,v) = \\
&\tfrac{1}{2}\left(\tfrac{4}{2}e^{-i2\pi\frac{u}{2}} + \tfrac{3}{2}e^{-i2\pi\frac{u}{2}} + \tfrac{2}{2}e^{-i2\pi\frac{u}{2}} + \tfrac{0}{2}e^{-i2\pi\frac{u}{2}}\right)\left(\tfrac{4}{2}e^{-i2\pi\frac{v}{2}} + \tfrac{3}{2}e^{-i2\pi\frac{v}{2}} + \tfrac{2}{2}e^{-i2\pi\frac{v}{2}} + \tfrac{0}{2}e^{-i2\pi\frac{v}{2}}\right) \\
&= \tfrac{1}{2}\left(4e^{-i\pi u} + 3e^{-i\pi u/2} + 2 + 0\right)\left(4e^{-i\pi v} + 3e^{-i\pi v/2} + 2 + 0\right) \\
&= \tfrac{1}{2}(4 + 3e^{-i\pi u/2} + 2e^{-i\pi v} + 3e^{-i\pi(u/2+v)} + 4e^{-i\pi v/2} + 2e^{-i\pi u} + 0 + 0)
\end{aligned}
$$

اگر بخواهیم این مقادیر را به صورت ماتریس نمایش دهیم، ماتریس تبدیل فوریه تصویر به شکل زیر خواهد بود:

$$
\boldsymbol{F} = \begin{bmatrix} \mathbf{9} & \mathbf{5} \\ \mathbf{-1} & \mathbf{3} \end{bmatrix}
$$

■ **سوال ۵**

الف) تابع findContoursیکی از توابع پرکاربرد و مهم در کتابخانه OpenCV برای پردازش تصویر و شناسایی مرزهای شکل‌ها است. این تابع برای تشخیص مرزهای هموار و نامنظم در تصاویر استفاده می‌شود. تابع findContoursتصویر ورودی را به عنوان ورودی می‌گیرد و مرزهای هموار یا نامنظم تصویر را به صورت نقاط مرزی در قالب لیستی از نقاط برمی‌گرداند.

برای استفاده از تابع findContours، ابتدا تصویر ورودی را به سیاه و سفید تبدیل می‌کنیم و سپس از تابع Canny برای یافتن لبه‌های تصویر استفاده می‌کنیم. سپس با استفاده از تابع findContours، مرزهای تشخیص داده شده در تصویر را به دست می‌آوریم و با استفاده از تابع drawContours، مرزها را در تصویر اولیه رسم می‌کنیم.

در کد زده شده، یک تصویر با سه مربع سیاه بارگذاری می‌شود و سپس از توابع Cannyو findContoursبرای شناسایی مرزهای شکل‌ها استفاده می‌شود. در انتها با استفاده از تابع drawContours، مرزهای تشخیص داده شده در تصویر رسم می‌شوند. سپس با استفاده از تابع imshow، تصویر نهایی را نمایش می‌دهیم.

در کد، تابع Canny برای پیش پردازش تصویر استفاده شده است.

استفاده از تابع Canny به عنوان پردازش اولیه در تشخیص مرزهای شکل‌ها به دلیل داشتن ویژگی‌های زیر می‌تواند مناسب باشد:

۱. **لبه‌های تشخیص داده شده با استفاده از تابع Canny، نسبت به روش‌های دیگر مانند(thresholding) ، دقیق‌تر و با کیفیت بالاتری هستند. به همین دلیل، استفاده از این روش در تشخیص مرزها می‌تواند دقت تشخیص را افزایش دهد.**

۲. **تابع Cannyقابلیت حذف نویز در تصویر را دارد و می‌تواند با حذف نویز، لبه‌های مورد نظر را بهتر تشخیص دهد. این قابلیت برای تصاویری که با شرایط نویزی مواجه هستند بسیار مفید است.**

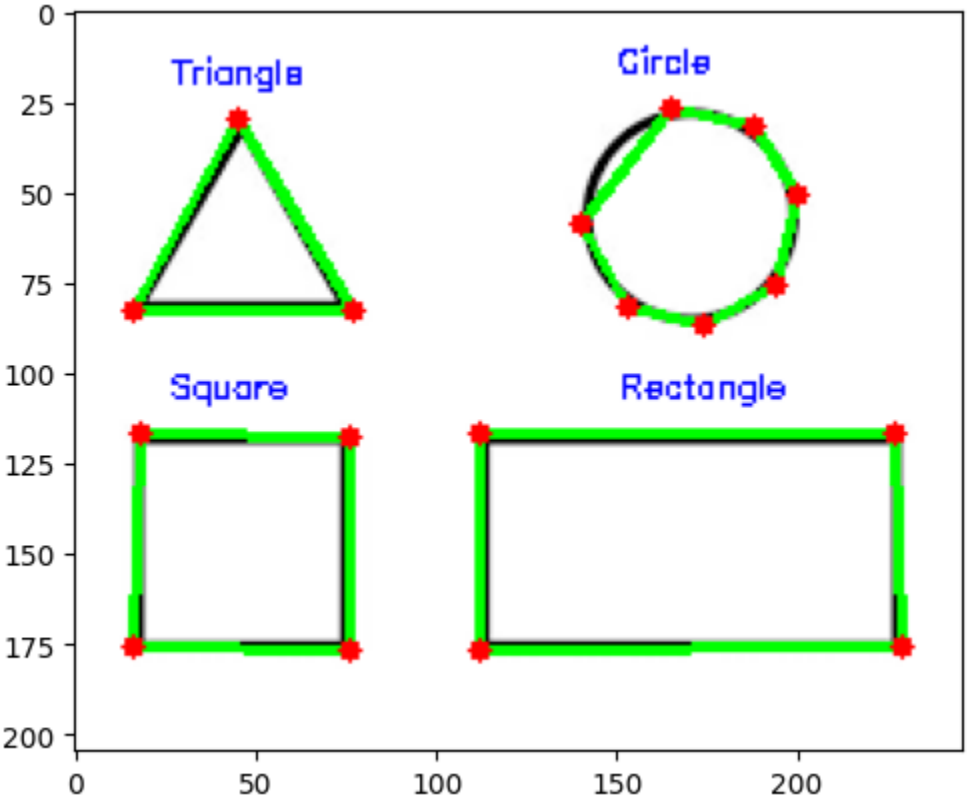
۳. **تابع Cannyقابلیت تشخیص لبه‌های ناهموار و پیچیده را دارد. این قابلیت در تشخیص مرزهای شکل‌های پیچیده مانند دایره‌های نامتماثل و شکل‌های ناهموار، بسیار مفید است.**

ب) این کد، تصویر ورودی را گرفته و شکل های مختلف روی آن را شناسایی کرده و با رنگ های مختلف و اطلاعات متنی آن را بر روی تصویر نشان می دهد.

در این کد، با استفاده از تابع cv2.findContours، لبه های تصویر به شکل اشکال مختلف استخراج می شود. سپس با استفاده از تابع cv2.approxPolyDP، این لبه ها به یک فرم ساده تر تقریب زده می شوند. در این مرحله، با توجه به تعداد اضلاع این شکل ها، شکل های مختلف شناسایی می شوند و نام هر شکل در کنار آن نشان داده می شود.

سپس با استفاده از تابع **cv2.drawContours**، لبه های شکل را با رنگ سبز روی تصویر رسم می کنیم. همچنین با استفاده از تابع **cv2.circle**، نقاط روی شکل با رنگ آبی رسم می شوند و با استفاده از تابع **cv2.putText**، نام هر شکل با رنگ قرمز در کنار آن نشان داده می شود.

در نهایت، با استفاده از **plt.imshow** تصویر نهایی را نمایش می دهیم.



■ سوال ۶

الف) هر چقدر سایز کرنل بیشتر باشد، اطلاعات بیشتری از تصویر از دست می رود و باعث می شود تصویر هموار تری داشته باشیم. در مواقعی که نویز تصویر زیاد است، بهتر است که سایز کرنل بیشتر باشد.

ب) فیلتر **Bilateral** یک فیلتر محافظت اطلاعات است که در پردازش تصویر مورد استفاده قرار می‌گیرد. این فیلتر برای کاهش نویز در تصاویر و حفظ حدّت‌ها و جزئیات آن‌ها مفید است.

فرمول این فیلتر برای هر پیکسل به شکل زیر است:

$$I_{filtered}(x) = \frac{1}{W_p} \sum_{y \in \Omega} I(y) \omega_p(||x - y||) \omega_s(||I(x) - I(y)||)$$

این تابع، فیلتر Bilateral را برای یک تصویر ورودی اعمال می‌کند.

ورودی‌های تابع شامل:

- **img: تصویر ورودی با ابعاد (H, W, C) که H و W ابعاد تصویر هستند و C تعداد کانال‌های رنگی تصویر.**
- **filter_size: اندازه فیلتر Bilateral که یک عدد صحیح است و نشان‌دهنده اندازه (fh, fw) فیلتر است.**
- **std: انحراف معیار هسته گاوسی در فضای محلی (فضای مکانی) که یک مقدار اعشاری است.**
- **rstd: انحراف معیار هسته گاوسی در فضای رنگی (فضای محدوده) که یک مقدار اعشاری است.**

تابع در ابتدا یک آرایه خالی به نام result به اندازه تصویر ورودی ایجاد می‌کند. سپس تصویر ورودی را با استفاده از تابع Reflect101 با ابعاد فیلتر منعکس می‌کند.

سپس با استفاده از دو حلقه تکرار، برای هر پچ ممکن در تصویر، محاسباتی انجام می‌دهد. در هر مرحله، یک پچ از تصویر با اندازه filter_size را انتخاب می‌کند و برای محاسبه فیلتر، مرکز این پچ را برای محاسبه انحراف معیار در فضای رنگی (range) مورد استفاده قرار می‌دهد. در واقع، فیلتر Bilateral ، با در نظر گرفتن شباهت رنگ و فاصله محلی، ترکیبی از یک فیلتر گاوسی در فضای رنگی و یک فیلتر گاوسی در فضای محلی است.

در این تابع، فیلتر گاوسی در فضای رنگی و فضای محلی با استفاده از دو انحراف معیار std و rstd محاسبه می‌شوند. سپس، این دو فیلتر با یکدیگر ضرب می‌شوند تا کرنل فیلتر نهایی تولید شود.