سوال یک

الف) در قسمت اول که تابعی که یک ماتریس ۱۰*۱۰ را رندوم انتخاب می کند را میزنیم و بعد تابع convolve را میزنیم به این صورت که ابتدا به اندازه سایز کرنل تقسیم بر دو reflect padding به ماتریس خود اضافه می کنیم و بعد در دو تا حلقه تودرتو هر دفعه اون قسمت از ماتریس که باید با کرنل conv بشود را در می اوریم و در نهایت ماتریس نهایی را return می کنیم.

و هم چنین می دانیم sobel افقی به صورت زیر است:

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

و sobel عمودی نیز به صورت زیر است:

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

اندازه و جهت گرادیان نیز به صورت زیر تعریف میشود

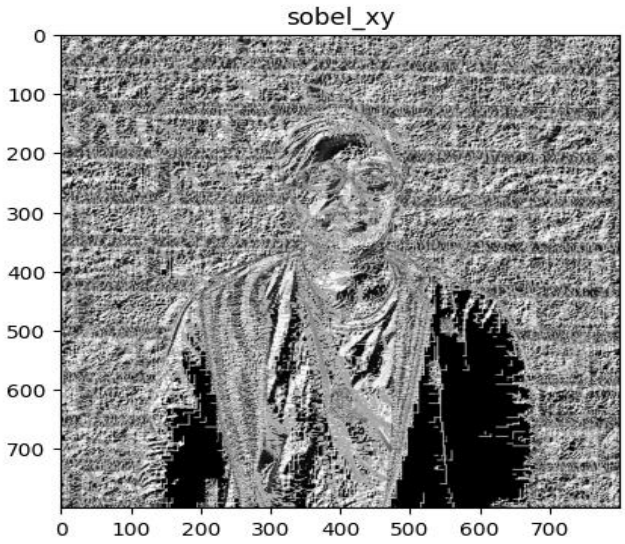
$$M(x,y) = \|\nabla f\| = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

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

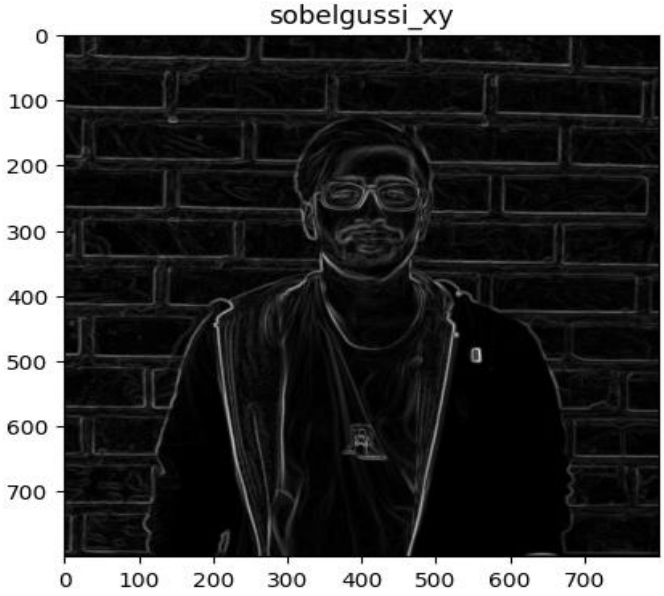
ب) برای این سوال فیلتر گوسی را خودمون پیاده کردیم که فرمول ان طبق زیر است:

$$G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}} = Ke^{-\frac{r^2}{2\sigma^2}}$$

و بدون فیلتر گوسی تصویر ما بهصورت زیر میشود:



و با فیلتر گوسی به صورت زیر میشود:



با اعمال فیلتر گوسی نویزهای کوچک و پراکنده در تصویر حذف میشوند با ترکیب این دو فیلتر باعث میشود لبه ها با دقت بیشتری شناسایی شوند و کیفیت تصویر نیز بیشتر باشد.

ج)cv2.sobel دارای پارامترهای زیر است:

۱.src: تصویر ورودی که می‌خواهید لبه‌های آن را تشخیص دهید.

۲.ddepth: عمق تصویر خروجی که می‌تواند ۸ بیتی، ۱۶ بیتی و ۳۲ بیتی باشد.

۳. dx,dy:مشخص میکند که sobel از کدام جهت باید اعمال شود.

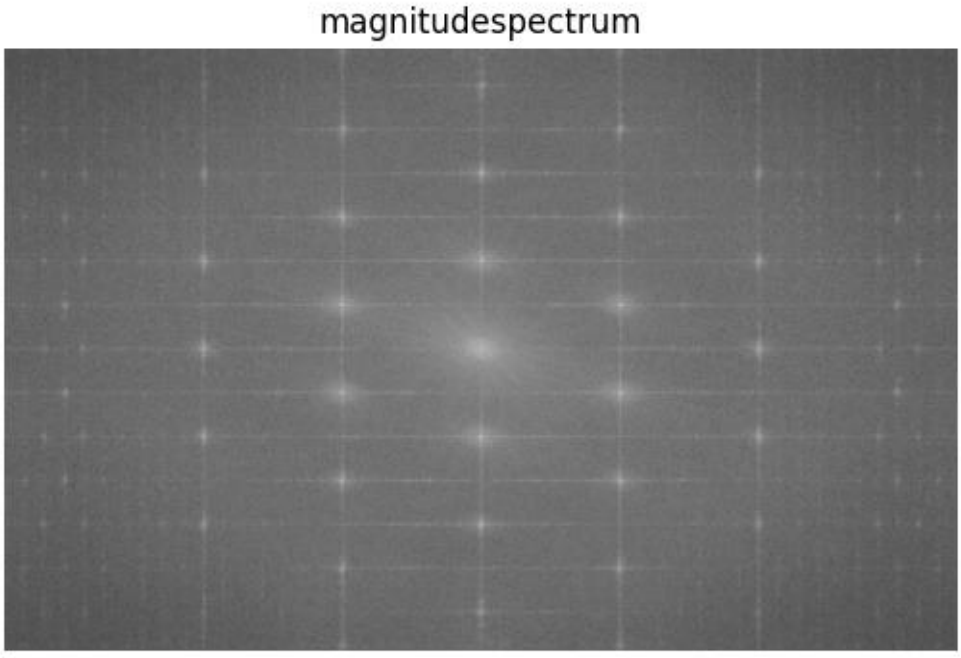
۴.ksize: اندازه فیلتر سوبل که باید عددی فرد باشد

۵.scale: مقیاس‌بندی خروجی فیلتر .Sobel مقدار پیش‌فرض ۱ است و می‌توان از مقادیر دیگر نیز استفاده کرد.

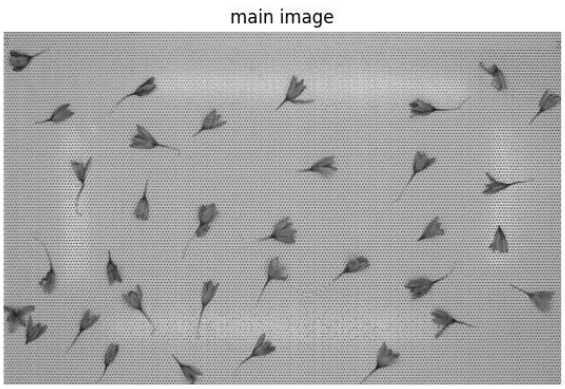
۶.delta:مقدار افست خروجی فیلتر که مقدار پیش فرض آن برابر با صفر است.

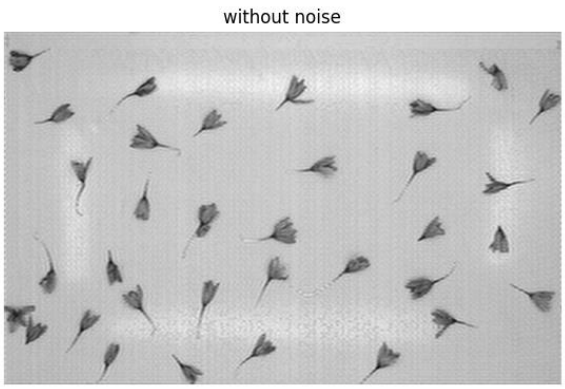
سوال دوم

الف)در ابتدا تصویر تبدیل فوریه را به دست میاوریم که به صورت زیر است:



من به نظرم نقاطی که نورانی تر هستند و نزدیک تر به مرکز هستند که مرکز نشان دهنده میانگین پیکسل ها هست نویز نیستند و نقاطی که دورتر از مرکز هستند را باید حذف کرد من یک فیلتر مستطیلی به نام mask استفاده کردم که مشخص کردم تا چه ابعادی برابر با یک باشد و بیرون از ان ابعاد برابر با ۰ باشد البته اینکه چه ابعادی دقیقا باید استفاده کرد باید با ازمون و خطا بهش رسید و بعد از اینکه این فیلتر را با تبدیل فوریه conv کردیم فوریه معکوس میزنیم و تصویر را به حوزه مکان می بریم نتیجه به صورت زیر است:

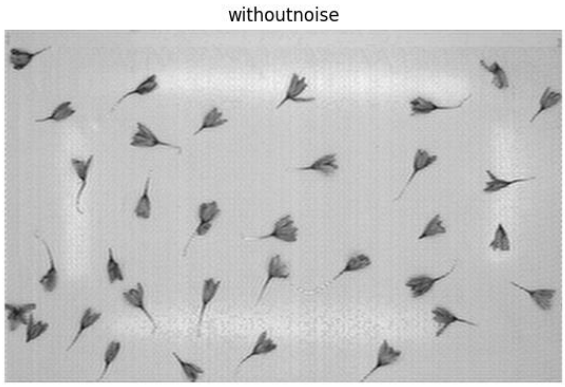


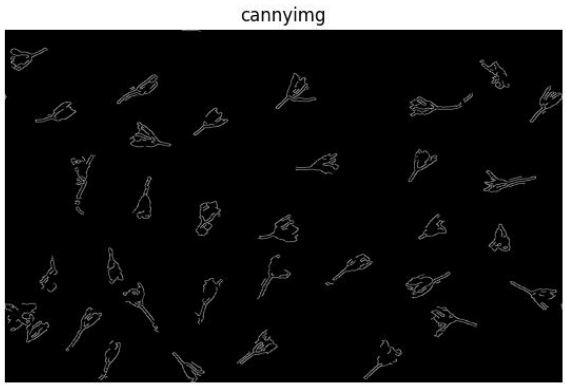


ب) باید threshold ها را جوری گذاشت تا جایی که ممکن هست لبه یابی گل ها دقیق تر بشود

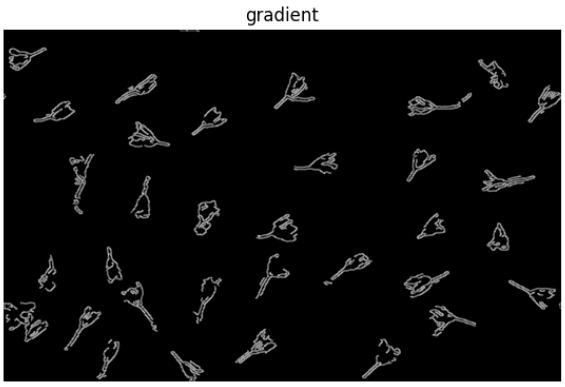
```
edge = cv2.Canny(img,60,150)
```

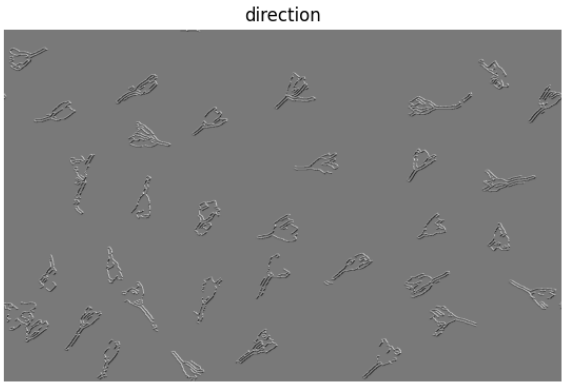
الان در این کد هر پیکسلی که گرادیان ان از ۶۰ کمتر باشه به عنوان لبه شناسایی نمیشه واگه بیشتر از ۱۵۰ باشه به عنوان لبه شناسایی میشه بین ۶۰ تا ۱۵۰ هم اگر مجاور پیکسل های پیوسته باشه شناسایی میشه.نتیجه canny به صورت زیر است:





ج)گرادیان و جهت گرادیان نیز به صورت زیر میشود





سوال سوم

الف)فیلتر های پایین گذر برای حذف نویز با فرکانس بالا و هم چنین حذف جزئیات تصویر مورد استفاده قرار می گیرند مثلا وقتی فیلتر smoothing میزنیم تا حدی نویز حذف میشه ولی جزئیات تصویر نیز کمی از بین می رود فیلتر پایین گذر یکی از ابزارهای مهم در پردازش تصاویر است که به ما این امکان را می‌دهد تا نویزها و جزئیات نامطلوب تصویر را حذف کنیم معمولا در پردازش تصاویر پزشکی، تصاویر ماهواره‌ای و تصاویر حرکتی استفاده میشود.

فیلتربالاگذر در پردازش تصویر برای حذف نویزهای با فرکانس پایین و همچنین تقویت تصویر مانند لبه ها و تفاوت های رنگی مورد استفاده قرار می گیرد به عنوان یک فیلتر تشدیدکننده (sharpening filter) برای تقویت لبه‌های تصویر و افزایش وضوح تصویر مورد استفاده قرار گیرد معمولا در تصاویر ماشینی، تصاویر ترکیبی و تصاویر با شدت نور کم استفاده میشود.

ب) از فیلتر بالاگذر استفاده شده است اگر دقت کنیم در تصویر سمت راست لبه ها تقویت شده اند پس از فیلتر بالاگذر استفاده شده است.

ج) نویز جمع شونده: در این نوع نویز، به تصویر یک سیگنال نویزی با یک شدت خاص اضافه می‌شود. به عنوان مثال، در تصاویر دیجیتال، نویز جمع شونده ممکن است به دلیل اضافه شدن سیگنال‌های نویزی از پردازش دیجیتال، نویزهای خطی از سیستم‌های دوربین و یا نویزهای محیطی مانند نویز موجود در تصاویر گرفته شده با شرایط نوری ضعیف وجود داشته باشد.

نویز ضرب شونده: در این نوع نویز، شدت سیگنال نویزی با شدت سیگنال اصلی ضرب می‌شود. به عنوان مثال، در تصاویر دیجیتال، نویز ضرب شونده ممکن است به دلیل نویز موجود در تصویر گرفته شده با شرایط نوری ضعیف،

نویزهای غیرخطی از سیستم‌های دوربین و یا نویزهای محیطی مانند نویز موجود در تصاویر گرفته شده با شرایط نوری ضعیف وجود داشته باشد.

برای حذف آن نیز می توان از راهکارهای زیر استفاده کرد

فیلترینگ: با استفاده از فیلترهای مختلفی مانند فیلتر گوسی، فیلتر میانگین و فیلتر میانه، می‌توان نویز جمع شونده و ضرب شونده را کاهش داد و تصویر را با کیفیت بهتری ارائه داد.

الگوریتم‌های پردازش تصویر: با استفاده از الگوریتم‌های مختلفی مانند الگوریتم‌های تشخیص لبه و الگوریتم‌های تخمین توزیع تصویر، می‌توان نویز را کاهش داد و تصویر را با کیفیت بهتری بازیابی کرد.

روش‌های تخمین بافت تصویر: با استفاده از روش‌های مختلفی مانند روش‌های کالمن و روش‌های ویولت، می‌توان بافت تصویر را تخمین زد و نویز را از بین برد.

شبکه‌های عصبی عمیق: با استفاده از شبکه‌های عصبی پیچشی یا CNN و شبکه‌های عصبی تمام متصل یاMLP ، می‌توان نویز را کاهش داد و تصویر را با کیفیت بهتری بازیابی کرد.

استفاده از فناوری‌های مختلف: فناوری‌هایی مانند فناوری‌های موجود در دوربین‌های دیجیتال و فناوری‌های موجود در نرم‌افزارهای پردازش تصویر می‌توانند برای کاهش نویز جمع شونده و ضرب شونده مورد استفاده قرار گیرند.

د) نویز نمک و فلفل یا نویز Salt and Pepper در تصاویر دیجیتال، نوعی نویز است که با ایجاد نقاط سفید و سیاه در تصویر، کیفیت تصویر را کاهش می‌دهد. این نوع نویز معمولاً به دلیل خرابی سنسور دوربین، ارتباط نامناسب با کانال انتقال و یا مشکلات پردازش تصویر ایجاد می‌شود.

یکی از روش‌های موثر برای حذف نویز نمک و فلفل، استفاده از فیلتر میانه(Median Filter) است. در این روش، با استفاده از یک پنجره مربعی با ابعاد مشخص، مقدار وسطی پیکسل‌های هر ناحیه از تصویر به عنوان مقدار پیکسل مرکزی آن ناحیه در نظر گرفته می‌شود. با این کار، نویز نمک و فلفل که به صورت نقاط سفید و سیاه در تصویر وجود دارد، از بین می‌رود و تصویر با کیفیت بهتری بازیابی می‌شود.

سوال چهارم

جواب این سوال در تصاویر زیر آمده است:

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v)\, e^{+j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

الف) تعداد سیناپس (0,0) تبدیل فوریه آن برابر با جمع تعداد تمام پیکسل ها / عبارت زیر حاصل می‌شود

$$F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{0}{M} + \frac{0}{N}\right)} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) =$$

بنابراین کافی است مقادیر پیکسل ها را جمع کنیم

ب)

x → / y ↓	0	1
0	4	0
1	3	2

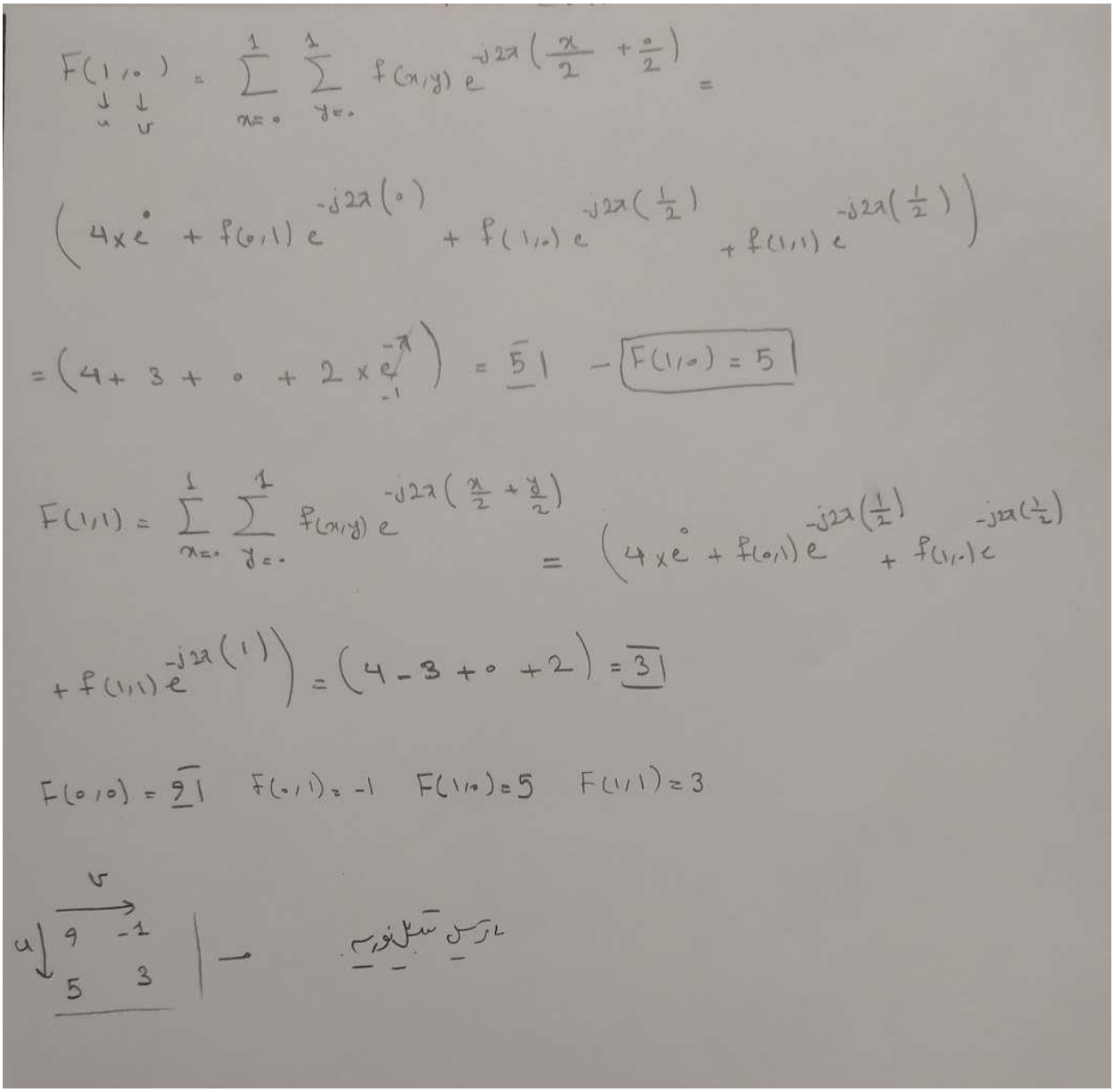
$$\rightarrow F(0,0) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y)\, e^{-j2\pi\left(\frac{0}{2} + \frac{0}{2}\right)} =$$

$$F(0,0) = (4+3+0+2) = 9 \rightarrow \boxed{F(0,0) = 9}$$

$$F(0,1) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y)\, e^{-j2\pi\left(\frac{0\times x}{2} + \frac{y}{2}\right)} = \left(4e^{0} + f(0,1)e^{-j2\pi\left(\frac{1}{2}\right)} + f(1,0)e^{0} + f(1,1)e^{-j2\pi\left(\frac{1}{2}\right)}\right)$$

$$= \left(4 + 3\times e^{-j\pi} + 0\times 1 + 2\times e^{-j\pi}\right) = \left(4 + 3\left(\cos(-\pi) + j\sin(-\pi)\right) + 2\left(\cos(-\pi) + j\sin(-\pi)\right)\right.$$

$$= \left(4 + 3(-1) + 2(-1)\right) = (4-5) = -1 \rightarrow \boxed{F(0,1) = -1}$$



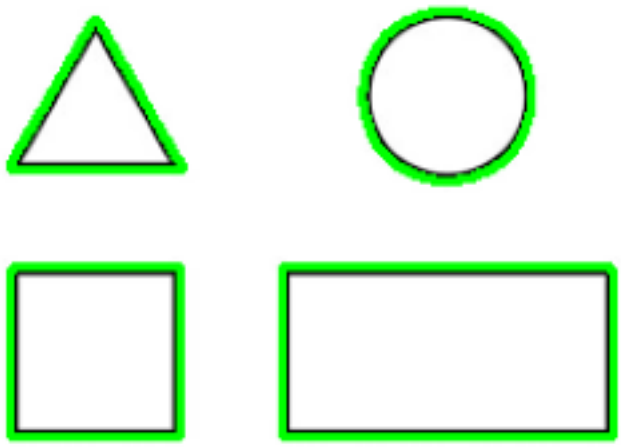
سوال پنج

الف)تصویر خوانده شده است.

ب) تابع cv2.findContoursیک تصویر سیاه و سفید را دریافت می‌کند و لبه‌های آن را شناسایی کرده، کانتورهای مختلف را در تصویر بازمی‌گرداند. به طور کلی، کانتورها به عنوان مجموعه‌ای از نقاط یا پیکسل‌ها در یک تصویر تعریف می‌شوند که در کنار هم قرار گرفته‌اند و معمولا برای شناسایی و تفکیک شی‌ها در تصویر استفاده می‌شوند.

برای پیدا کردن مرز یک شکل با استفاده از تابع cv2.findContours، اول باید تصویر را سیاه وسفید کنیم.با استفاده ازتابع cv2.Canny، لبه‌های تصویر را شناسایی می‌کنیم. سپس، با استفاده از تابع cv2.findContours، کانتورهای مختلف در تصویر را پیدا می‌کنیم. برای رسم مرز شکل، می‌توانیم از تابع cv2.drawContoursاستفاده کنیم. به طور خلاصه، برای پیدا کردن مرز یک شکل در تصویر، ابتدا تصویر را سیاه و سفید می کنیم و سپس با استفاده از تابع cv2.Cannyلبه‌های تصویر را شناسایی کرده و با تابع cv2.findContoursکانتورهای مختلف را پیدا کرده و با تابع cv2.drawContoursمرز شکل را رسم می‌کنیم. در این روش، تابع cv2.Cannyبرای پیدا کردن لبه‌های تصویر و تابع cv2.findContoursبرای پیدا کردن کانتورها استفاده می‌شود.دلیل استفاده از canny نیز به این علت است که تابع findContoursبرای پیدا کردن کانتورها در یک تصویر، باید با تصویری کار کند که لبه‌های آن به خوبی تشخیص داده شده باشند. بنابراین، اگر از تصویر اصلی به عنوان ورودی به findContoursاستفاده کنید، ممکن است که کانتورهایی که شامل لبه‌های ضعیف یا ناواضح هستند، شناسایی نشوند.

خروجی تصویر قسمت ب به صورت زیر است:



قسمت ج) تابع approxPolyDP برای تخمین یک کانتور با یک چندضلعی ساده کاربرد دارد. این تابع با استفاده از یک الگوریتم خاص، تعدادی نقطه را از یک کانتور استخراج می‌کند که می‌توانند به عنوان نقاط گوشه کانتور استفاده شوند.

توضیح کد:در ابتدا یک حلقه بر روی کل کانترها میزنیم و برای هر کانتر نقاط گوشه را پیدا می کنیم پارامتر های تابع approxPolyDP مطابق زیر است:

epsilon : این پارامتر مشخص می کند که تقریبی چه میزان از نواحی منحنی را در نظر بگیرد. معمولاً این پارامتر به صورت طولی مشخص می شود و برابر با یک عدد ضرب شده در طول نواحی منحنی است.

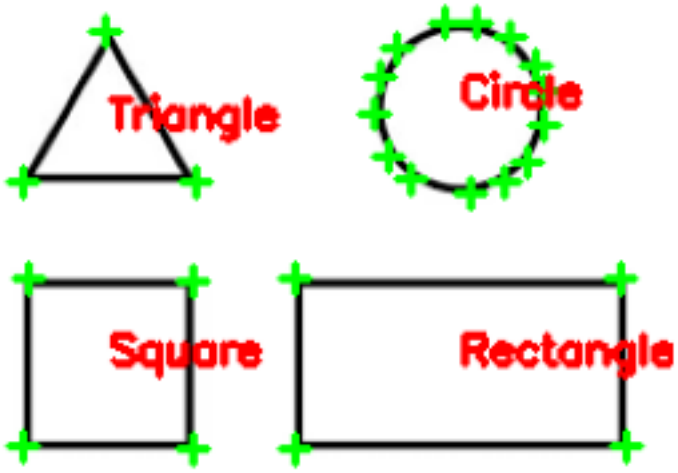
closed : این پارامتر مشخص می کند که خروجی چندضلعی تقریبی با خطوط کاملاً بسته باشد یا نه. اگر این پارامتر Trueباشد، خروجی چندضلعی به صورت بسته خواهد بود و اگر Falseباشد، خروجی چندضلعی به صورت باز خواهد بود. پیشفرض برای این پارامتر نیز Trueاست.

```
for contour in contours:
    # Approximate the contour to a polygon
    approx = cv2.approxPolyDP(contour, 0.01*cv2.arcLength(contour, True), True)
```

و در ادامه برای که نقاط گوشه را علامت بگذاریم یک for بر روی .approx میزنیم و x,y گوشه را به دست می اوریم و علامت می گذاریم.

```
for point in approx:
    x, y = point[0]
    # Draw a marker at the corner point
    cv2.drawMarker(image2, (x, y), (0, 255, 0), markerType=cv2.MARKER_CROSS, markerSize
```

و بعد بر اساس .length approx مشخص می کنیم متعلق به چه کلاسی هست.فقط با استفاده از cv2.boundingRect تشخیص می دهیم که شکل ما مستطیل است یا مربع یعنی عرض و ارتفاع را به دست می اوریم اگر برابر بودند مربع است.و در نهایت نوع کلاس را داخل شکل می نویسیم که شکل مطابق زیر است



د) می‌توانیم از ویژگی‌های مختلفی استفاده کنیم که مرتبط با خصوصیات هندسی شکل‌ها هستند. برخی از ویژگی‌هایی که می‌توان برای تشخیص اشکال هندسی استفاده کرد عبارتند از:

تعداد گوشه‌ها: تعداد گوشه‌های هر شکل می‌تواند به عنوان یک ویژگی مفید در تشخیص نوع شکل مورد استفاده قرار گیرد.

نسبت اضلاع: در برخی از اشکال هندسی مانند مربع، اضلاع با طول یکسان است. در برخی دیگر اشکال، نسبت اضلاع ثابت است. این ویژگی نیز می‌تواند به عنوان یک ویژگی مفید در تشخیص شکل مورد استفاده قرار گیرد.

مساحت: مساحت شکل نیز می‌تواند به عنوان یک ویژگی مفید در تشخیص شکل استفاده شود.

قطر: در برخی از اشکال هندسی مانند دایره، قطرها به طور یکسان هستند. این ویژگی نیز می‌تواند در تشخیص شکل مورد استفاده قرار گیرد.

محیط: محیط شکل نیز می‌تواند به عنوان یک ویژگی مفید در تشخیص شکل استفاده شود.

زوایای داخلی: تعداد و اندازه زوایای داخلی شکل نیز می‌تواند به عنوان یک ویژگی مفید در تشخیص شکل مورد استفاده قرار گیرد.

تقارن: در برخی از اشکال هندسی، تقارن بر روی اندازه و شکل اجزای شکل وجود دارد که می‌تواند به عنوان یک ویژگی مفید در تشخیص شکل استفاده شود.

سوال شش

الف) در این قسمت سعی داریم که با سه فیلتر متوسط گیر ، میانه و گوسی آشنا شویم. ابتدا برای padding باید تابع reflect101 را کامل نمایید و برای این کار مجاز به استفاده از حلقه ها for(، whileو...(نیستید . همچنین تاثیر سایز کرنل در خروجی را نیز تحلیل کنید .)در تمامی این بخش مجاز به استفاده از کتابخانه های تخصصی پردازش تصویر نیستید

در ابتدا تابع refelect را پیاده سازی کردم در ابتدا یک ارایه نامپای با سایز خود تصویر ولی به علاوه سایز فیلتر می گیرم و بعد تصویر را در وسط این ارایه نامپای می گیرم و بعد و بعد برای سمت های بالا و پایین و چپ وراست و سمت گوشه ها یا همان corner ها مقدار را به دست می اورم هر چند با np.pad نیز میشد این سوال را زد.

برای فیلتر متوسط گیر دو حلقه تودرتو بر روی سایزهای تصویر داریم و بعد پنجره ای که میانگین آن را به میخواهیم به دست می اوریم و میانگینش را برابر با پیکسل وسط میگذاریم

برای فیلتر میانه دقیقا مشابه بالا فقط میانه را برابر با پیکسل وسط میگذاریم.

برای فیلتر گوسی باید مقدار های کرنل را محاسبه کنیم مشابه بالا دو حلقه تو در تو داریم و برای هر خانه کرنل تفاوت x,y ان را با خانه وسط به دست می اوریم و مقدار ان خانه طبق فرمول زیر به دست می اید

$$G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}} = Ke^{-\frac{r^2}{2\sigma^2}}$$

که k برای نرمال سازی هست که جمع همه خانه های کرنل برابر با یک شود

در هر سه فیلتر ما اگر سایز کرنل را کم بگیریم نویز های کمتری از بین رفته اند ولی جزئیات بیشتری از تصویر مشخص است و اگر سایز کرنل خیلی بزرگ بگیریم نویز های بشتری از بین میروند ولی جزئیات کمتری از تصویر مشخص است و تصویر مات تر و تارتر نسبت به کرنل کوچک تر است در واقع یک trade off بین جزنیات تصویر و از بین بردن نویز است.

ب(در این قسمت میخواهیم با فیلتر Bilateral آشنا شویم . ابتدا درباره ی آن در اینترنت جستجو کرده و فرمول آن، پارامترهای آن و چرایی استفاده از آن را در گزارش خود شرح دهید . همچنین درباره ی تاثیر مقادیر کم و زیاد دو انحراف معیار توضیح دهید.)در تمامی این بخش مجاز به استفاده از کتابخانه های تخصصی پردازش تصویر نیستید.)

مشکل فیلتر گوسی این بود که در کرنل ان خانه هایی که فاصله یکسانی با مرکز داشتند مقدار یکسانی می گرفتند بدون توجه به این نکته که آن خانه ها میتوانند تفاوت رنگی زیادی با مرکز و حتی خودشون داشته باشند و در این صورت اگر فیلتر گوسی میزدیم باعث میشد لبه هایی که جهش رنگی داریم و این لبه ها در شناسایی اجسام بسیار مهم هستند از بین بروند و تصویر مات شود در bilateral filter این نکته تفاوت رنگی را در نظر می گیریم فرول آن به صورت زیر است:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \epsilon S} G_{\sigma_s}(||p-q||) G_{\sigma_r}(|I_p - I_q|) I_q$$

Normalization Factor

Space Weight

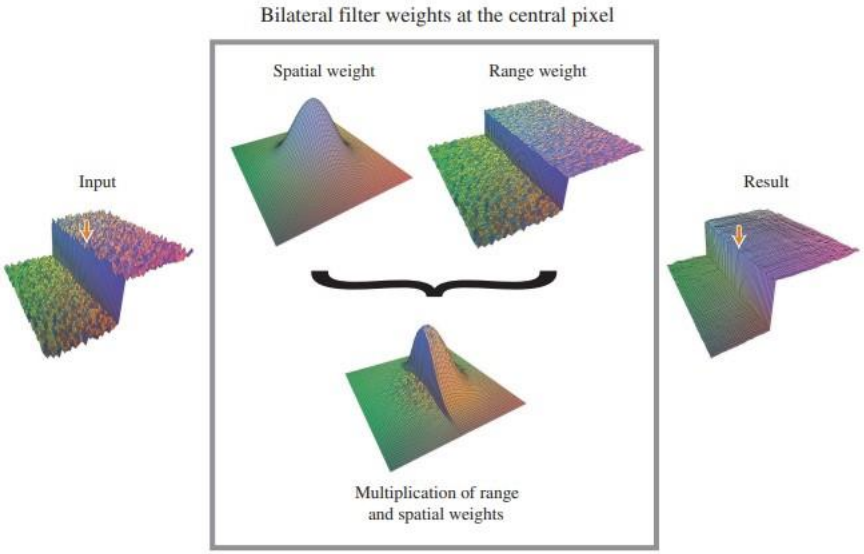
Range Weight

که در ابتدا قسمت space weight ان را توضیح میدهم منظور از ||p-q|| فاصله ی این دو پیکسل با هم هست و فرمول G به صورت زیر است:

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

که x همان فاصله دو پیکسل p,q است و phi برابر با همان phis هست که در کد با std مشخص کردیم

در قسمت range weight منظور از |Ip-Iq| تفاوت رنگی دو پیکسل p,q است همان طور که میبینید تفاوت رنگی بین دو پیکسل نیز در این فرمول در نظر گرفته شده است.Wp نیز برای نرمال سازی هست که جمع همه خانه های کرنل برابر با ۱ شود.در ادامه نیز یک شکل اوردم که به فهم بیشتر این فرمول کمک می کند



هر چقدر پارامتر std که مربوط به فاصله پیکسل تا پیکسل وسط در کرنل هست مقدار بیشتری باشه تصویر تارتر هست و لبه های تصویر به خوبی مشخص نیست و هر چقدر کم تر باشه جزئیات بیشتر هست ولی نویز بیشتر هست

هر چه پارامتر rstd که مربوط به فاصله رنگی پیکسل تا پیکسل وسط در کرنل هست هر چه مقدار بیشتر باشه لبه ها و انجاهایی که جهش رنگی داریم بیشتر مشخص است و هر چقدر کمتر باشد لبه های تصویر کمتر مشخص هست و نویز هم بیشتر است

در مورد خود پیاده سازی تابع نیز مراحل زیر را طی کردم:

در ابتدا یک تابع برای فصله دو پیکسل و یک تابع برای تابع گوسی مطابق زیر تعریف کردم

```
def gaussian(x,sigma):
    return (1.0/(2*np.pi*(sigma**2)))*np.exp(-(x**2)/(2*(sigma**2)))

def distance(x1,y1,x2,y2):
    return np.sqrt(np.abs((x1-x2)**2+(y1-y2)**2))
```

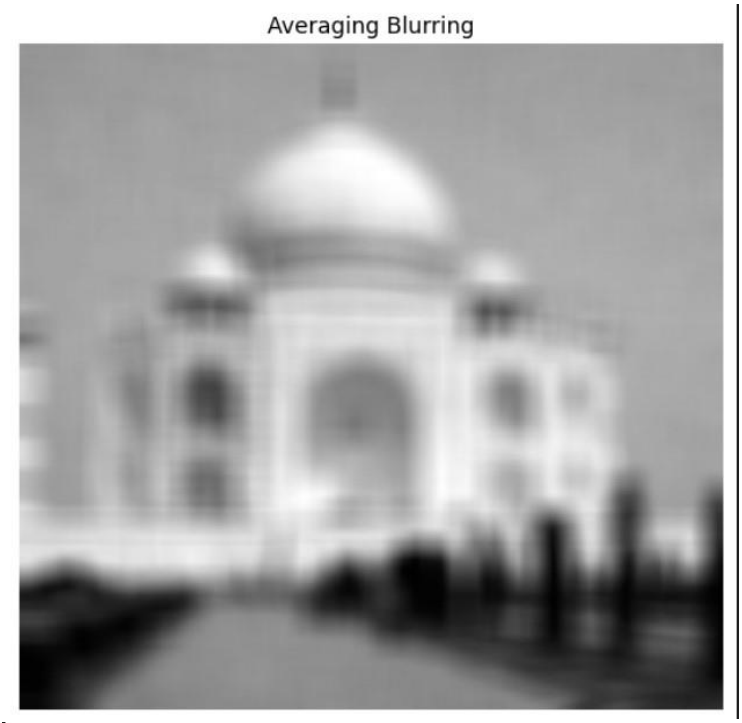
در ادامه یک ارایه نامپای به نام mask تعریف کردم و نقاطی در کرنل که فاصله آن از center بیشتر هست index ان ها را برای موقعی که لوپ میزنم در نظر نمی گیریم(این قسمت برای کمتر شدن خطا و محاسبات انجام دادم)

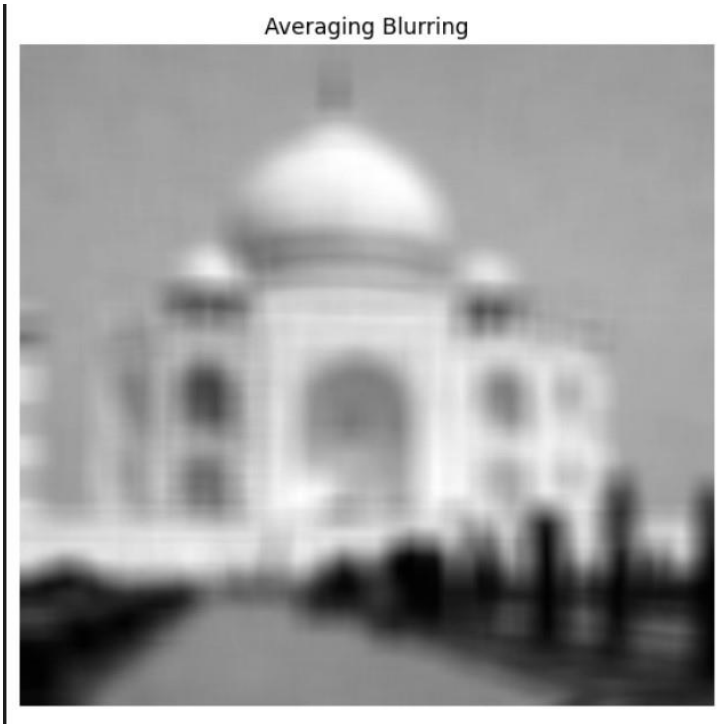
```
# Create a circular mask with the given diameter
mask = np.ones((filter_size, filter_size))
center = filter_size // 2
for i in range(filter_size):
    for j in range(filter_size):
        if distance(i, j, center, center) > center:
            mask[i, j] = 0

# Get the indices of the non-zero elements in the mask
mask_indices = np.where(mask)
```

و در ادامه سه حلقه تودرتو داریم و کرنل مناسب برای هر پیکسل را طبق فرمولی که گفته شد به دست می اوریم.

ج(در این قسمت هدف، نوشتن تمام فیلترهای قسمت های الف و ب با کتابخانه ی OpenCV و مقایسه ی آن با پیاده سازی خودمان است. (راهنمایی: اگر کدهای شما درست باشد ، خروجی ها باید کامال یکسان باشدخروجی ها یکسان است و به صورت زیر است





عکس بالاتر مربوط به پیاده سازی خودم هست و عکس پایین تر مربوط به پیاده سازی opencv هست در بقیه موارد نیز به همین صورت است.

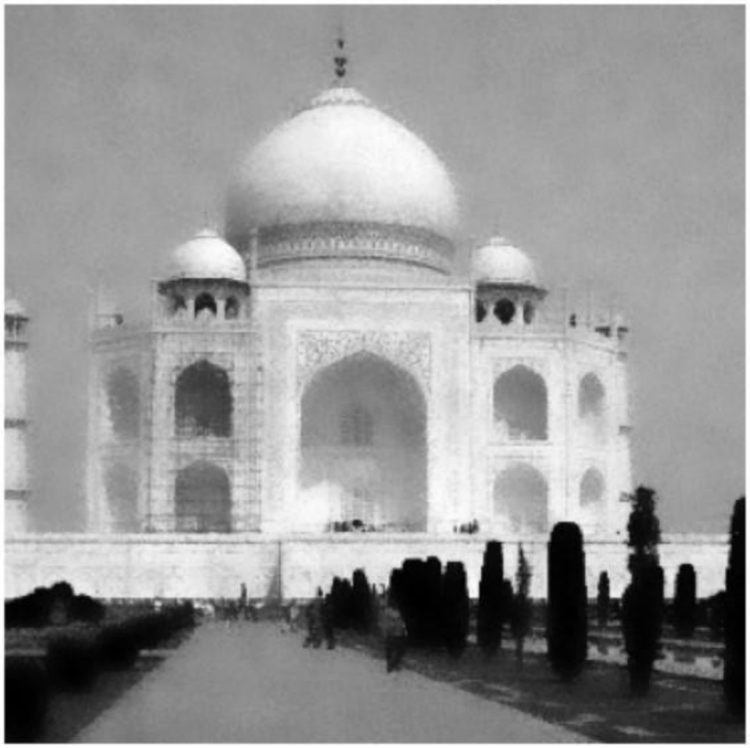
Median Blurring

Median Blurring

Gaussian Blurring

Gaussian Blurring

Bilateral blurring



Bilateral blurring