Plot and put in your report three 16x16 image patches from grace hopper.png loaded in grayscale.

Discuss in your report why you think it is good for the patches to have zero mean

It is beneficial to normalize the patches to have zero mean and unit variance because different points of the image may be illuminated separately, or different images may be illuminated separately. If I wanted to match a ball in two images which were subject to different illumination, having normalized patches would remove much of the variance caused by illumination.

Discuss in your report in 2-3 sentences, why the patches from the previous question would be good or bad for things like matching or recognizing an object. Consider how those patches would look if we changed the object's pose, scale, illumination, etc.

Patches are useful for the reasons listed in the previous question: we can create invariance to illumination and even scaling. It is also relatively straightforward to implement as opposed to using a laplacian filter and scaling to find blobs from every pixel. The downsides, however, are that context of the overall image is lost (think finding dominant orientation, especially between images taken at different angles. Useful pixels may spill into different patches at different angles) and success may be highly dependent upon the right patch size for the right image.

Show in your report that a convolution by a 2D Gaussian filter is equivalent to sequentially applying a vertical and horizontal Gaussian filter
Use the definition of the Gaussian filter for variables i and j in k:

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{(-\frac{i^2+k^2}{2\sigma^2})} \Rightarrow G_x(i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{i^2}{2\sigma^2})},\ G_y(j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{j^2}{2\sigma^2})}$$

$$G(i,j) = G_x(i) * G_y(j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{i^2}{2\sigma^2})} * \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{j^2}{2\sigma^2})} = \frac{1}{2\pi\sigma^2} e^{(-\frac{i^2}{2\sigma^2})} * e^{(-\frac{j^2}{2\sigma^2})}$$

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{(-\frac{i^2+j^2}{2\sigma^2})},$$

Thus for a filter of size k, at any given location in k, $(i_1, j_1)\ to\ (i_1, j_k)\ to\ (i_k, j_k)$

$G_x(i) * G_y(j) = G(i,j)$. Thus, it follows that $G_x * G_y = G$

Plot the following output and put it in your report and then describe what Gaussian filtering does to the image in one sentence. Load the image grace hopper.png as the input and apply a Gaussian filter that is 3 × 3 with a standard deviation of σ = 0.572.

Here the gaussian filter has smoothed the original image, removing some of the noise while blurring parts of the image.

Discuss in your report why it is a good idea for a smoothing filter to sum up to 1.
You want filter values to sum up to 1 so that the original image maintains its mean values. Meaning, if the filter values summed up to anything noticeably less than 1 the the intensity of the image would decrease by about that lower factor. Conversely, if the values of the filter summed to a value much greater than 1, the new image would have a much greater average intensity.

Derive in your report the convolution kernels for derivatives

i). We have a kernel $k_x \in R^{1\times3}$, which fits the equation

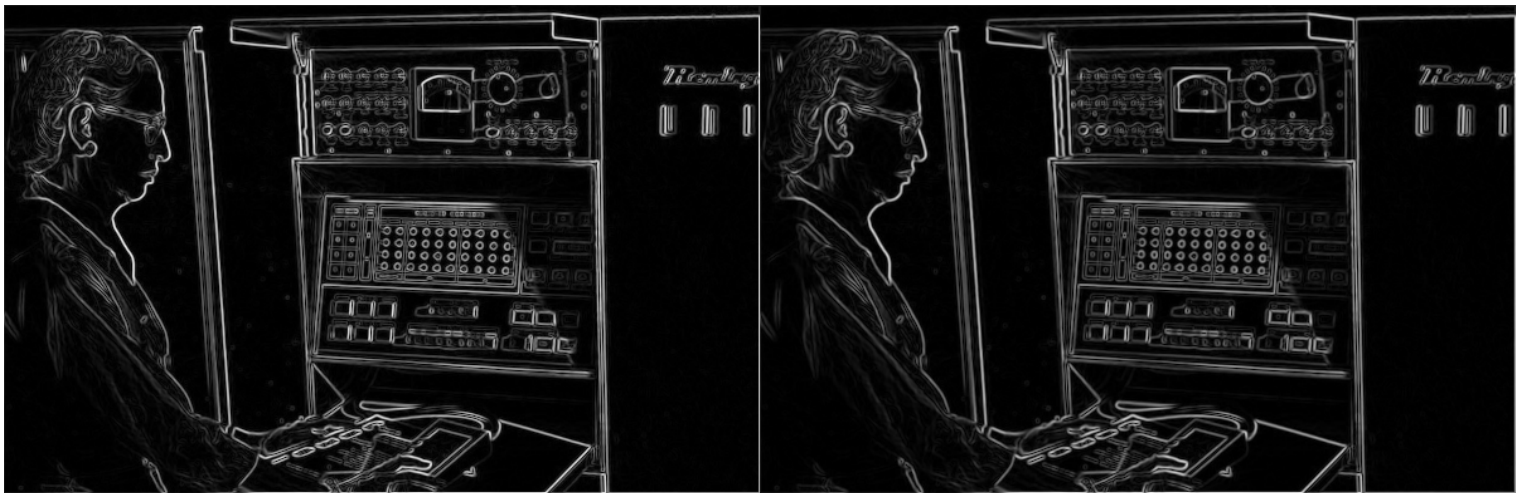
$I * k_x = I_x = I(x+1, y) - I(x-1, y)$.

A kernel of $k_x = [-1, 0, 1]\ @\ x = 1 \ => \ -I(0, y) + 0 * I(1, y) + I(2, y) = I(2, y) - I(0, y)$,

$I(x+1, y) - I(x-1, y) = I * k_x, \ where\ [-1, 0, 1]$ (i)

Then, $I(x, y+1) - I(x, y-1) = I * k_y, \ where\ [-1; 0; 1]$ (ii)

Use the original image and the Gaussian-filtered image as inputs respectively and use edge detection() to get their gradient magnitudes. Plot both outputs and put them in your report. Discuss in your report the difference between the two images in no more than three sentences



The images are very similar in output but the gaussian filtered image is much smoother. Many artifacts, noticeably on the jacket and the interface around the computer, have been smoothed over in the gaussian filtered image.

Show in your report the following result: If the input image is I and we use GS as our Gaussian filter, taking the horizontal-derivative (i.e., ∂ ∂x I(x, y)) of the Gaussian-filtered image, can be approximated by applying the Sobel filter (i.e., computing I ∗ Sx).

$\frac{\delta}{\delta x} I(x, y) = I * k_x$,

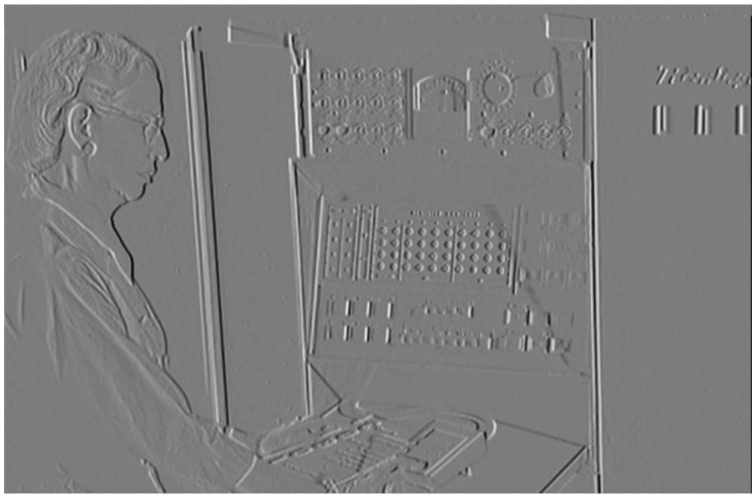
- apply the Gaussian filter, G_S, $I * G_S * k_x = \frac{\delta}{\delta x} I(x, y) * G_S = I * S_x$.

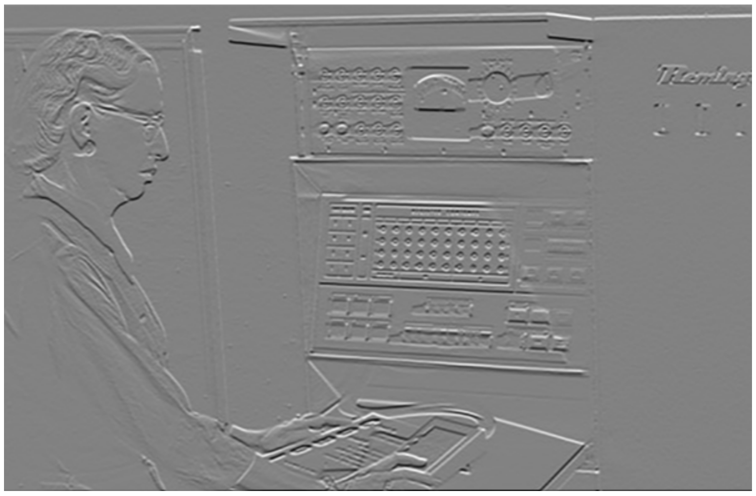
- $G_S * k_x = S_x$,

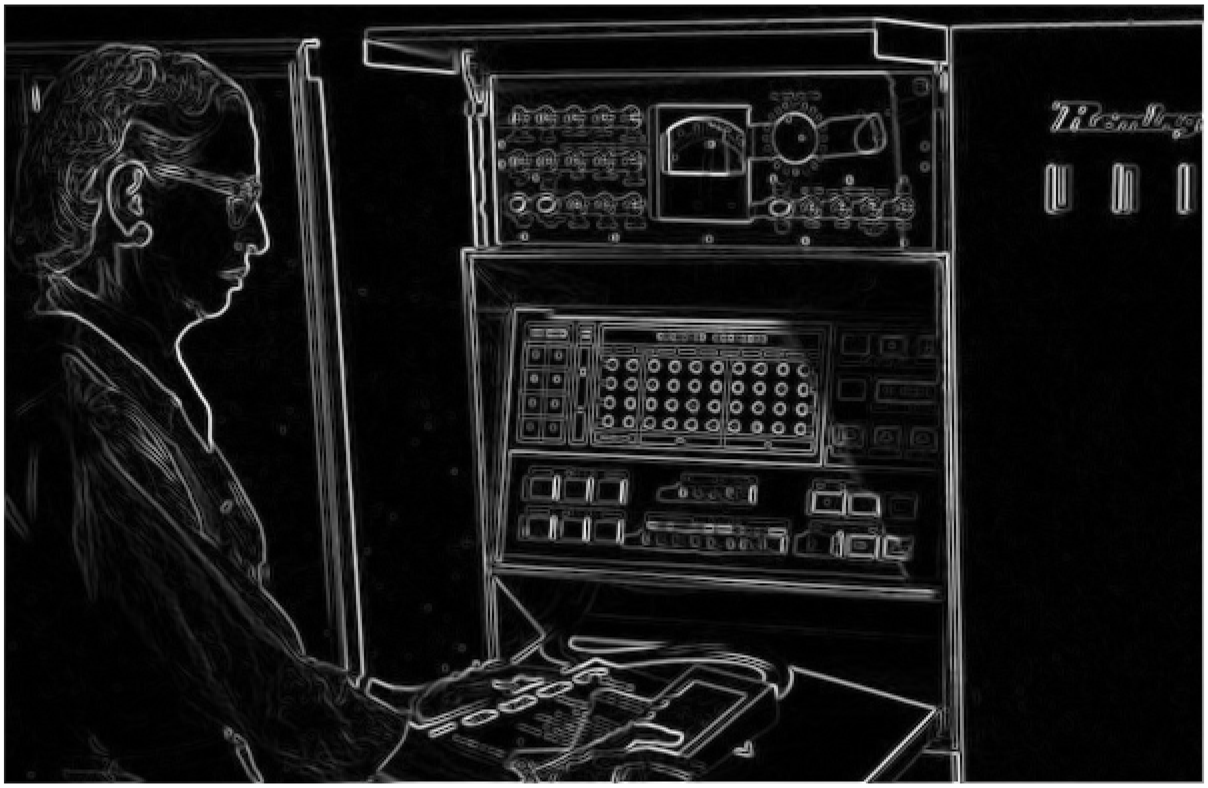
$G_S * k_x = [[1, 0, -1], [2, 0, -2], [1, 0, -1]]$ this is the same value as S_x,

thus$G_S * k_x = S_x$

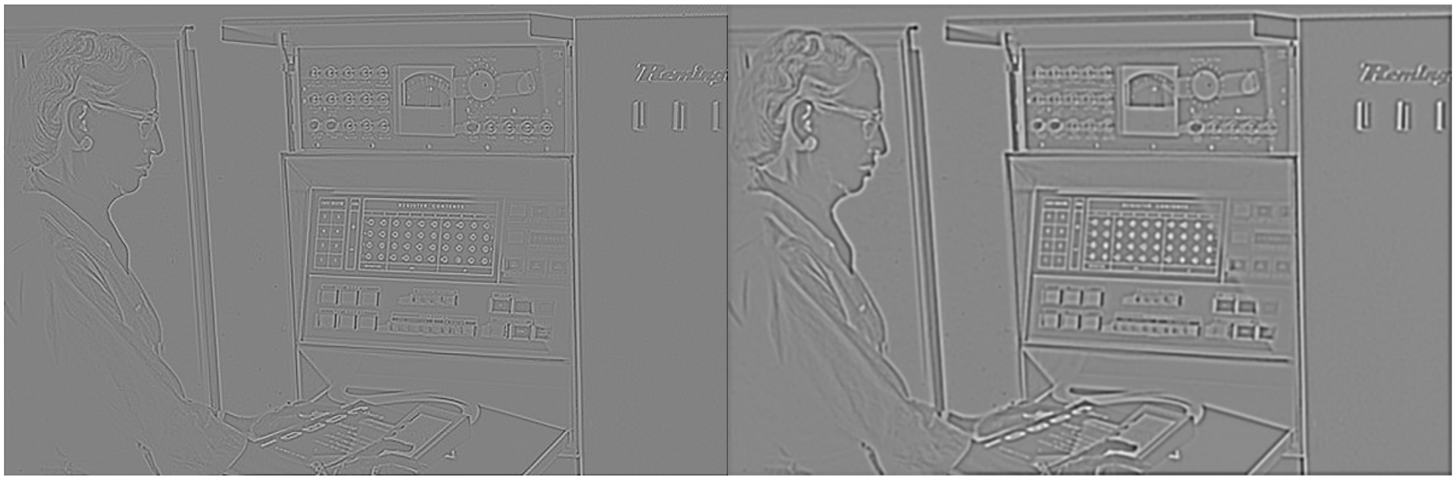
Plot the following and put them in your report: I ∗ Sx, I ∗ Sy, and the gradient magnitude. with the image 'grace hopper.png' as the input image I.







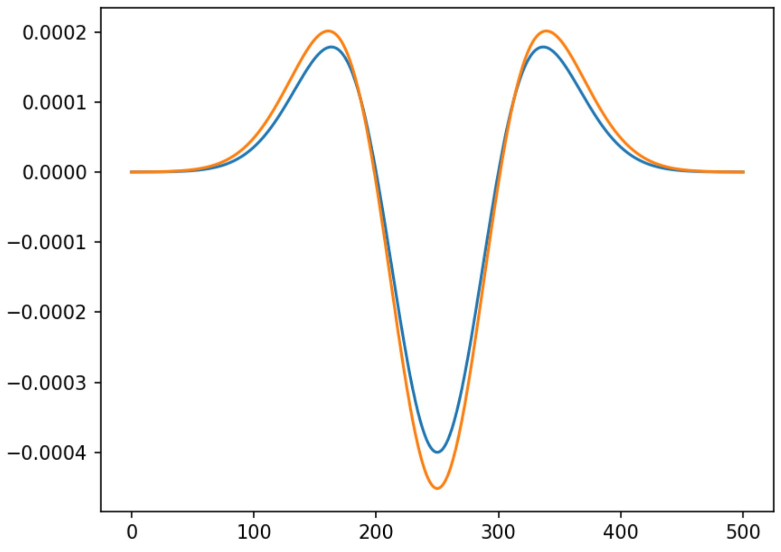
In filters.py, you are given two LoG filters. You are not required to show that they are LoG, but you are encouraged to know what an LoG filter looks like. Include in your report, the following: the outputs of these two LoG filters and the reasons for their difference. Discuss in your report whether these filters can detect edges. Can they detect anything else?



The first output is the result of a convolution with a 3x3 LoG filter, the second is with a 9x9 filter. Since the second filter is much larger, it will result in a much smoother image. LoG filters are ideal for detecting edges, so yes these could. These filters would also be useful for finding blobs. However, to do this you would need to find the size of the filter that maximizes response.

Discuss in your report why computing (I ∗ Gkσ) − (I ∗ Gσ) might successfully roughly approximate convolution by the Laplacian of Gaussian. You should include a plot or two showing filters in your report

The DoG is simply the difference of two Gaussian filtered versions of the image, where the first Gaussian has a larger standard deviation than the other. The larger Gaussian acts as a smoothing filter that removes noise and small details, while the smaller Gaussian acts as an edge detector. By subtracting the two filtered images, we get the DoG, which emphasizes the locations in the image where the intensity changes rapidly, which are the edges. Below, the blue line represents the LoG filter, and the orange line is the DoG filter.

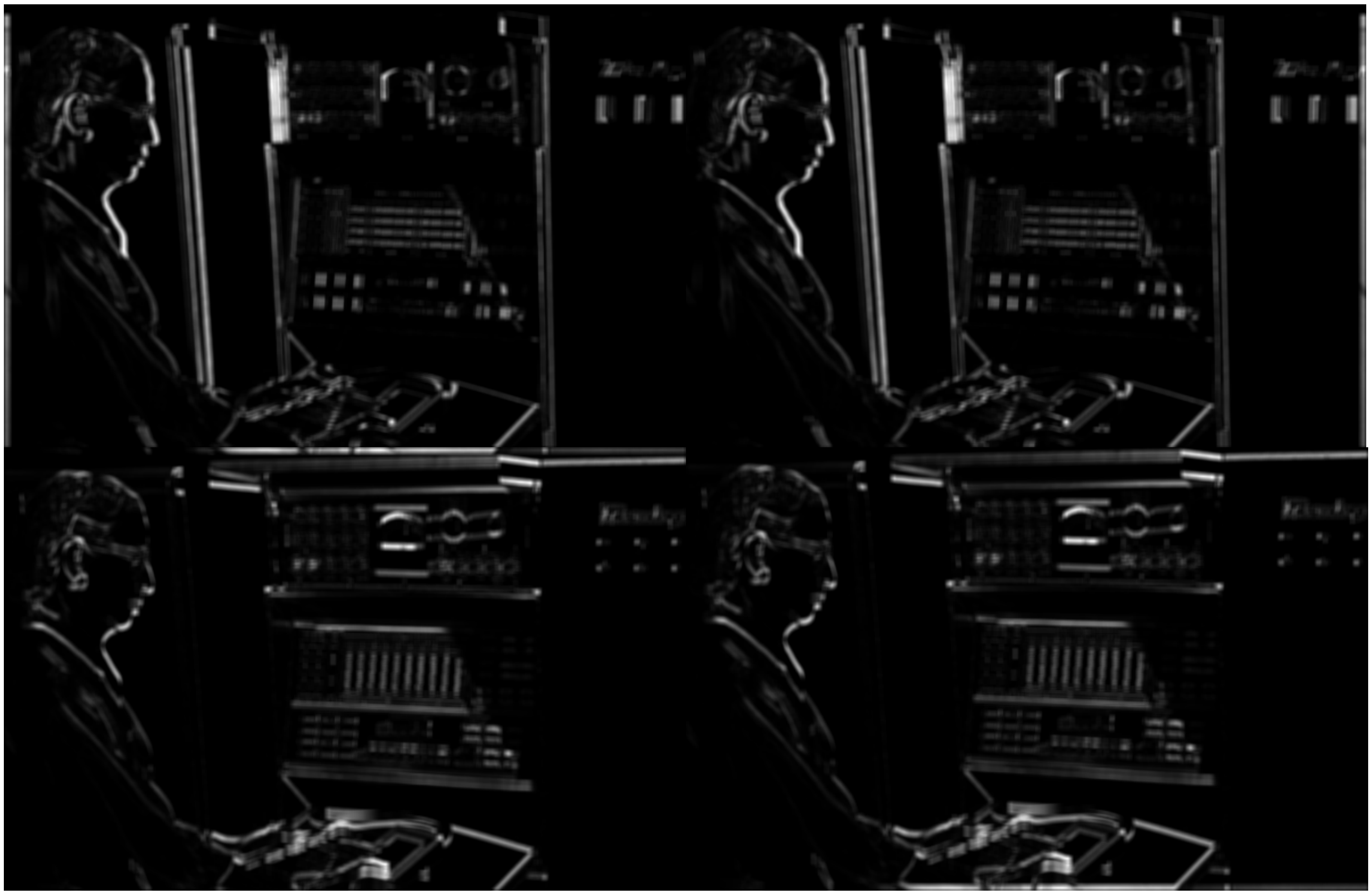


Write out each of the four remaining filters. No need to format them prettily; something like [[0,0,0], [0,1,0], [0,0,0]] works.

```
filter0 = np.diag([0, 1, 0])
filter1 = np.array([[1,1,1],
                [1,1,1],
                [1,1,1]])
filter2 = np.array([[.11,.11,.11],
                [.11,.11,.11],
                [.11,.11,.11]])
filter3 = np.array([[0,0,0],
                [1,0,0],
                [0,0,0]])
filter4 = np.array([[-1,0,1],
                [-1,0,1],
                [-1,0,1]])
```

What does filter 1 do, intuitively and how does it differ from filter 2?
Filter 1 blurs the image by summing all of the surrounding pixels into the output pixel. Filter 2 does the same but with each pixel contributing .11 to the intensity of the output pixel as opposed to a whole 1.0 for filter 1.

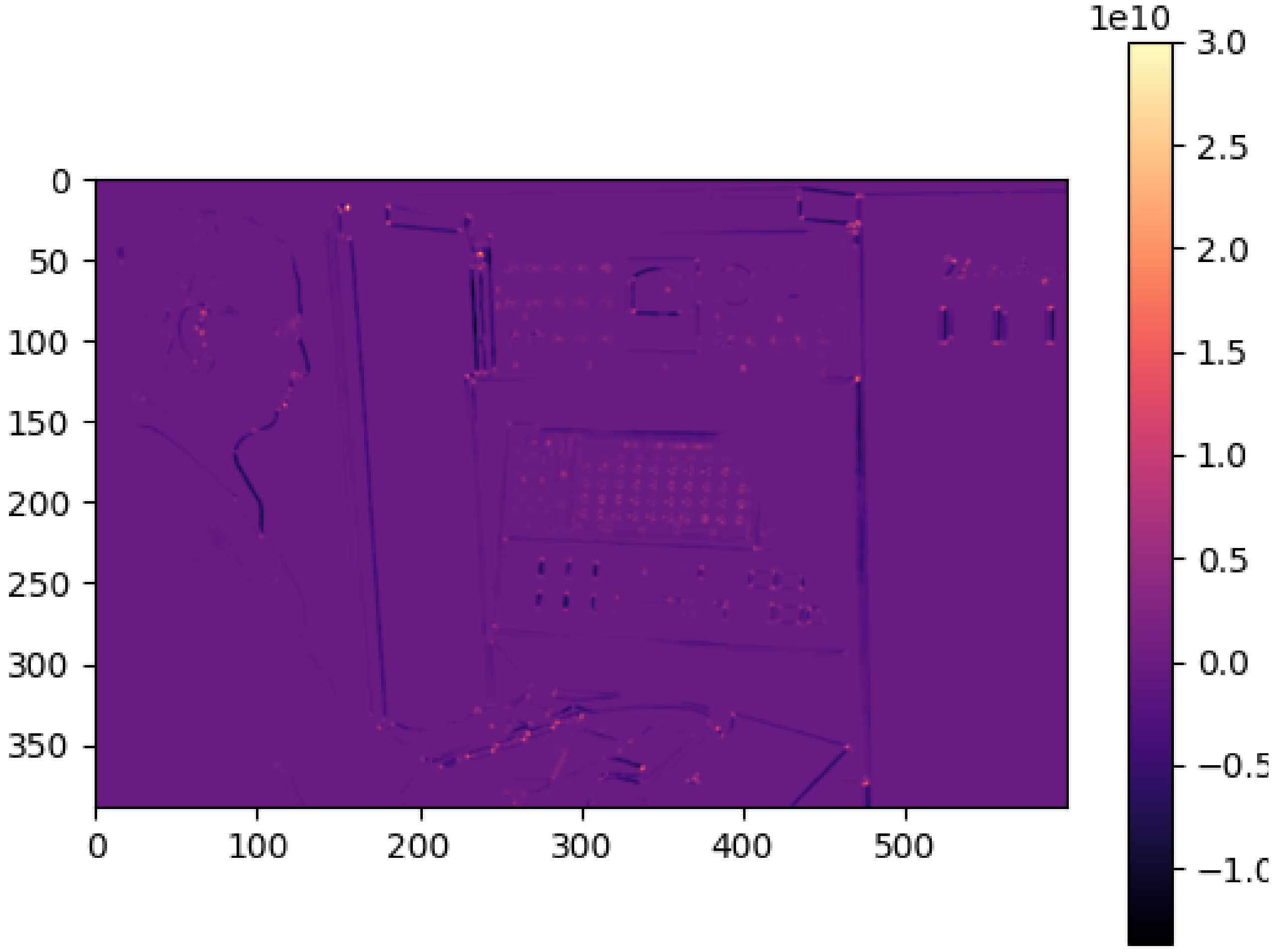
Plot and put in your report your output for for grace hopper.png for (u, v) = {(0, 5),(0, −5),(5, 0),(−5, 0)} and window size (5, 5)



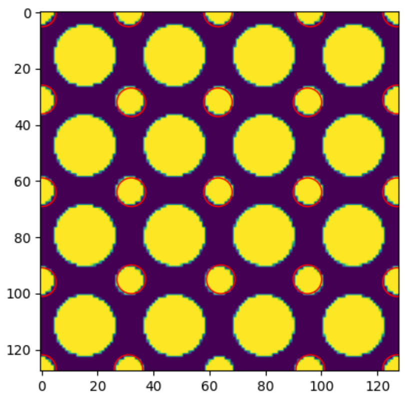
Discuss in your report why checking all the us and vs might be impractical in a few sentences.
Checking all the us and vs would take much longer and would provide minimal benefits. The goal of corner detection is to determine if there is a large change in intensity with a change in both the x and y directions, so just testing changes in x and y independently at one (or just a few) magnitude(s) is all that is necessary.

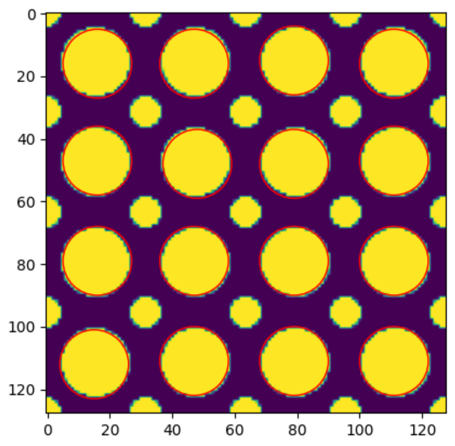
.

Generate a Harris Corner Detector score for every point in a grayscale version of ’grace hopper.png’, and plot and include in your report these scores as a heatmap.



Plot and include in your report the two responses and report the parameters used to obtain each. Comment in your report on the responses in a few lines: how many maxima are you observing? are there false peaks that are getting high values?
Sigma values:
Small - 5.1, 5.35
Large - 11, 11.5





Observing same # maxima as there are dots for respective tests. No false peaks were strong enough to cause incorrect output, but in the output for the small polka DoG you could see local maxima around the larger dots. This was not true, however, for the smaller dots in the large DoG

Find and include in your report a set of parameters for generating the scale space and finding the maxima that allows you to accurately detect the cells in each of those images. Feel free to pre-process the images or the scale space output to improve detection. Include in your report the number of detected cells for each of the images as well.

Setting r1 and r2 to be 5 & 5.35 respectively
I used images 2, 4, 5 & 6
Detected TP cells:
Cell2 - ~70 cells
Cell4 - ~64 cells
Cell5 - ~65 cells
Cell6 - ~92

Include in your report the visualized blob detection for each of the images and discuss the results obtained as well as any additional steps you took to improve the cell detection and counting. Include those images in your zip file under cell detections as well.
In order to minimize the amount of FPs in the image and make the cell blobs clearer I thresholded the image at 12, where pixel intensities below the value were set to 0.