Harris Corner Detection Report

September 6, 2019

This is the report for Q1 of Assignment 3. All the code is written in Python3 using NumPy for matrix operations, matplotlib for visualizations and Pillow for loading image. The `boat.mat` file contained a rotated 8-bit grayscale image of a boat which was saved as `boat.jpg` for easier access. The runtime of the algorithm ~ 2sec

1 Parameters Used

- k in corner-ness measure = 0.06
- Smoothing derivatives:
  - Window size = 5
  - $\sigma = 1.4$
- Weights for structure tensor:
  - Window size = 5
  - $\sigma = 1.4$

2 Image Derivatives

We've used Sobel operators of aperture = 3 to compute I_x and I_y. Since the Sobel operators are seperable filters (have rank 1), we have used the optimised algorithm for calculating the derivatives. We define the following function to filter an image I with a seperable filter whose factore are filter_y$_{(1\times3)}$ and filter_x$_{(3\times1)}$

[ ]:
```
def seperable_conv(I, filter_x, filter_y):
    h, w = I.shape[:2]
    n = filter_x.shape[0]//2
    I_a = np.zeros(I.shape)
    I_b = np.zeros(I.shape)
    for x in range(n, w-n):
        patch = I[:, x-n:x+n+1]
        I_a[:,x] = np.sum(patch * filter_x, 1)
    filter_y = np.expand_dims(filter_y, 1)
    for y in range(n, h-n):
        patch = I_a[y-n:y+n+1, :]
        I_b[y,:] = np.sum(patch * filter_y, 0)
    return I_b
```

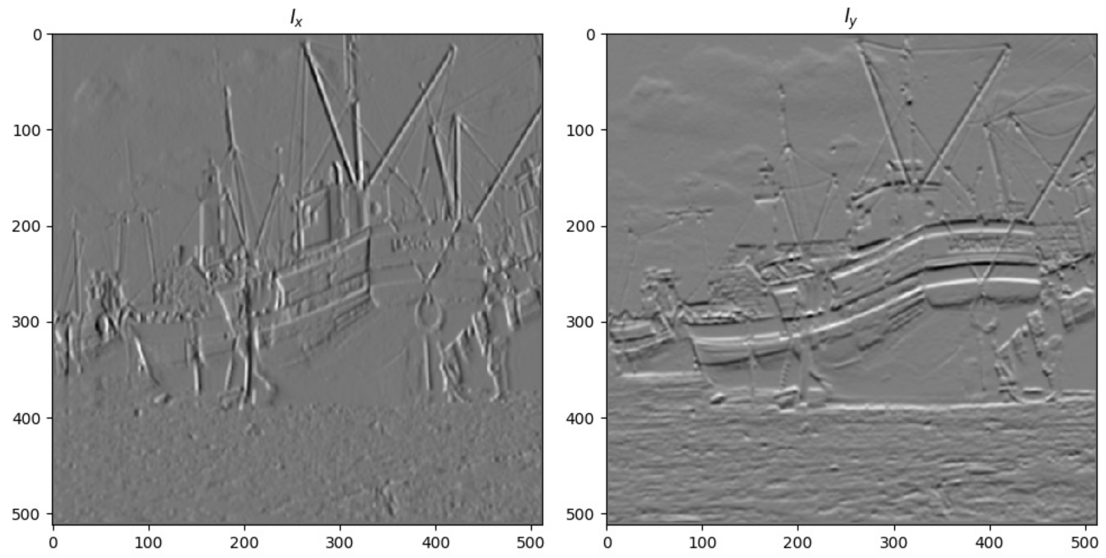
Now, for Sobel derivatives, we just take the filters as $[-1,0,1]$, $[1,2,1]^T$ and vice-versa.

```
[ ]: h, w = I.shape
     sobel_1 = np.array([-1, 0, 1])
     sobel_2 = np.array([ 1, 2, 1])
     I_x = seperable_conv(I, sobel_1, sobel_2)
     I_y = seperable_conv(I, sobel_2, sobel_1)
```

We now apply gaussian blur to smoothen the image. Since gaussian filter is also seperable, we take advantage of this fact and use the above functtion

```
[ ]: def gaussian_mask(n, sigma=None):
         if sigma is None:
             sigma = 0.3*(n//2) + 0.8
         X = np.arange(-(n//2), n//2+1)
         kernel = np.exp(-(X**2)/(2*sigma**2))
         return kernel

     g_kernel = gaussian_mask(n_g)
     I_x = seperable_conv(I_x, g_kernel, g_kernel)
     I_y = seperable_conv(I_y, g_kernel, g_kernel)
     D_temp = np.zeros((h,w,2,2))
```



3 Structure Tensor

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \Bigg|_{(u,v)}$$

Using smoothened derivatives, we calculate the structure tensor A at every pixel.

[ ]:
```
D_temp = np.zeros((h,w,2,2))
for y in range(1, h-1):
    for x in range(1, w-1):
        a, b = I_x[y,x], I_y[y,x]
        D_temp[y, x] = np.array([[a*a, a*b],
                                 [a*b, b*b]])
g_filter = gaussian_mask(n_w)
g_filter = np.dstack([g_filter]*4).reshape(n_w,2,2)
D = seperable_conv(D_temp, g_filter, g_filter)
```

4 Eigenvalues

Since A is a 2×2 matrix. It's eigenvalues have a closed form soultion.

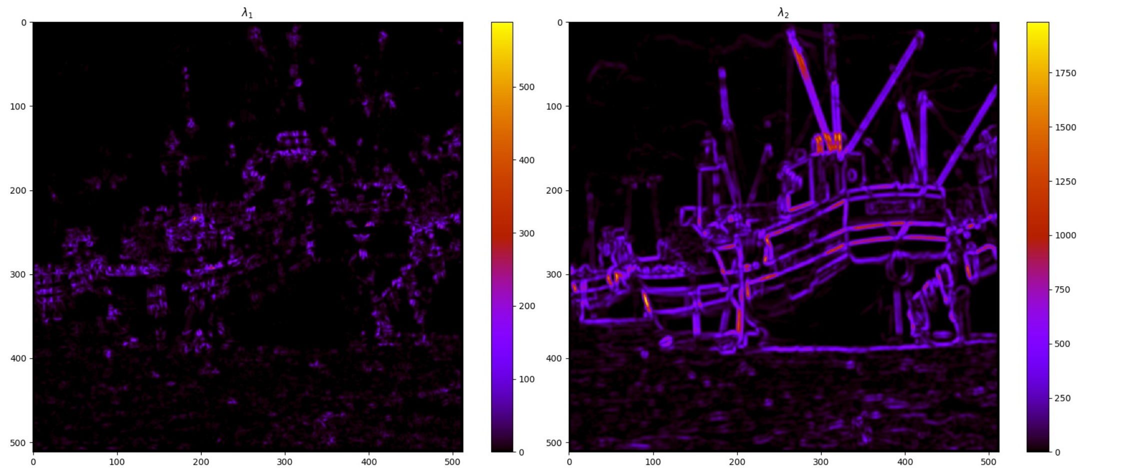
Let,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Then,

$$\lambda_1 = \frac{a+d}{2} - \frac{\sqrt{(a-d)^2 + 4bc}}{2}, \lambda_2 = \frac{a+d}{2} + \frac{\sqrt{(a-d)^2 + 4bc}}{2}$$

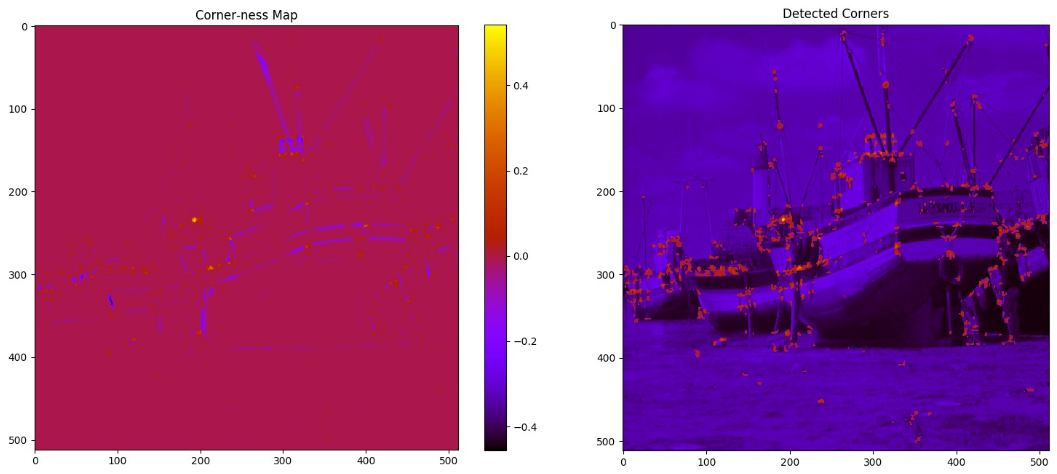
[ ]:
```
L_1 = np.zeros(I.shape)
L_2 = np.zeros(I.shape)
n = n_w//2
for y in range(n, h-n):
    for x in range(n, w-n):
        a,b,c,d = D[y,x].ravel()
        t1 = (a+d)/2
        t2 = np.sqrt((a-d)**2+4*b*c)/2
        L_1[y, x] = t1-t2
        L_2[y, x] = t1+t2
```



5 Corner-ness Measure

Corner-ness measure C was calculated as

$$C = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$


Corner-ness Map
Detected Corners