**Computer Vision**
**Dr. Mohammadi**
**Fall 2022**
**Hoorieh Sabzevari - 98412004**
**HW5**

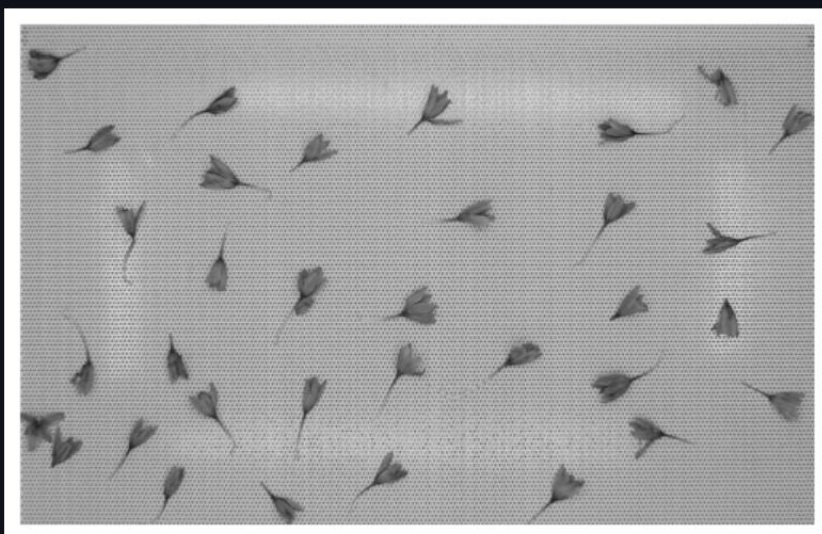
۱.

الف) ابتدا تصویر را می‌خوانیم و نمایش می‌دهیم.

```
path = "images/img_01.jpg"
img = cv2.imread(path, 0)
plt.imshow(img, cmap="gray")
plt.axis('off')
```

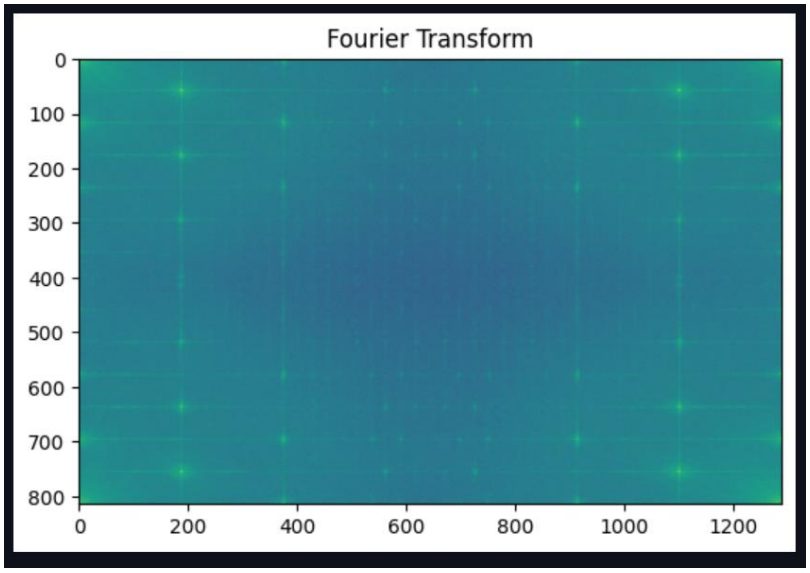
✓ 0.3s

```
(-0.5, 1289.5, 814.5, -0.5)
```



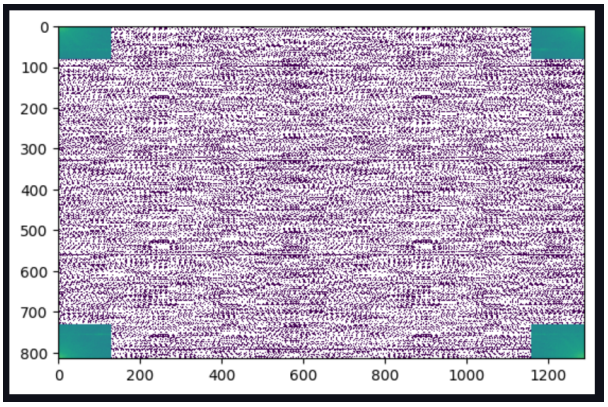
سپس با دستور fft تبدیل فوریه‌ی تصویر را می‌گیریم و abs آن را نمایش می‌دهیم. همچنین برای افزایش کنتراست از تابع log استفاده می‌کنیم. (منبع)

```
# https://scipy-lectures.org/intro/scipy/auto_examples/solutions/plot_fft_image_denoise.html
im_fft = fftpack.fft2(img)
plt.figure()
plt.imshow(np.abs(im_fft), norm=LogNorm(vmin=5))
plt.title('Fourier Transform')
```




همانطور که می‌بینیم نقاط پیک در تبدیل فوریه نشان‌دهنده‌ی نویز متناوب تصویر است. برای رفع آن‌ها به اندازه‌ی دو مستطیل از تبدیل فوریه را سیاه می‌کنیم. گوشه‌های تصویر را نباید از بین ببریم، چون میانگین تصویر است و اطلاعات مهم تصویر را دارا هستند.

```
k = 0.1
im_fft2 = im_fft.copy()
x, y = im_fft2.shape
im_fft2[int(x*k):int(x*(1-k))] = 0
im_fft2[:, int(y*k):int(y*(1-k))] = 0
plt.figure()
plt.imshow(np.abs(im_fft2), norm=LogNorm(vmin=5))
```
✓ 1.4s

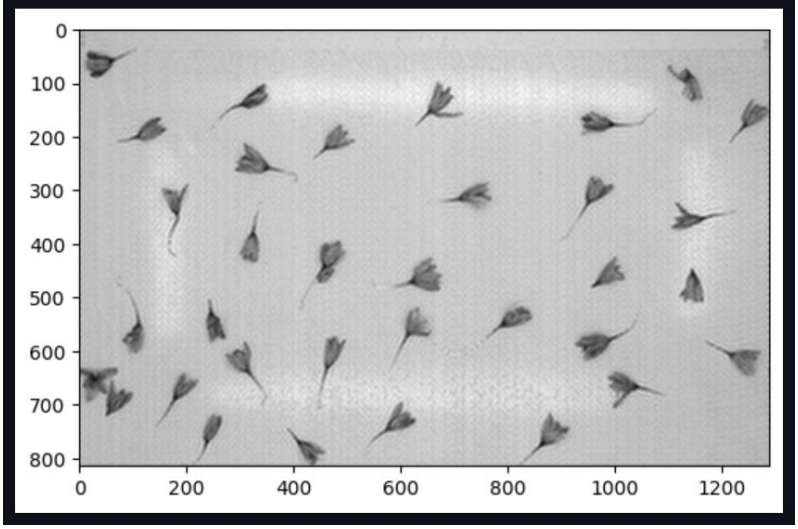


سپس از تصویر تبدیل فوریه‌ی معکوس گرفته و نمایش می‌دهیم.

```
im_new = fftpack.ifft2(im_fft2).real
cv2.imwrite("images/img_03.jpg", im_new)
plt.figure()
plt.imshow(im_new, plt.cm.gray)
```

✓ 2.3s

```
<matplotlib.image.AxesImage at 0x1989bcdb6d0>
```




همانطور که می‌بینیم تصویر کمی smooth شده و نویز متناوب در بک‌گراند تصویر تقریبا از بین رفته است.

ب) سینتکس تابع لبه‌یاب canny به صورت زیر است:

```
cv2.Canny(image, T_lower, T_upper, aperture_size, L2Gradient)
```

پارامتر اول که تصویری است که می‌خواهیم لبه‌یابی کنیم. پارامتر دوم و سوم آستانه‌ی حد پایین و بالا، پارامتر چهارم اندازه‌ی فیلتر و پارامتر آخر یک مقدار Boolean برای افزایش دقت کار است.

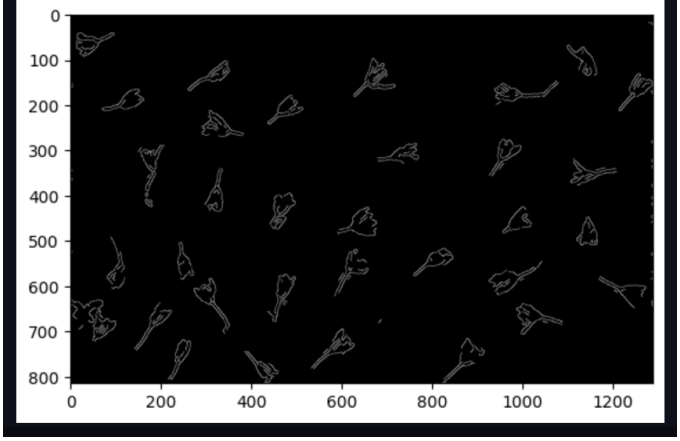
در این تابع تصمیم می‌گیرد که کدام لبه ها واقعاً لبه هستند و کدام نه. برای این کار به دو مقدار آستانه، minVal و maxVal نیاز داریم. هر لبه‌ای با گرادیان شدت بیشتر از maxVal مطمئناً لبه است و آنهایی که زیر minVal هستند مطمئناً لبه نیستند، بنابراین دور انداخته می‌شوند. آن‌هایی که بین این دو آستانه قرار دارند بر اساس اتصال آن‌ها لبه یا غیر لبه طبقه بندی می‌شوند.

در این سوال هم با امتحان کردن اعداد مختلف، بهترین مقادیر برای انتخاب این دو آستانه را پیدا کردیم و خروجی به صورت زیر در آمد که فقط گل‌های زعفران باقی مانده‌اند. (منبع)

```
image = cv2.imread("images/img_03.jpg")
im_edges = cv2.Canny(image,50,100)
cv2.imwrite("images/img_04.jpg", im_edges)
plt.figure()
plt.imshow(im_edges, plt.cm.gray)
```

✓ 0.4s

```
<matplotlib.image.AxesImage at 0x19887a48820>
```




ج) با استفاده از تابع sobel مشتق عمودی و افقی تابع را گرفته و با استفاده از تابع arctan2 جهت گرادیان را حساب می‌کنیم. سپس مقادیر را به درجه تبدیل می‌کنیم.

```
image = cv2.imread("images/img_04.jpg")
sobelX = cv2.Sobel(src = image, ddepth = cv2.CV_64F, dx = 1, dy = 0, ksize = 1)
sobelY = cv2.Sobel(src = image, ddepth = cv2.CV_64F, dx = 0, dy = 1, ksize = 1)
gradien_directions = np.arctan2(sobelX, sobelY) * 180 / np.pi
# print(sobelX)
# print(sobelY)
print(gradien_directions)
```

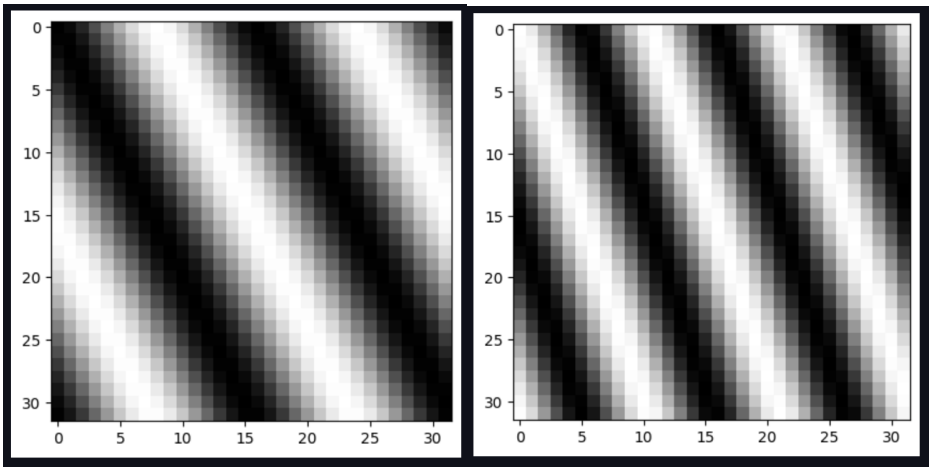
د) ابتدا باید تصویر را لبه‌یابی کنیم، سپس با استفاده از رای‌گیری و جهت گرادیان، جاهایی که مشتق خیلی سریع تغییر می‌کند را پیدا کنیم. یعنی در واقع دو خط موازی، سپس نقاط ابتدا و انتهای این خطوط را پیدا کرده و در ابتدا برش می‌زنیم.

۲. همانطور که می‌بینیم با شیفت پیکسل سفید به سمت راست خط‌های مورب تبدیل فوریه بیشتر شده و باریک‌تر شده‌اند و تغییرات فرکانسی بیشتر شده است.

```
mat = np.zeros((32, 32))
mat[31][2] = 255
plt.imshow(np.abs(mat), cmap="gray")

mat_fft = np.fft.fftshift(np.fft.fft2(mat))
plt.figure()
plt.imshow((mat_fft).real, cmap="gray")

mat[31][2] = 0
mat[31][3] = 255
mat_fft = np.fft.fftshift(np.fft.fft2(mat))
plt.figure()
plt.imshow((mat_fft).real, cmap="gray")
```



۳. همانطور که می‌بینیم مشتق عمودی این ماتریس صفر است اما در راستای افق تغییرات داریم.

```
mat1 = np.zeros((5,5))
for i in range(5):
    mat1[i][2] = 10
print(mat1)

mat2 = np.zeros((5,5))
mat3 = np.zeros((5,5))

mat2 = cv2.Sobel(src = mat1, ddepth = cv2.CV_64F, dx=1, dy=0, ksize=1)
mat3 = cv2.Sobel(src = mat1, ddepth = cv2.CV_64F, dx=0, dy=1, ksize=1)

print(mat2)
print(mat3)
```

```
[[ 0.  0. 10.  0.  0.]
 [ 0.  0. 10.  0.  0.]
 [ 0.  0. 10.  0.  0.]
 [ 0.  0. 10.  0.  0.]
 [ 0.  0. 10.  0.  0.]]
[[  0.  10.   0. -10.   0.]
 [  0.  10.   0. -10.   0.]
 [  0.  10.   0. -10.   0.]
 [  0.  10.   0. -10.   0.]
 [  0.  10.   0. -10.   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.]]
```

۴. با استفاده از روش گفته‌شده سر کلاس ابتدا مختصات نقاط آبی رنگ را پیدا می‌کنیم، سپس میانگین مجموعه‌ی x, y, x*y, x*x را حساب کرده و داخل فرمول‌های داخل اسلاید قرار می‌دهیم. به این ترتیب شیب خط و عرض از مبدا محاسبه می‌شود. سپس پاره خط مربوطه را رسم می‌کنیم.

```
image = cv2.imread("images/img_02.jpg", 0)
img = image.copy()
h = image.shape[0]
w = image.shape[1]
x = []
y = []
xy = []
xx = []
```

```
for j in range(h):
    for i in range(w):
        if(image[j][i] != 255):
            x.append(i)
            y.append(j)
l = len(x)

for i in range(l):
    xy.append(x[i] * y[i])

xx = [n * n for n in x]

meanX = np.mean(x)
meanY = np.mean(y)
meanXY = np.mean(xy)
meanXX = np.mean(xx)

m = (meanX * meanY - meanXY) / (meanX * meanX - meanXX)
c = meanY - m * meanX

p1 = (x[0], int(m * x[0] + c))
p2 = (x[-1], int(m * x[-1] + c))

print(m,c)
line = cv2.line(img, p1, p2, (255, 0, 0), 1)
plt.figure()
plt.imshow(line, cmap="gray")
plt.title('Line')
```

```
0.7274769723288649 148.25616245476817
Text(0.5, 1.0, 'Line')
```