گزارش تمرین پنجم بینایی ماشین

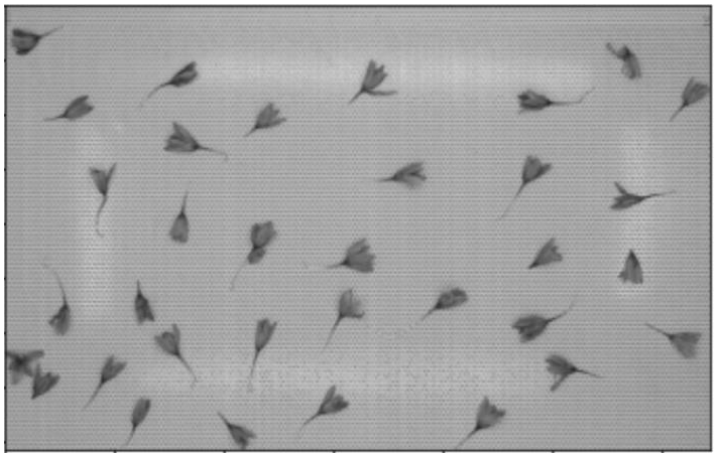
نام تهیه‌کننده: ملیکا نوبختیان

شماره دانشجویی: ۹۷۵۲۲۰۹۴

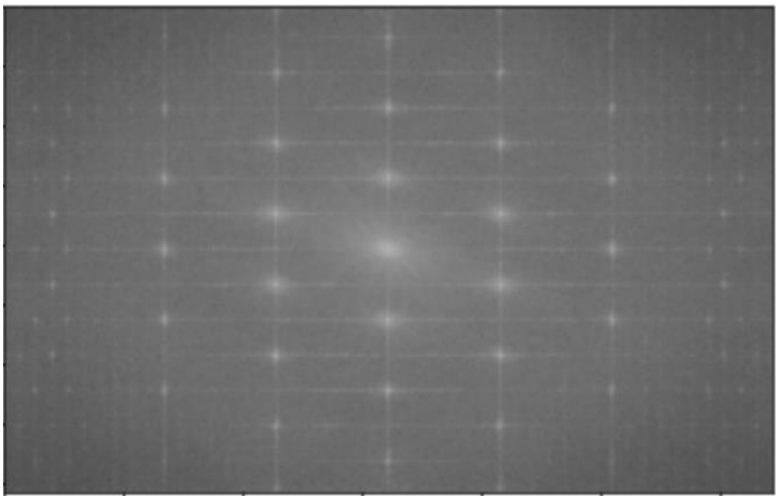
نسخه: ۱

۱- سوال اول

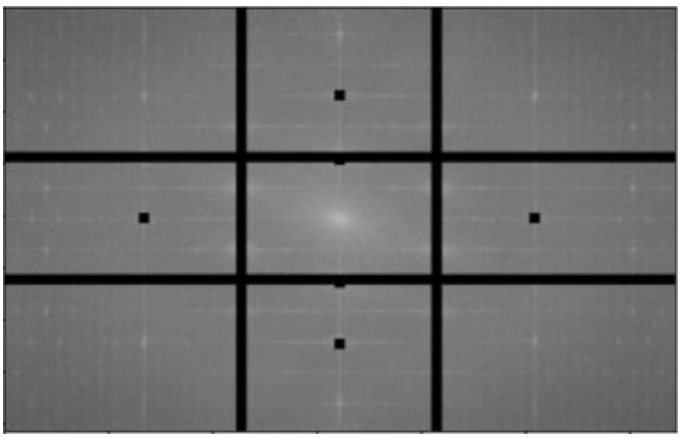
ابتدا تصویر اصلی را به صورت gray scale می‌خوانیم:



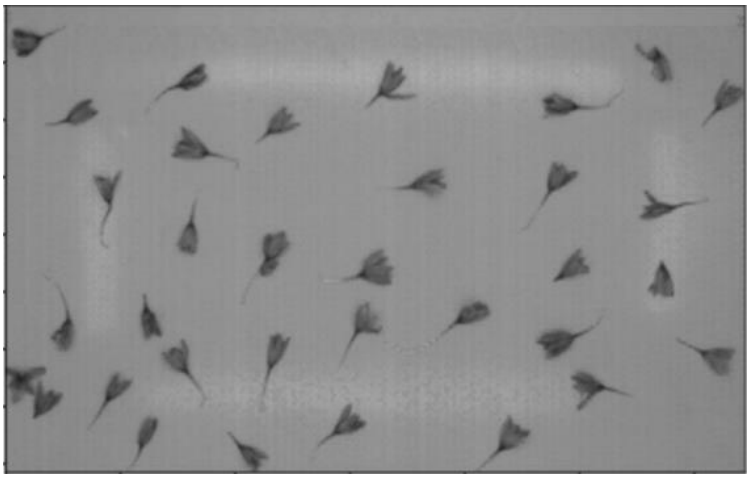
Fft تصویر به صورت زیر خواهد بود:



همان طور که در تصویر مشخص است غیر از نقطه روشن تصویر که peak دارد نقاط دیگری هم مشاهده می‌شوند که چنین وضعیتی را دارند. هم چنین یک سری خطوط مرتبط روشن نیز در فوریه داریم. ما یک سری از این موارد را به عنوان نویز در نظر می‌گیریم و به این شکل از فوریه حذف می‌کنیم:



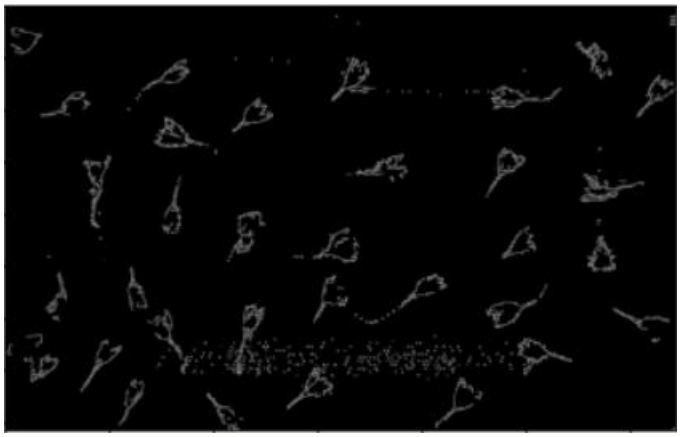
اگر از این تصویر inverse FFT بگیریم، تصویر denoised شده به شکل زیر خواهد بود:



نسبت به تصویر اولیه بهبود قابل لمس است زیرا نقاط دایره‌ای در پس زمینه تا حدود خوبی حذف شده‌اند.

نتیجه اعمال لبه‌یاب canny به این تصویر به صورت زیر خواهد بود:

```
edge_detect = cv2.Canny(original_image, 150, 400)
```



با توجه به شکل می‌بینیم که با تقریب نسبتا خوبی تنها گل‌های زعفران در تصویر باقی مانده‌اند. پارامترهایی که باید به تابع canny بدهیم، در ابتدا خود تصویر و سپس minVal و maxVal هستند. مقادیری که کمتر از minVal باشند به طور قطعی لبه نخواهند بود و مقادیری که بزرگ‌تر از maxVal باشند، حتما لبه خواهند بود. مقادیری که بین این دو حالت قرار می‌گیرند بسته به وضعیتشان یا لبه خواهند بود یا نه. اگر چنین پیکسلی به یک پیکسل لبه متصل باشد، آن هم لبه خواهد بود و در غیر این صورت به عنوان لبه شناخته نخواهد شد.

برای گرفتن مشتق از تصویر از عملگرهای sobel استفاده کردم:

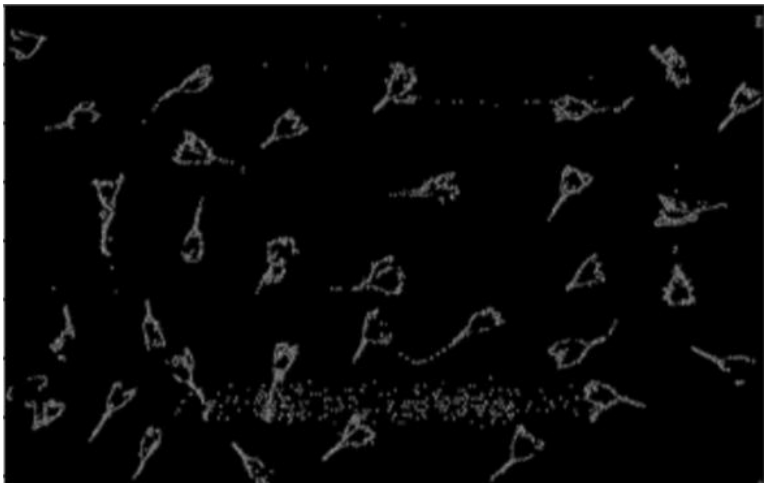
```
v_sobel = np.array([
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1],
], dtype='float64')

h_sobel = np.array([
    [-1, 0, 1],
```

```
    [-2, 0 ,2],
    [-1, 0, 1],
], dtype='float64')
```

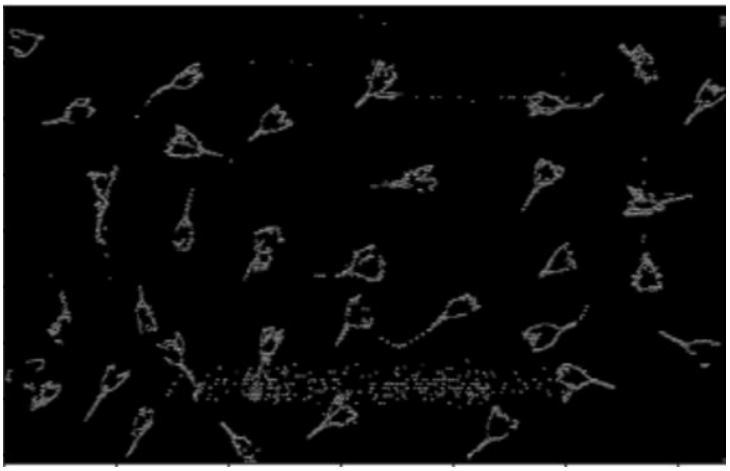
مشتق افقی تصویر به صورت زیر خواهد بود:

```
im_h = cv2.filter2D(edge_detect, ddepth=-1, kernel=h_sobel)
```



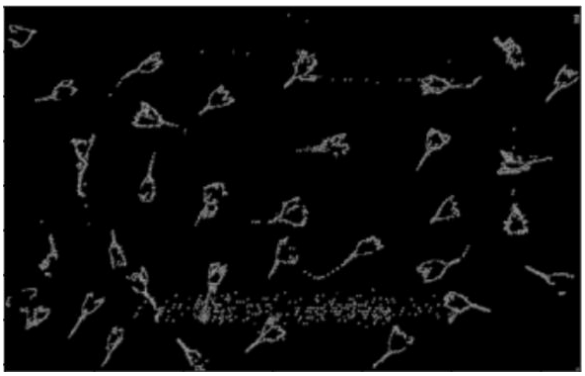
مشتق عمودی تصویر نیز به شکل زیر خواهد بود:

```
im_v = cv2.filter2D(edge_detect, ddepth=-1, kernel=v_sobel)
```



در نهایت گرادیان تصویر را به شکل زیر به دست می‌آوریم:

```
gradiant = np.power(im_h, 2) + np.power(im_v, 2)
gradiant = gradiant.astype('float64')
gradiant = np.sqrt(gradiant)
```



جهت گرادیان را نیز به شکل زیر به دست می‌آوریم:

```
arctan = np.arctan2(im_v, im_h)
array([[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., 0.,
..., 0., 0., 0.], [0., 0., 0., ..., 0., 0., 0.]], dtype=float16)
```

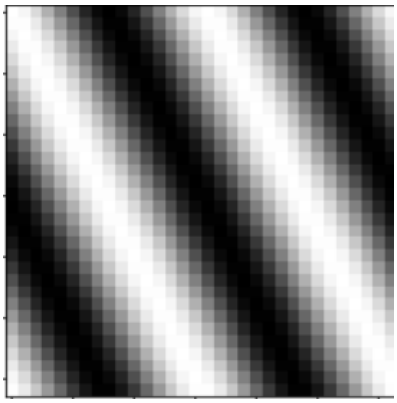
هر چند در اینجا به خوبی مشخص نیست ولی arctan دارای مقادیر غیرصفری است که جهت گرادیان را برای ما مشخص می‌کند.

ساقه گل‌های زعفران عموما به شکل یک خط صاف و مستقیم است. جهت گرادیان نیز از قسمت روشن به سمت قسمت تاریک است. اما در ناحیه گلبرگ ناگهان روند خط صاف و مستقیم عوض می‌شود وگلبرگ‌ها شکل و حالت دیگری به خود می‌گیرند. می‌توانیم همین تغییر ناگهانی در جهت گرادیان را به عنوان نمونه‌ای از نقطه جداسازی گلبرگ از ساقه زعفران در نظر بگیریم.

۲- سوال دوم

برای اینکه بهتر بتوانیم شکل تبدیل فوریه را ببینیم، یک آرایه تعریف کردیم و تنها عنصر موجود در سطر آخر ستون سوم آن را برابر ۱ قرار داده‌ایم. تبدیل فوریه آن به شکل زیر خواهد بود:

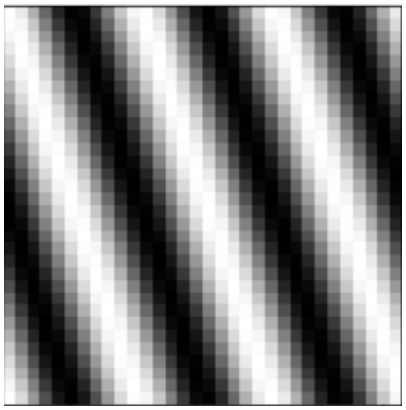
```
arr1 = np.zeros((32, 32))
arr1[31, 2] = 1
f_shift=np.fft.fft2(arr1)
plt.imshow(np.real(f_shift),"gray")
plt.show()
```



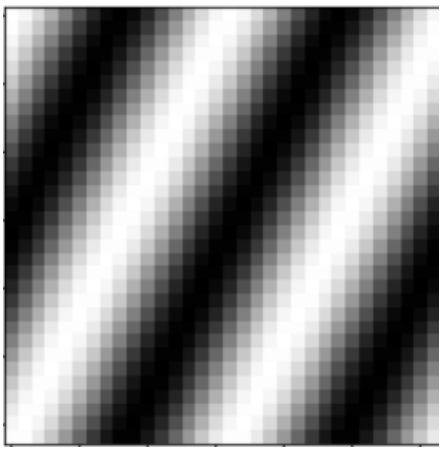
اگر این پیکسل را یک واحد به سمت راست انتقال دهیم، تبدیل فوریه به شکل زیر تغییر خواهد کرد:

```
arr1 = np.zeros((32, 32))
arr1[31, 3] = 1
f_shift=np.fft.fft2(arr1)
plt.imshow(np.real(f_shift),"gray")
```

```
plt.show()
```



نسبت به حالت قبل تعداد خطوط سفید بیشتر شده‌است و هم چنین زاویه آن‌ها با خط عمود سمت چپ کاهش پیدا کرده‌است. هر چقدر این روند پیشروی پیکسل به سمت راست را ادامه بدهیم، این روند هم ادامه خواهد داشت ولی در جایی دوباره روند برعکس شده و تعداد خطوط سفید کمتر می‌شود ولی این بار خطوط سفید عکس می‌شوند و روند افزایش و کاهش زاویه نسبت به خط عمود سمت راست خواهد بود. برای مثال اگر پیکسل شماره ۳۰ را یک کنیم فوریه به شکل زیر خواهد بود:



به طور کلی اینکه پیکسل کجا باشد و مولفه‌های تعیین کننده x و y آن کجا باشند، درتعداد و زاویه خطوط موثر خواهد بود و ما در اینجا تاثیر تغییر x در یک y ثابت را مشاهده کردیم.

۳- سوال سوم

ابتدا به شکل زیر یک ماتریس با لبه عمودی می‌سازیم:

```
np_arr = np.zeros((5, 5))
np_arr[:,3] = 1
```

```
np_arr[:,4] = 1
```

```
array([[0., 0., 0., 1., 1.],
       [0., 0., 0., 1., 1.],
       [0., 0., 0., 1., 1.],
       [0., 0., 0., 1., 1.],
       [0., 0., 0., 1., 1.]])
```

نتیجه اعمال عملگر مشتق افقی sobel به شکل زیر است:

```
np_h = cv2.filter2D(np_arr, ddepth=-1, kernel=h_sobel)
```

```
array([[0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.]])
```

نتیجه اعمال عملگر مشتق عمودی sobel به شکل زیر است:

```
np_v = cv2.filter2D(np_arr, ddepth=-1, kernel=v_sobel)
```

```
array([[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.]])
```

و در نهایت گرادیان تصویر به شکل زیر خواهد بود که چون مشتق عمودی برابر صفر بود، برابر همان مشتق افقی خواهد بود:

```
gradiant = np.power(np_h, 2) + np.power(np_v, 2)
gradiant = gradiant.astype('float64')
gradiant = np.sqrt(gradiant)
```

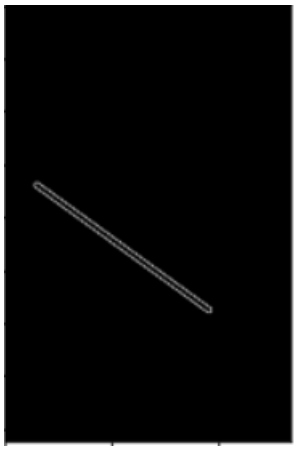
```
array([[0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.],
       [0., 0., 4., 4., 0.]])
```

۴- سوال چهارم

در قدم اول باید لبه‌یاب canny را روی تصویر اجرا کنیم تا نقاط موجود روی خط را پیدا کنیم:

```
edge_detect = cv2.Canny(original_image, 250, 300)
plt.imshow(edge_detect,"gray")
```

```
plt.show()
```



حالا طبق فرمول‌های زیر به محاسبه معادله خط می‌پردازیم:

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

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

ابتدا به محاسبه شیب خط می‌پردازیم و مقادیر عنصرهای موجود در فرمول را به دست می‌آوریم. تنها باید دقت کنیم چون در تصویر نقطه مبدا مختصات از گوشه بالا سمت چپ تصویر آغاز می‌شود در مورد مقادیر y باید دقت کنیم. من برای حل این مشکل مقادیر y را از طول تصویر کم کرده و منفی کردم. در نهایت به شکل زیر به محاسبه شیب خط پرداختم:

```
x_edge, y_edge = np.where(edge_detect > 0)[1] , -
np.where(edge_detect > 0)[0] + edge_detect.shape[0]
x_mean, y_mean = np.mean(x_edge), np.mean(y_edge)
xy_mean = np.mean(x_edge * y_edge)
x2_mean , xmean_2 = np.mean(np.power(x_edge, 2)), np.power(x_mean, 2)
m = ((x_mean * y_mean) - xy_mean ) / (xmean_2 - x2_mean)
```

در نهایت به مقدار زیر برای m رسیدم:

```
m = -0.7260330441850827
```

در محاسبه c نیز به شکل زیر عمل کردم و در نهایت به مقدار زیر رسیدم:

```
c = y_mean - m * x_mean
c = 263.5842782217144
```