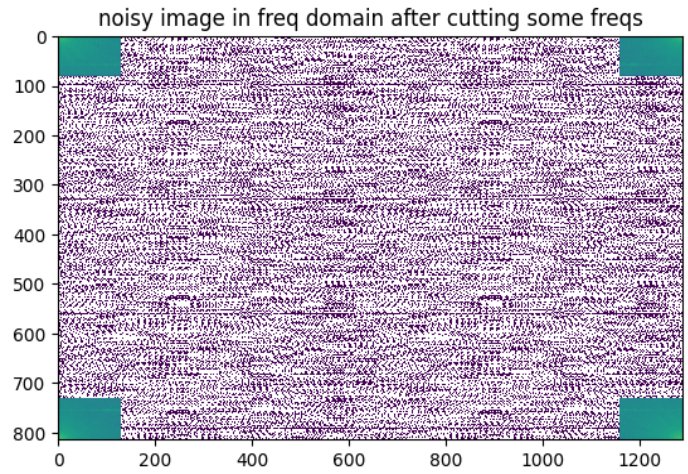
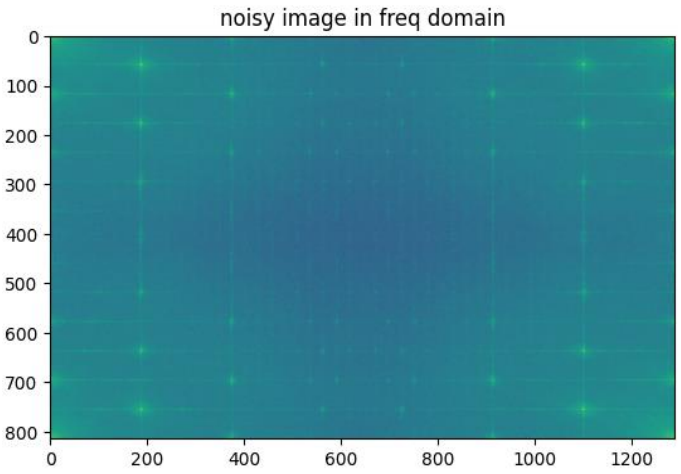
۱.الف

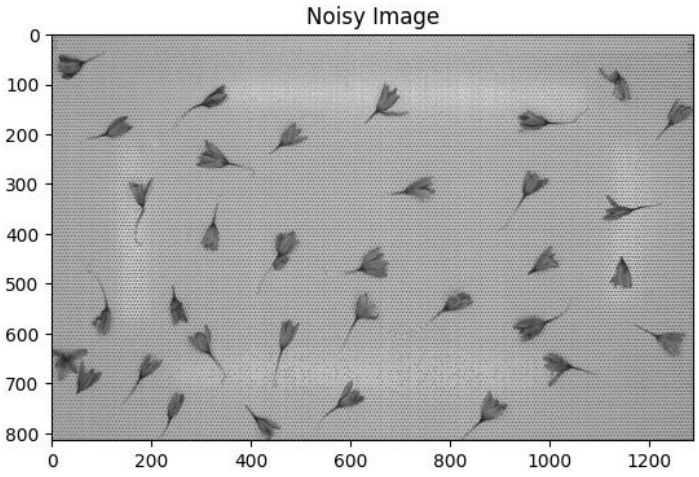


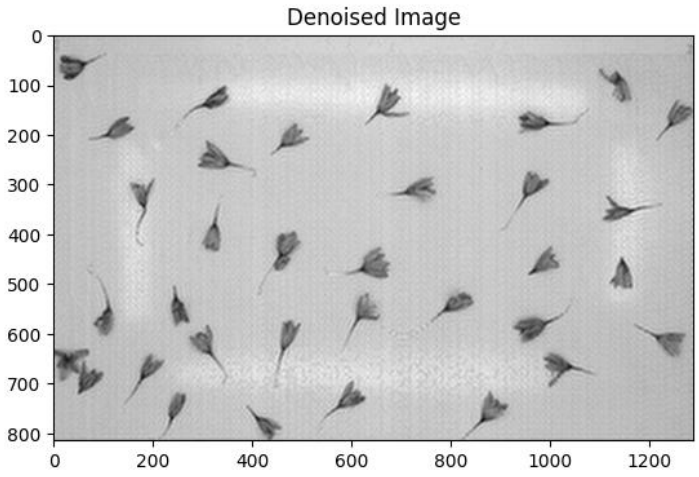



```
# Convert to freq domain
denoised = np.fft.fft2(denoised)
plt.imshow(np.abs(denoised), norm=LogNorm(vmin=5))
plt.title("noisy image in freq domain")
plt.show()

# Remove 80% of middle freqs and keep 10% of beginning and 10% of endin
keep_ratio = 0.1
denoised[int(height * keep_ratio):int(height * (1 - keep_ratio))] = 0
denoised[:, int(width * keep_ratio):int(width * (1 - keep_ratio))] = 0
plt.imshow(np.abs(denoised), norm=LogNorm(vmin=5))
plt.title("noisy image in freq domain after cutting some freqs")
plt.show()
```

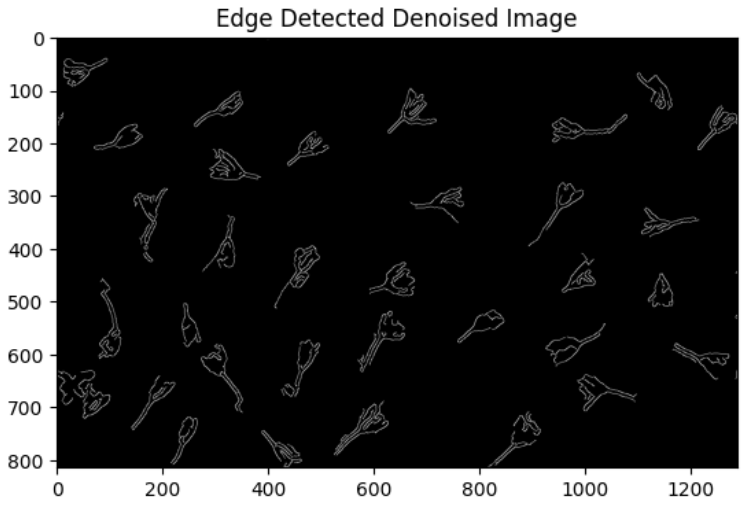
ابتدا به کمک ماژول آماده *numpy.fft* و تابع *fft2* (تبدیل فوریه دوبعدی) تصویر ورودی را به فضای فرکانسی بردم. برای نمایش مقدارها وقتی *plt.imshow* را بدون آرگومان اضافی اجرا کردم صفحه تمام بنفش در خروجی دیدم که برای متمایزشدن نقاط خروجی تصویر را بعد از نرمالایزکردن در اسکیل لگاریتمی با *LogNorm* روی خروجی بردم که نتیجه تصویر بالا سمت راست شد و تمایز فرکانس‌های مختلف کاملا مشخص است. در قسمت بعدی کد، ۸۰٪ فرکانس‌های بالا را حذف کردم. از آن‌جایی که ماهیت فرکانس‌ها به خاطر یکی بودن ۰ و *2π* حالت چرخشی (*circular*) دارد پس ۱۰٪ از فرکانس‌های نزدیک صفر و از ۱۰٪ فرکانس‌های انتهایی (که در واقع بسیار نزدیک به صفر هستند) را نگه داشتم و بقیه را حذف کردم. این کار را هم در راستای *x* و هم در راستای *y* انجام دادم تا قسمت‌های با فرکانس بالا که نمایانگر تغییرات شدید ناگهانی هستند (و احتمالا نویزها را شامل می‌شوند) حذف شوند و در نهایت نویزها از بین بروند. بطور مشابه، با اسکیل لگاریتمی شدت روشنایی فرکانس‌های مختلف را دوباره نمایش دادم که تصویر بالا سمت چپ نشان‌دهنده‌ی قسمت‌های حذف‌شده و گوشه‌های تصویر فرکانس‌هایی هستند که نگه داشتم.





بعد از برگرداندن تصویر از فضای فرکانسی به فضای مکان، خروجی به صورت سمت راست است. هرچه درصد فرکانس‌هایی که نگه داشتیم را کمتر کنیم نویز پشت زعفران‌ها کمتر می‌شود اما جزئیات تصویر هم بیشتر کم می‌شود. تصویری که در بالا به آن رسیدم با حذف ۸۰٪ فرکانس‌های میانی و نگه داشتن ۱۰٪ بالا و پایین است.

```
img_edge = cv2.Canny(np.uint8(denoised_image), 1, 110)
plt.imshow(img_edge, cmap="gray")
plt.title("Edge Detected Denoised Image")
plt.show()
```

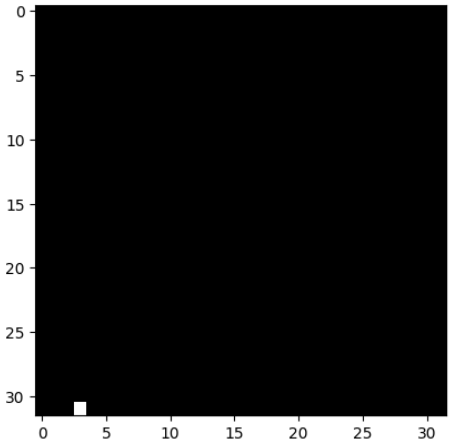


۱.ب. پارامتر اول تابع، تصویر ورودی است که می‌خواهیم canny روی آن اجرا شود. وقتی تصویر خروجی قسمت الف را به تابع دادم ارور خوردم و برای رفع مشکل ابتدا رنگ‌های تصویر را به فرمت uint8 تبدیل کردم بعد به تابع پاس دادم. پارامتر دوم و سوم T1 و T2 حد آستانه‌های مرحله آستانه‌گذاری دوسطحی در الگوریتم Canny داده می‌شوند. به این صورت که تمام پیکسل‌هایی که اندازه گرادیان آن‌ها از T1 بیشتر باشد لبه نیستند و حذف می‌شوند، تمام پیکسل‌هایی که اندازه گرادیان آن‌ها از T2 بیشتر باشد (و در مرحله حذف مقادیر غیربیشینه حذف نشده باشند) لبه تشخیص داده می‌شوند و تمام پیکسل‌هایی که اندازه گرادیان بین T1 و T2 داشته باشند جزو پیکسل‌های مشکوک به لبه هستند و تنها در صورتی جزو لبه حساب می‌شوند که حداقل با یکی از پیکسل‌هایی که جزو لبه هستند، مجاور باشند. در اینجا مقادیری که به تابع پاس دادم T1=1 و T2=110 بود که با آزمون و خطا و بررسی لبه‌های خروجی به این اعداد رسیدم. یعنی هر پیکسلی که مقدار اندازه گرادیان صفر باشد لبه نیست و اگر بالاتر از ۱۱۰ باشد حتما لبه است و اگر پیکسلی بعد از مرحله حذف مقادیر غیربیشینه اندازه گرادیان بین ۱ تا ۱۱۰ داشته باشد و حداقل با یکی از نقاط دیگر لبه مجاور باشد هم لبه تشخیص داده می‌شود.

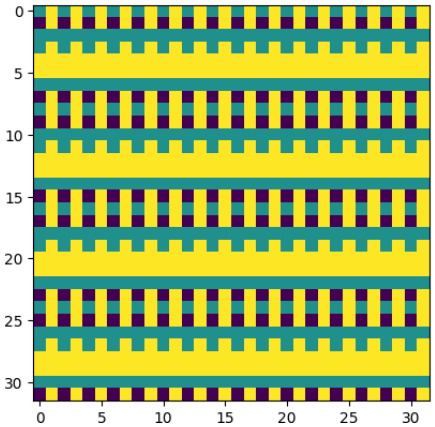
```
dy, dx = np.gradient(img_edge.astype(np.float64))
degrees = np.degrees(np.arctan2(dy, dx))
```

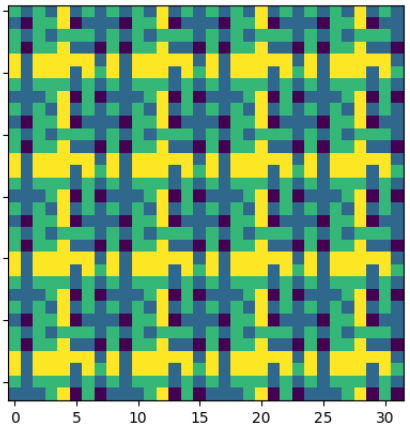
۱.ج. ابتدا نوع داده تصویر را به np.float64 تغییر دادم و سپس به کمک تابع آماده gradient خود numpy گرادیان تصویر را محاسبه کردم. در مرحله بعدی با تابع arctan2 جهت گرادیان را به دست آوردم و در نهایت به واحد درجه تبدیل کردم.

۲. تصویر داده شده را به کمک آرایه‌ی numpy ساختم تا بتوانم تبدیل فوریه دوبعدی بگیرم و مولفه‌های فرکانسی را بررسی کنم.



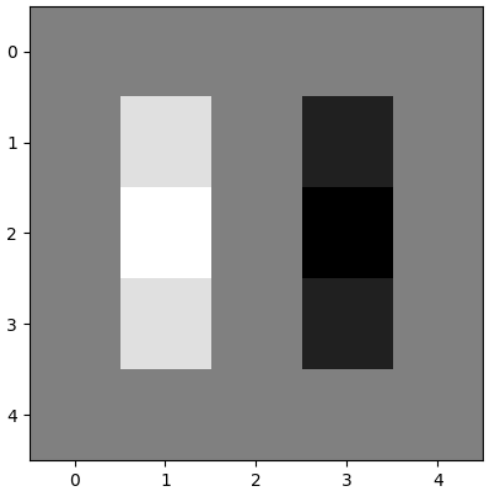
```
img = np.zeros((32, 32))
img[31][3] = 255
plt.imshow(img, cmap="gray")
plt.show()
```

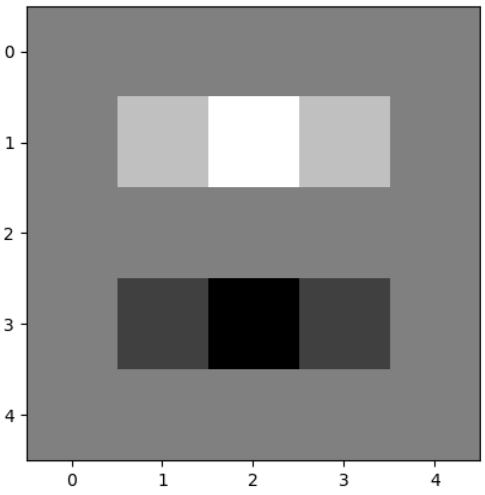




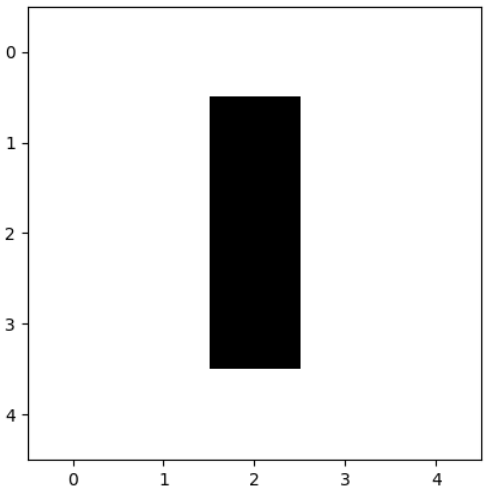
طیف فرکانسی سمت راست مربوط به تبدیل فوریه تصویر اولیه و طیف سمت چپ مربوط به تصویر بعد از انتقال دادن پیکسل به سمت راست است. همانطور که مشاهده می‌شود بعد از جابجا کردن فقط یک پیکسل طیف فرکانسی بسیار تغییر می‌کند و یک تغییر کوچک در حوزه مکان تاثیر بسیار زیادی در حوزه فرکانس می‌گذارد. هر دو طیف رسم شده متناوب هستند و یک الگو چندین بار در طیف فرکانس‌های مختلف بصورت متناوب تکرار شده است.

۳.

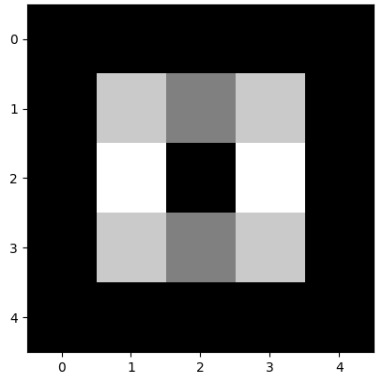




```
img = np.ones((5, 5))
for i in range(1, 4):
    img[i][2] = 0
plt.imshow(img, cmap="gray")
plt.show()
```



مطابق خواسته سوال یک تصویر ۵در۵ با لبه‌ی عمودی ایجاد کردم و فیلترهای *Sobel* را یکبار برای یافتن لبه‌های عمودی و یکبار برای یافتن لبه‌های افقی اجرا کردم.



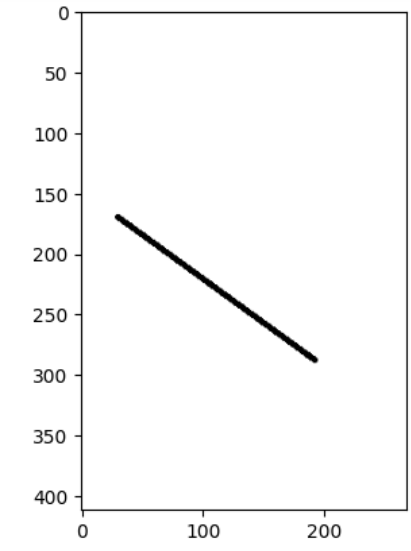
در نهایت اندازه‌ی هر نقطه را با استفاده از $\sqrt{{g_x}^2 + {g_y}^2}$ بدست آوردم. که در روبرو مشاهده می‌کنید.

در دو تصویر بالا (g_x و g_y) صفر رنگ خاکستری است و مقادیر مثبت و منفی که لبه‌ها را نمایش می‌دهند سفید یا سیاه هستند. در تصویر روبرو چون همه مقادیر مثبت هستند نمایش تصویر به این صورت است که پیکسل‌های با رنگ صفر سیاه نمایش داده شده‌اند و پیکسل‌های خاکستری و سفید مقادیر مثبت هستند. در تابع *filter_2d* که از تمرین قبلی برداشتم شرط گذاشتم که به ازای نقاط مرزی تصویر، لبه تشخیص داده نشود و صفر گذاشته شود. به بیان دیگر بجای *zero_padding* از *mirroring* استفاده کردم تا نقاط مرزی تصویر به اشتباه لبه‌های تصویر شناسایی نشوند.

۴.

y = -0.0007298491717875443x + 134.36391604323816

از رابطه داده شده در اسلاید۲۲ فایل FCV_08 استفاده کردم و معادله بالا را برای خط داده شده محاسبه کردم.



y = -0.0007298491717875443x + 134.36391604323816

$$m, x = \arg\min \sum_i (mx_i + c - y_i)^2$$

$$m = \frac{\bar{x}\bar{y} - \overline{xy}}{\bar{x}^2 - \overline{x^2}}$$

$$c = \bar{y} - m\bar{x}$$

```
img = cv2.imread("images/img_02.jpg")

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

_, binary_bw = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
plt.imshow(binary_bw, cmap="gray")
plt.show()

x, y = np.where(binary_bw)

N = len(x) # equal to len(y)
x_bar = sum(x) / N
y_bar = sum(y) / N
xy_bar = sum(x[i]*y[i] for i in range(N)) / N
x2_bar = sum(x[i]**2 for i in range(N)) / N

m = (x_bar*y_bar - xy_bar) / (x_bar**2 - x2_bar)
c = y_bar - m*x_bar

print(f"y = {m}x + {c}")
```

ابتدا تصویر را سیاه و سفید کردم و سپس با استفاده از تابع *threshold* تمامی پیکسل‌هایی که از ۱۲۷ (نصف رنگ سفید مطلق) بیشتر بودند را سفید مطلق و بقیه پیکسل‌ها را سیاه مطلق کردم تا سیاه‌وسفیدی که داریم در حالت دوگانه باشد و همه نقاط با رنگ صفر نقاط خط باشند. سپس با کمک تابع *np.where* مختصات نقاط خط را پیدا کردم و در نهایت به محاسبه معادله داده شده پرداختم و m و c را به دست آوردم.

منابع:

سوال ۱.الف: Image denoising by FFT — Scipy lecture notes (scipy-lectures.org)

matplotlib.colors.LogNorm — Matplotlib 3.6.0 documentation

numpy.fft.fft2 — NumPy v1.23 Manual

numpy.fft.ifft2 — NumPy v1.23 Manual

سوال ۱.ب: Python | cv2 Canny() Method - Java2Blog

Depth error in 2D image with OpenCV Python - Stack Overflow

سوال ۱.ج: numpy - Gradient orientation with arctan2 results in flipped angles - Stack Overflow

سوال ۴: Python OpenCV: Converting an image to black and white - techtutorialsx

numpy.where() – Explained with examples – thisPointer

python - Numpy where() on a 2D matrix - Stack Overflow