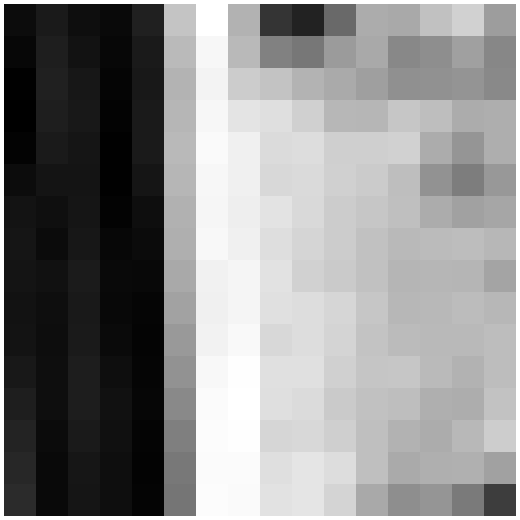
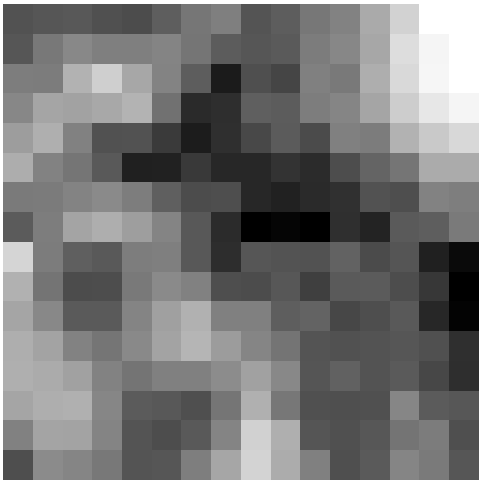
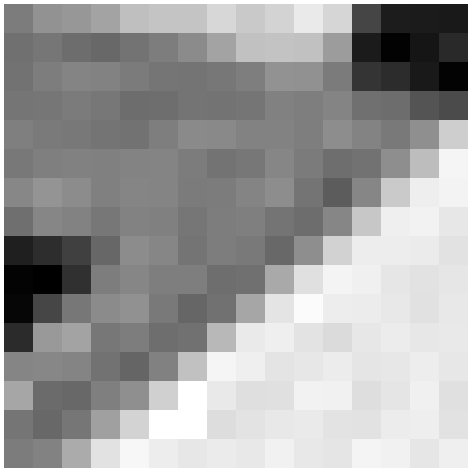
Image Filtering

Image Patches

1)





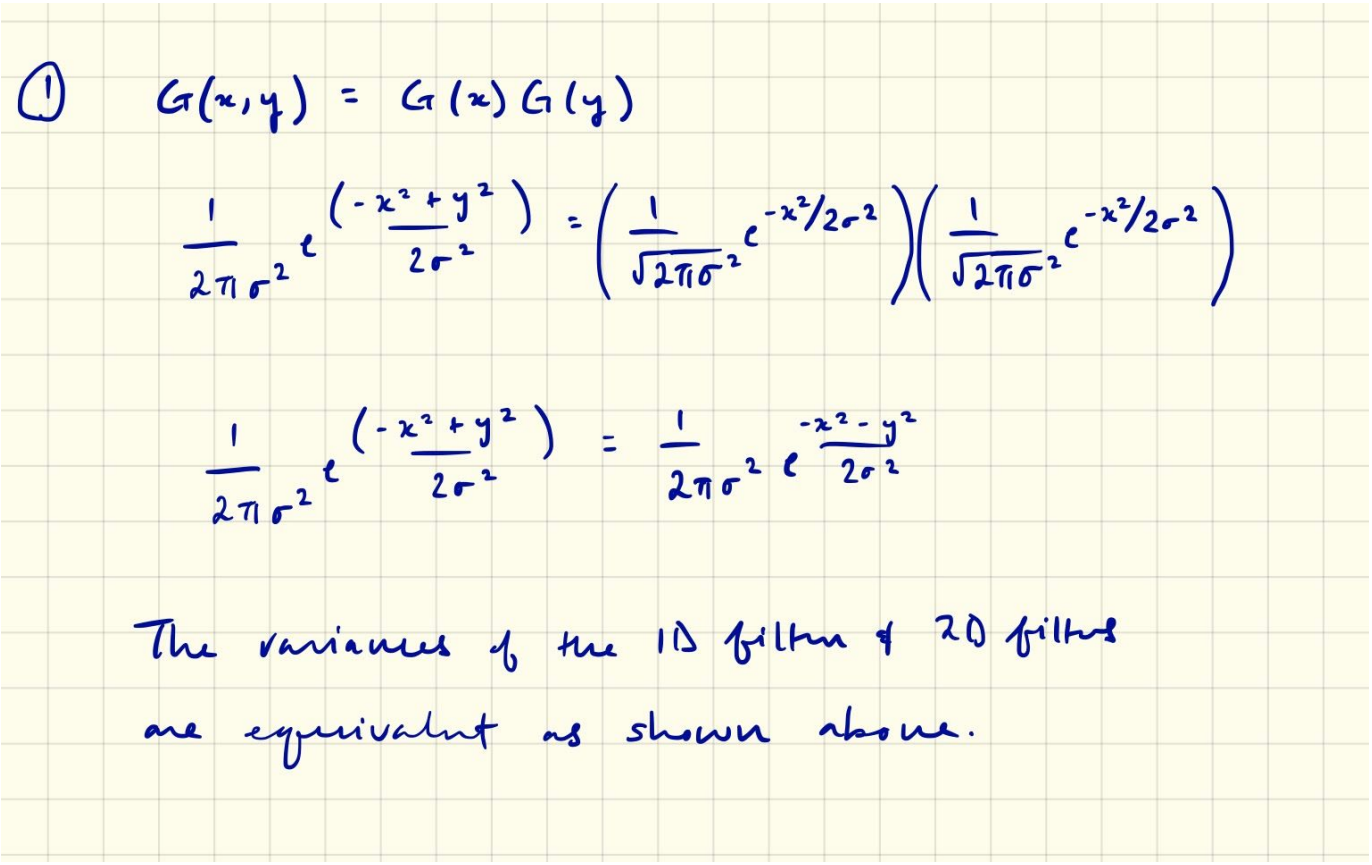


2) Image patches have certain use cases where they serve as good descriptors and other instances where they're poor descriptors. Patches generally provide the best advantage when they're unique. For instance, if we wanted to find the outline of an object's path in an image, the patch for this outline would be quite unique. Something like template matching would be able to leverage this unique property where the colors are different at

the edge. On the flip side, not all patches are so uniquely recognizable and these images are not quite invariant enough to changes in lighting and viewing position.

Gaussian Filter

1) We know that a 2D Gaussian filter is equivalent to the convolution of 2, one-dimensional Gaussian filters. This is demonstrated in Lecture 8. The two-dimensional kernel at G(x, y) is the outer product of the 2 one-dimensional kernels at a given point - x, y. The derivation below confirms this.



2) A Gaussian filter is usually used to blur an image or reduce noise.

3)

(3)

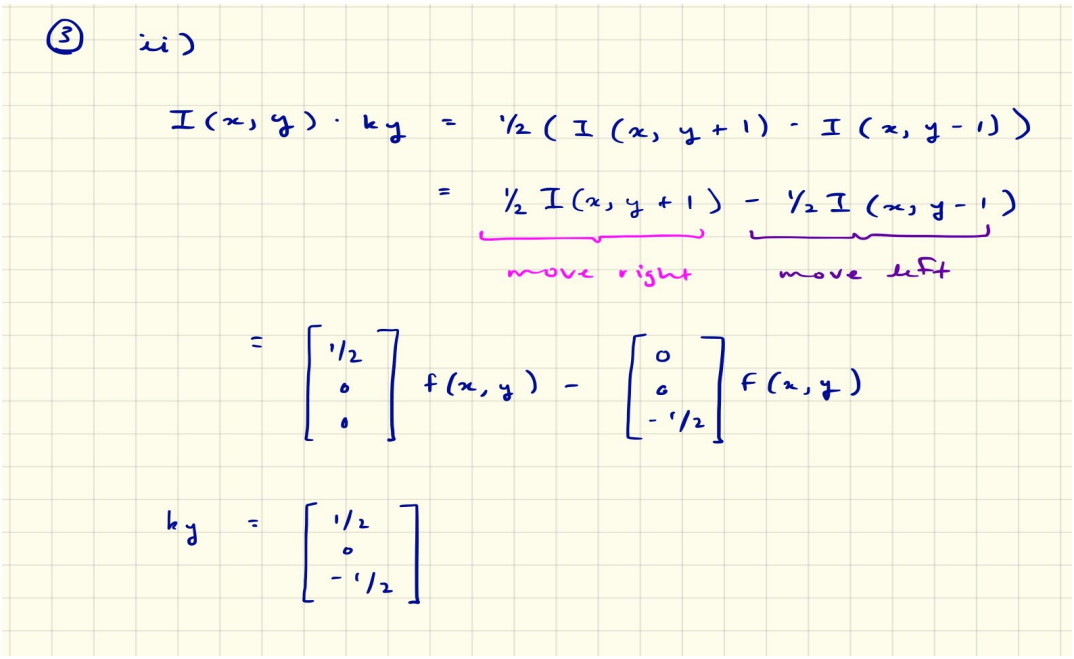
i) $I(x,y) \cdot k_x = \frac{1}{2}\left(I(x+1,y) - I(x-1,y)\right)$

$= \underbrace{\frac{1}{2} I(x+1,y)}_{\text{shift up}} - \underbrace{\frac{1}{2} I(x-1,y)}_{\text{shift down}}$

$= \begin{bmatrix} 1/2 & 0 & 0 \end{bmatrix} f(x,y) -$

$\begin{bmatrix} 0 & 0 & 1/2 \end{bmatrix} F(x,y)$

$k_x = \begin{bmatrix} 1/2 & 0 & -1/2 \end{bmatrix}$



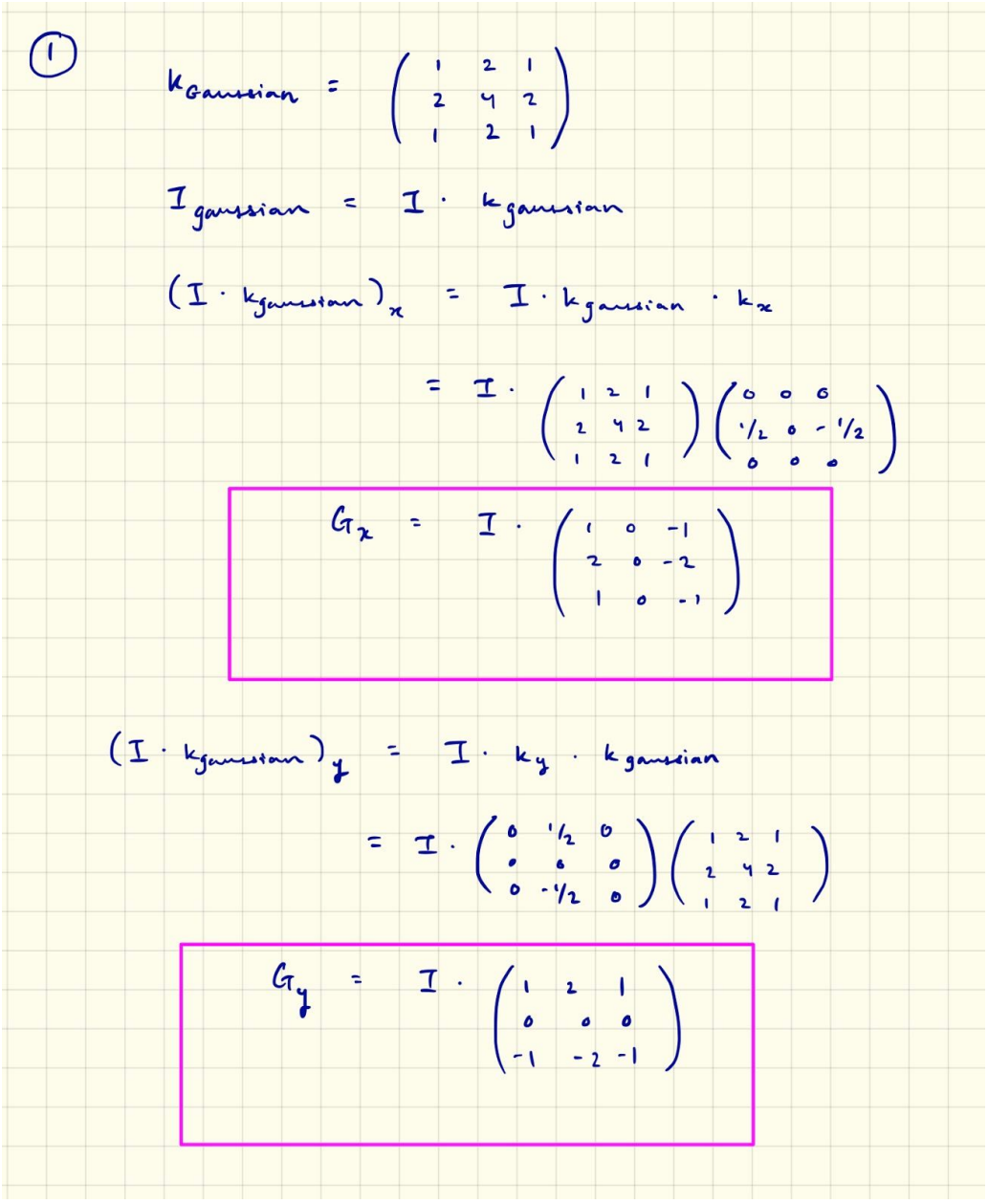
4) Compared to the original image with edge detection, the gaussian filtered image with edge detection resulted in much sharper edges and greater fidelity. For instance, the buttons on Grace Hopper's machine as well as some of the lines on her arms are better defined in the gaussian filtered image with edge detection.

Original image with edge detection

Edge detection with gaussian filter applied beforehand

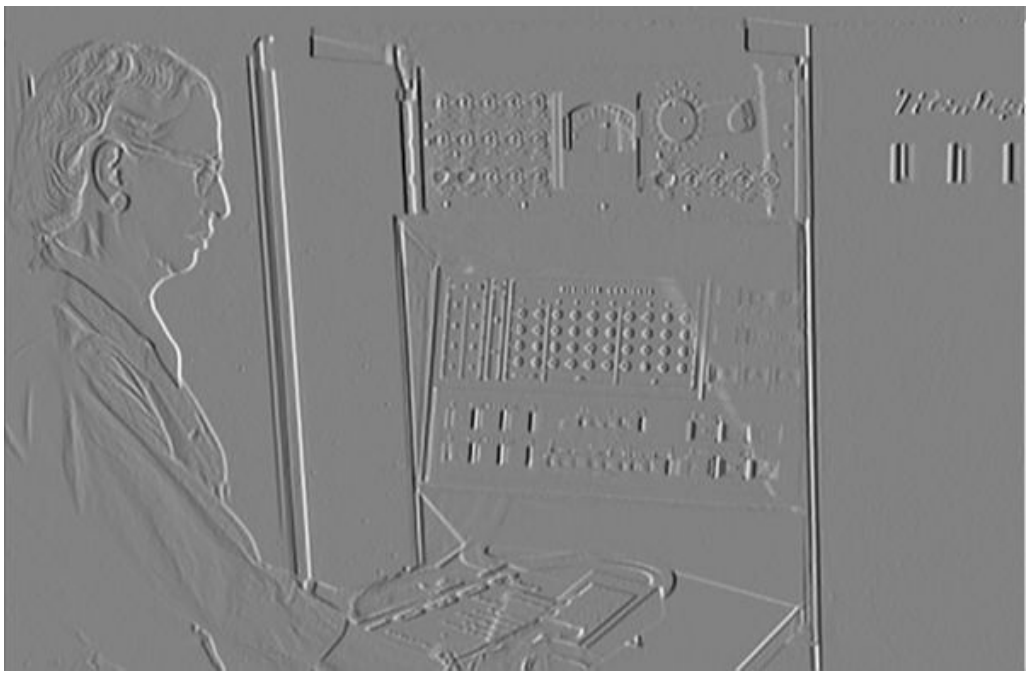
Sobel Operator

1) Derivations.

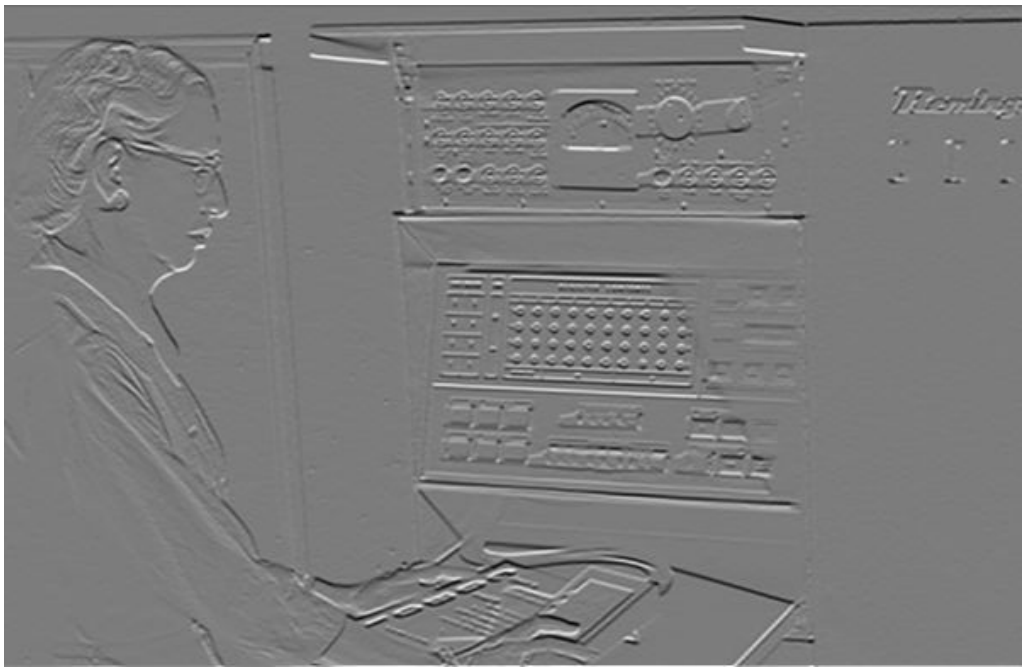


2) Sobel operators.

Gx



Gy



Sobel Edge Detection

3) Linear combination of two Sobel operators.

a) Derivations.

(3)

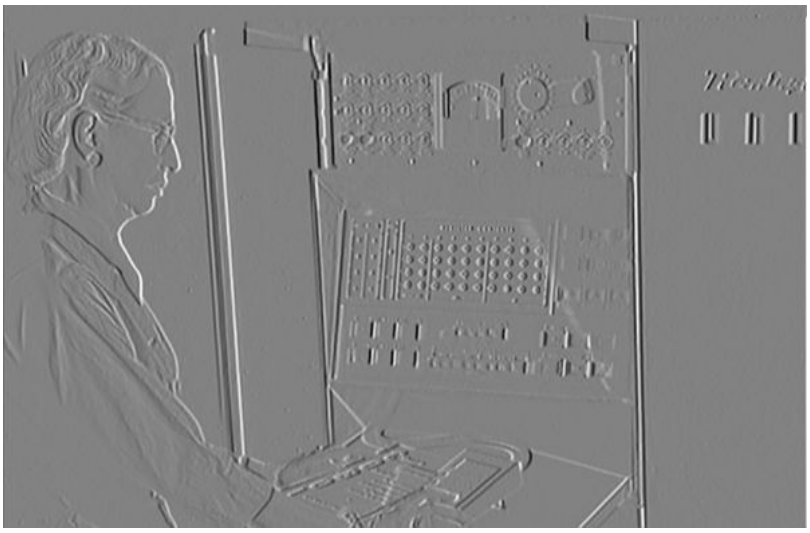
$$S(I,\alpha) = G_x \cos(\alpha) + G_y \sin(\alpha) = I \cdot k(\alpha)$$

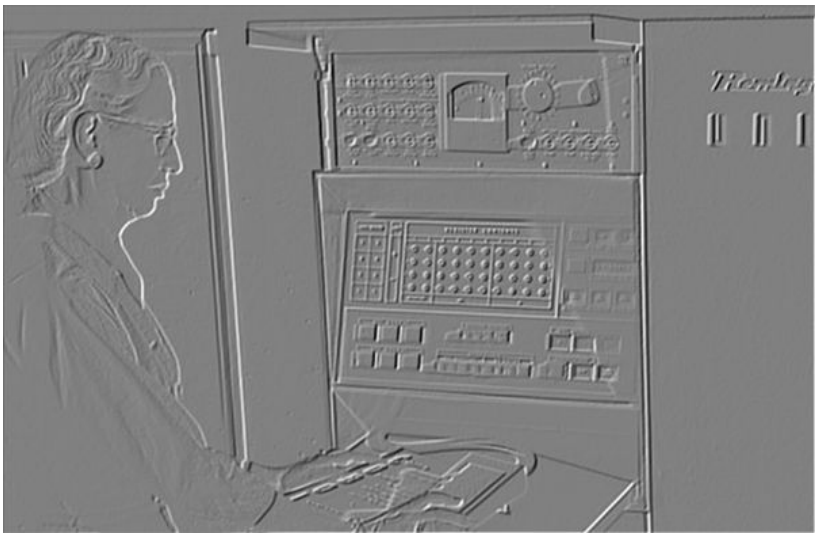
$$= I \begin{pmatrix} \cos\alpha & 0 & -\cos\alpha \\ 2\cos\alpha & 0 & -2\cos\alpha \\ \cos\alpha & 0 & -\cos\alpha \end{pmatrix} +$$

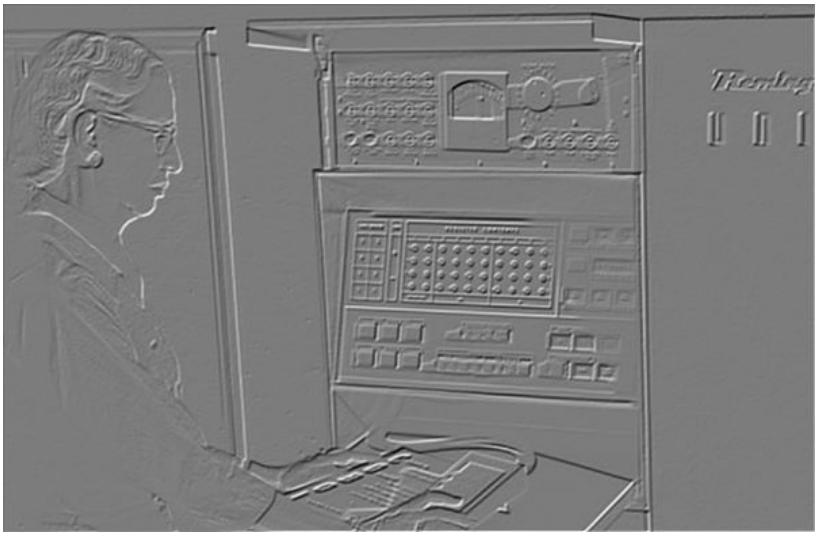
$$I \begin{pmatrix} \sin\alpha & 2\sin\alpha & \sin\alpha \\ 0 & 0 & 0 \\ -\sin\alpha & -2\sin\alpha & -\sin\alpha \end{pmatrix}$$

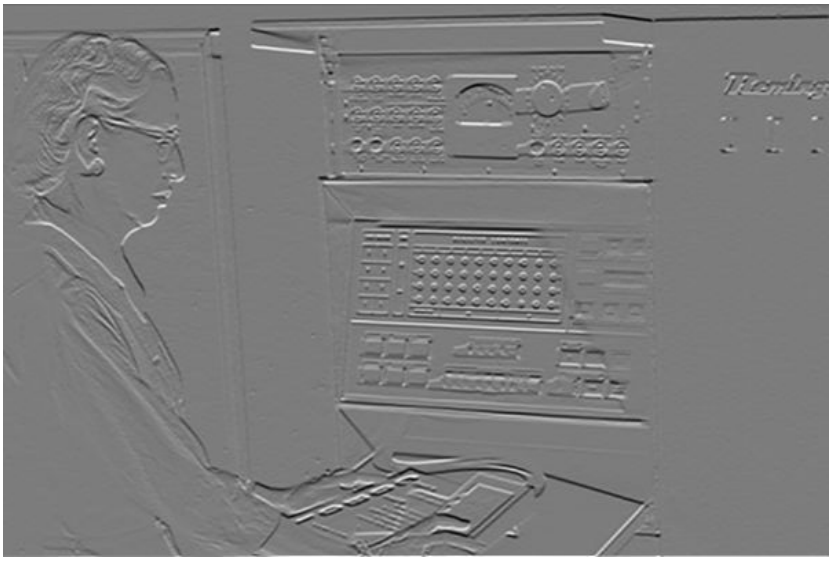
$$k(\alpha) = \begin{bmatrix} \cos\alpha + \sin\alpha & 2\sin\alpha & -\cos\alpha + \sin\alpha \\ 2\cos\alpha & 0 & -2\cos\alpha \\ \cos\alpha - \sin\alpha & -2\sin\alpha & -\sin\alpha - \cos\alpha \end{bmatrix}$$

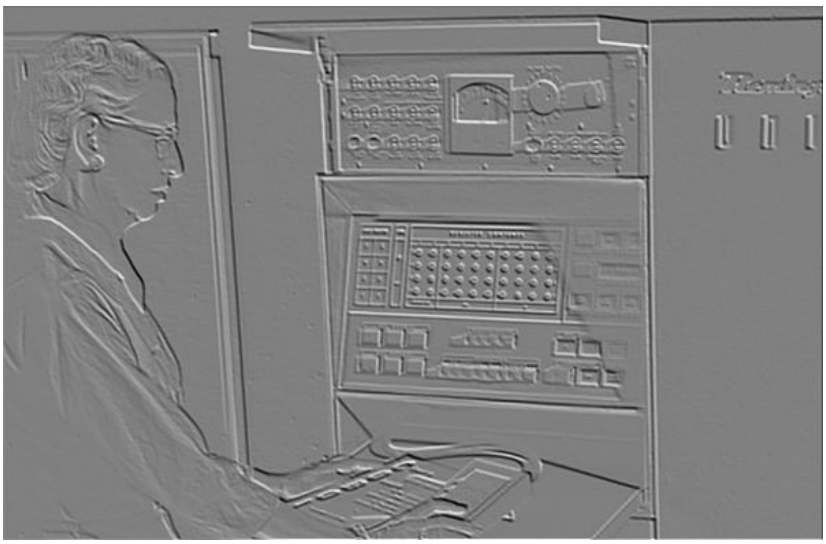
b) Steerable filter's output.

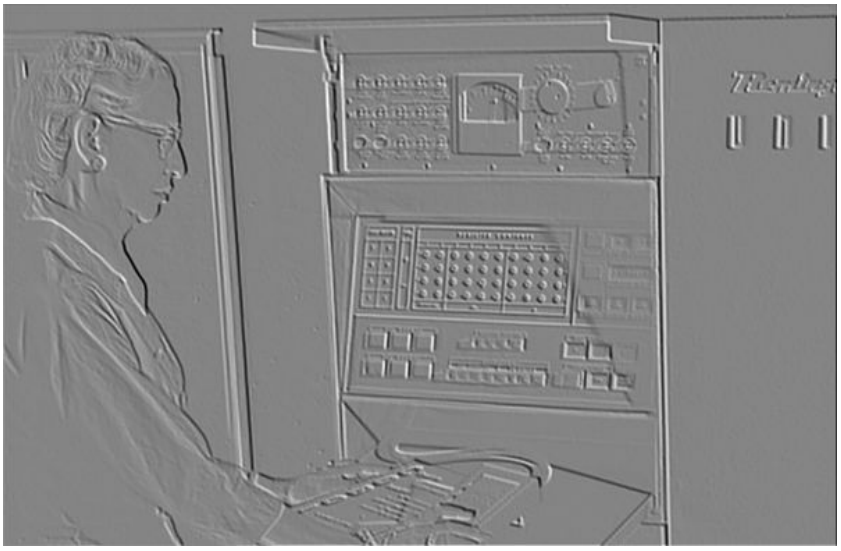








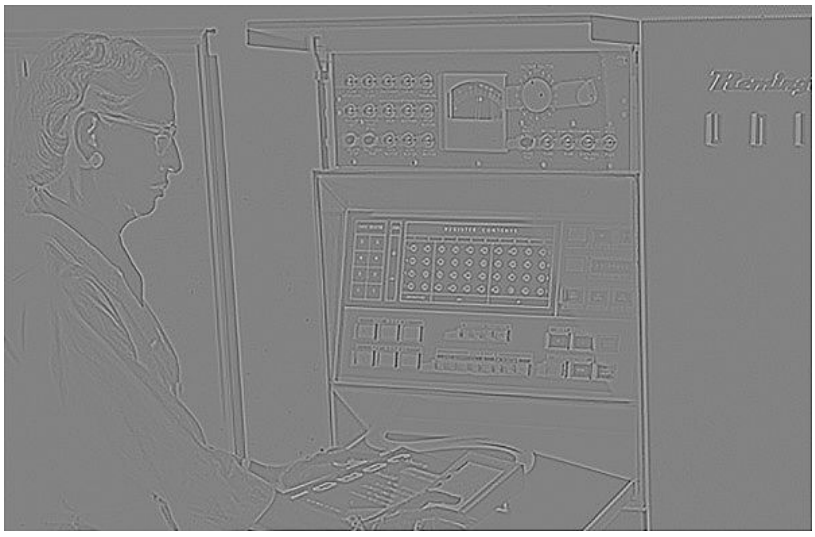




c) A steerable filter is still detecting edges. Alpha changes the orientation at which we detect these edges. For instance, as alpha increases the edges become more distinct.

LoG Filter

1) The edges in the first image are much more pronounced while the second image still has the edges, but they're much more fine and nuanced. I think this is likely because the filter for the second image is more exhaustive than the filter for the first image. I think this filter would also be quite competent at detecting corners given that corners are signaled by significant transitions in pixel values at the edges.



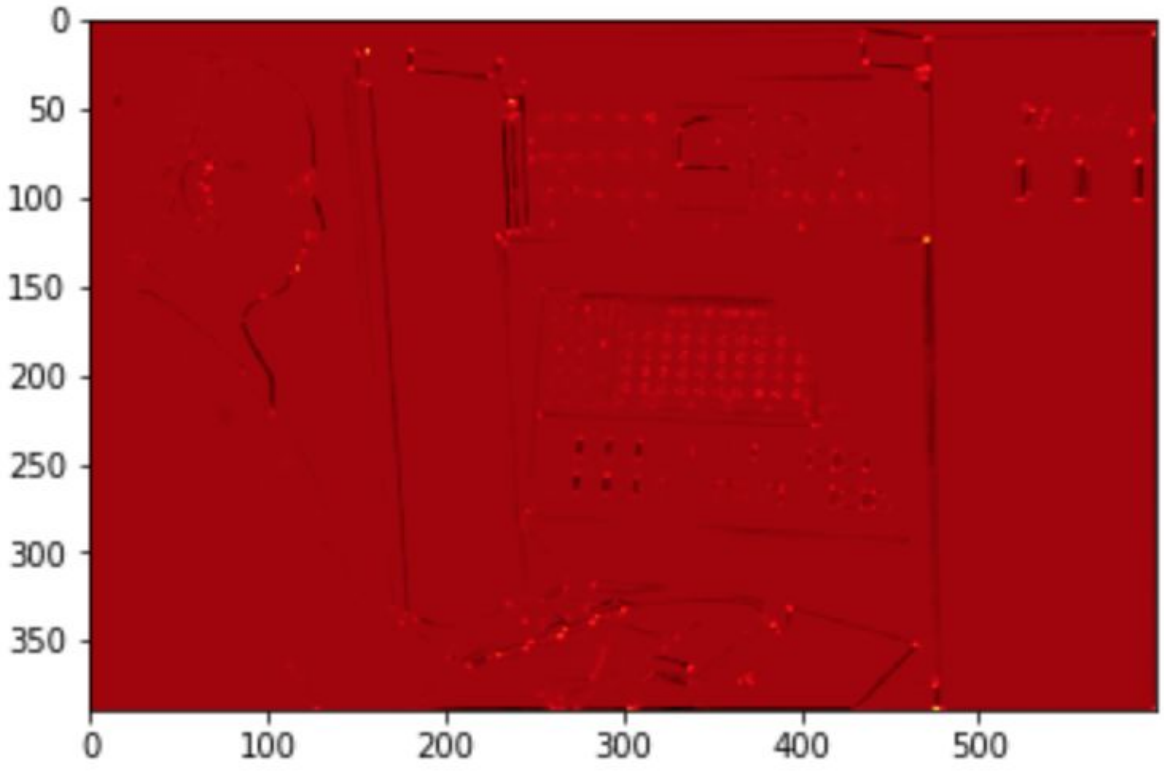
2) We use the difference of Gaussians to make this process more efficient. Using these separable filters is much faster than using a Laplacian in a two dimensional convolution. Because the Laplacian is a second derivative of a Gaussian, it's roughly similar to two

Gaussians with different sigmas. Therefore, if we use the difference of the Gaussians to create a scale space, it’s also roughly similar to a scale space formed with a Laplacian

Feature Extraction

1) One of the reasons this isn't quite practical is because edges not just corners match the criteria described by Moravec. For instance, a pixel on the edge, but not on the corner could feasibly have a score that's high for all offsets. In other words, lot of false positives make this a slightly weak approach to detecting corners.

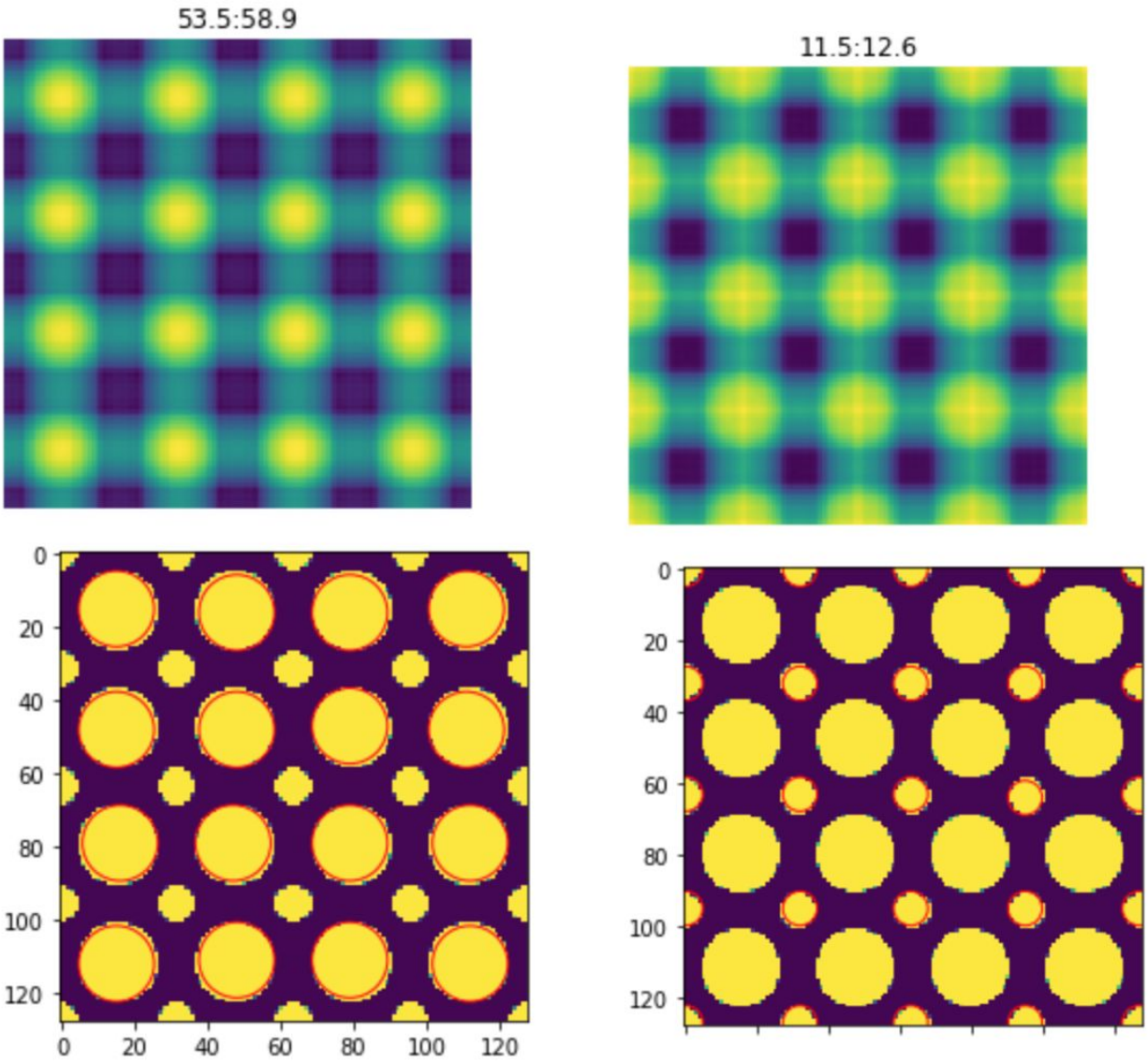
2) Heatmap for Harris Corner Detector



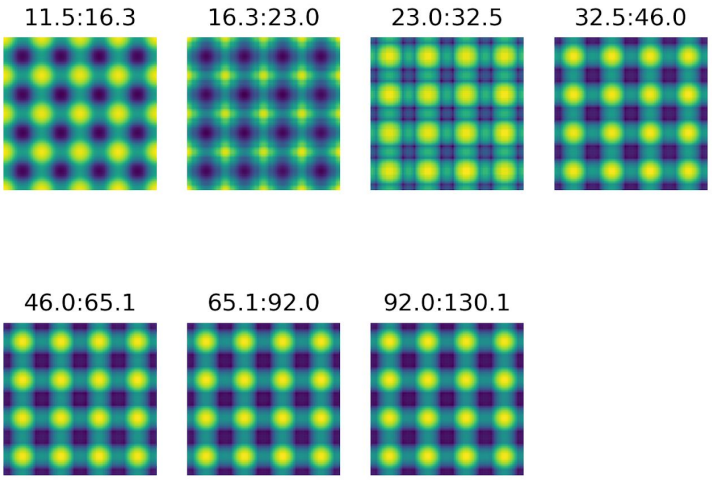
Blob Detection

1) Single-scale Blob Detection

   For the small circles, the sigma value used was **11.5 & 12.6 respectively** and for the big circles, the sigma value used was **53.5 and 58.5** respectively. Surprisingly, I didn't actually notice any false positives. Overall, for the large dots I observed 16 maxima and for the small dots, I observed the same number of maxima.



2) I can clearly see different maxima in the image. However, I did notice a difference in my results when I used the gaussian_filter function from scipy when compared to my own gaussian_filter function.



3) The best parameter values I found were 6 & 3 respectively for k_xy and k_s. However, I did have an issue where my large blobs weren't being located. Here are some of the images I generated. (Note the two circles on the right hand-side when the parameters were 9 & 1 respectively).