CS371 Digital Image Processing
Exercise 3
Image Filtering

Alexandros Angelakis
csd4334@csd.uoc.gr

November 27, 2022

1 Implementing my own 2D convolutions

For the first part of the assignment, we were called to implement our own functions for performing 2D convolutions on images. The first function computes the valid convolution with an arbitrary 2D filter, and the second the same-size convolution with a separable 2D filter, with padding type either zero-padding or pixel boundary value replication.
The convolution algorithm is pretty simple, we multiply each value of the image with each value of the kernel and then we sum, to get the value of the new, filtered image.
Let's take some filters to do the convolution with our example image below.

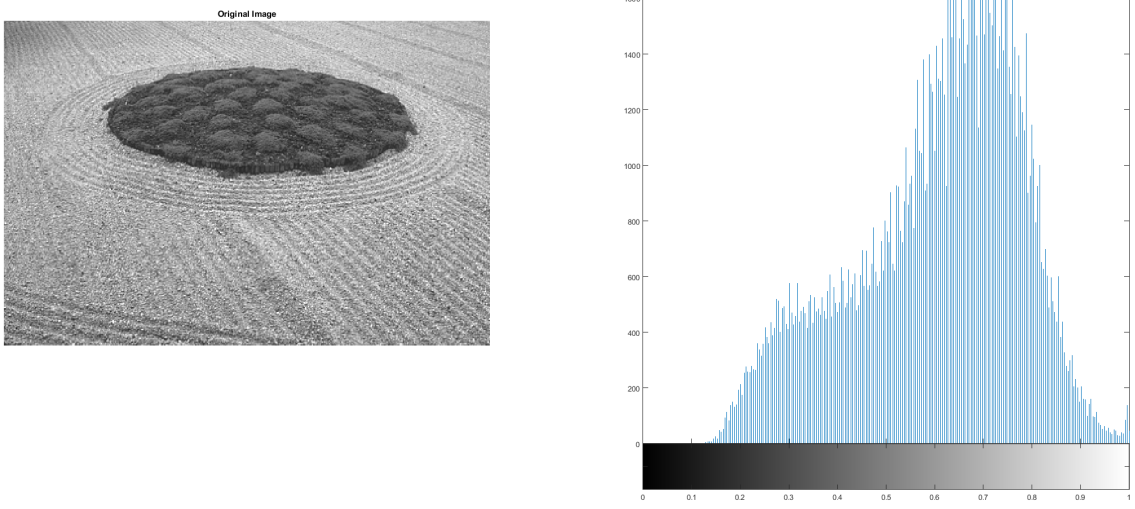
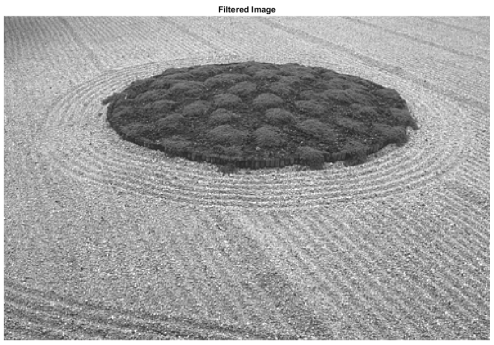


Figure 1: Original image with its histogram

Let's filter the image with the identity filter $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ using the first function. Since we do a valid convolution, we should see the same image, slightly smaller.



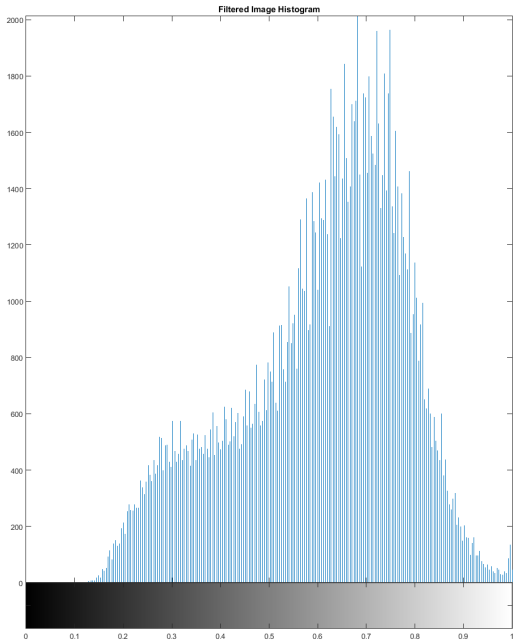
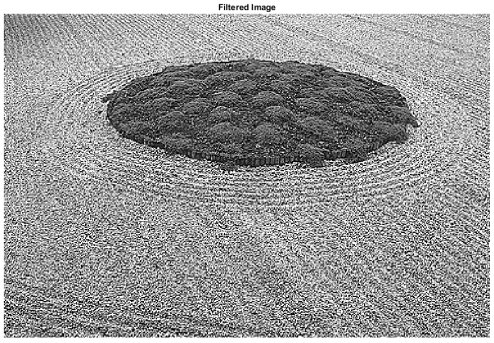


Figure 2: Filtered image with the identity filter and its histogram

If we look very closely, we can see the difference between those two images, it's like the filtered image is a zoomed version of the original one.

Let's filter the image with another filter now, the sharpen filter $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$



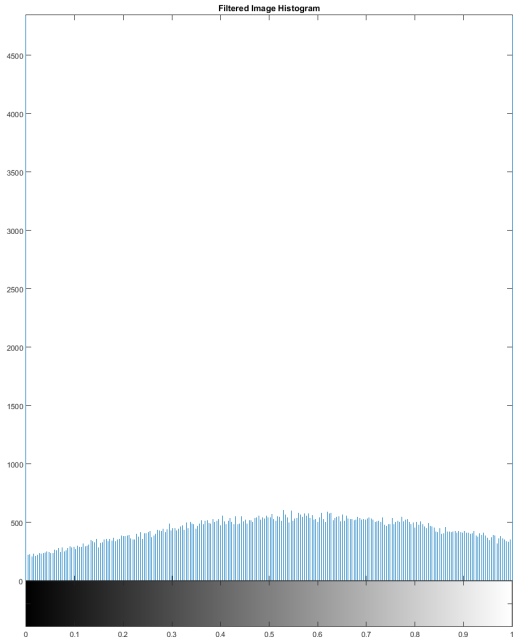


Figure 3: Filtered image with the sharpen filter and its histogram

As we can see, the filtered image is a lot more sharper. That means, we can easily see the details of the image. The histogram has a lot of differences between the original one. It is spreaded all over the x axis, it feels like we performed histogram equalization.
Since the second function uses the first function, but instead of a valid convolution it performs a same-size convolution using a separable kernel, the results will be almost the same with the above examples.

2 Building my own filters

2.1 Smoothing and Binarization

We tested the smoothing Gaussian Filter using three different values for the standard deviation parameter σ : 3,5,7. First of all, let's see how the smoothing filter affected the original image without doing the binarization, using the different sigmas. The padding-type we use here is **zero-padding**.
As we can see, by increasing the σ parameter, the smoothing is increasing more and more, making the image blurrier. The effect of the zero-padding padding type is pretty noticeable, the dark effect around the image is pretty clear, and it's becoming larger when we increase the σ. We'll see later if those dark spots in the image will still be in the binarized images.
My implementation and MATLAB's have the same output, so there are no differences.

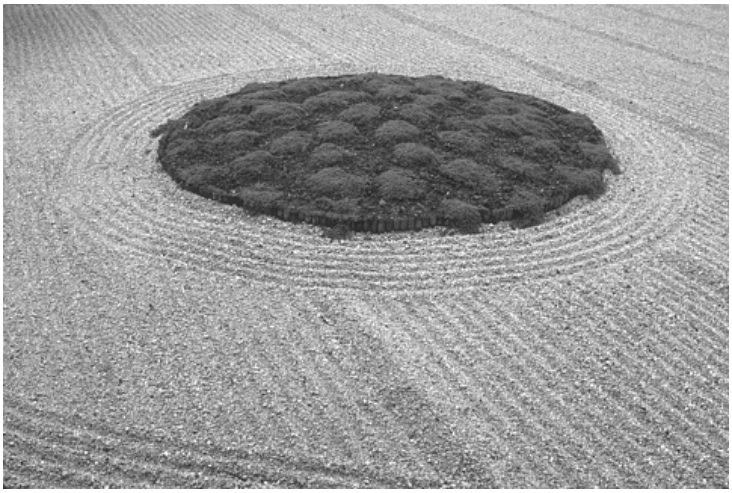


Figure 4: Original image

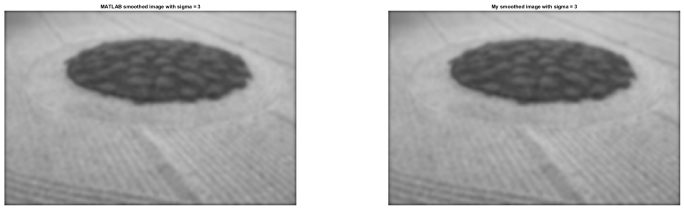


Figure 5: Smoothing Gaussian Filter with $\sigma = 3$

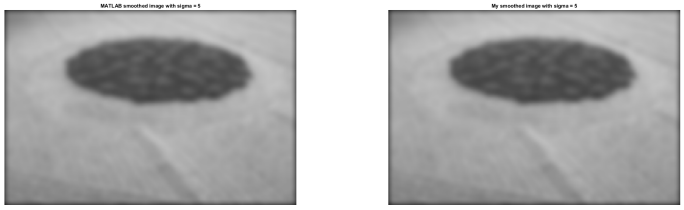


Figure 6: Smoothing Gaussian Filter with $\sigma = 5$

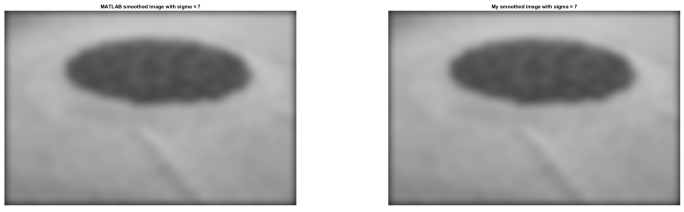


Figure 7: Smoothing Gaussian Filter with $\sigma = 7$

Let's take a look of how a different padding type will affect the smoothing of the image. We'll use the **pixel boundary value replication** as padding type. The padding type does not change the blurriness of the images, but it does change the dark spots around the image. If we look very closely we might see them, but there are not that noticeable. Thus, it'll be interesting to see the binarized images and compare them with the other padding type.

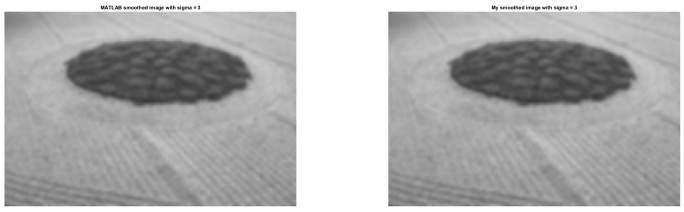


Figure 8: Smoothing Gaussian Filter with $\sigma = 3$

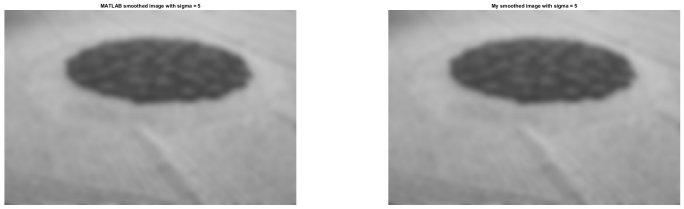


Figure 9: Smoothing Gaussian Filter with $\sigma = 5$

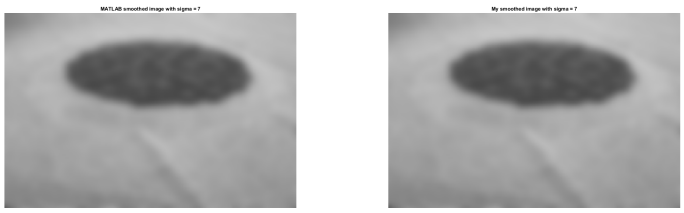


Figure 10: Smoothing Gaussian Filter with $\sigma = 7$

Now that we covered the smoothing part of the images, let's compare and visualize the binarized smoothed images. It is pretty clear that the circle in the middle of the image is the thing we want to highlight. The above images were filtered with padding type zero-padding. It confirms what we said before. The dark spots around the image are still here, very noticeable and very noisy. The circle of the image is not as highlighted as we wanted to be, especially when we increase the σ. But we can say that, the larger the σ is, the smoother the original image is. Thus, the circle of the middle of the image is more highlighted, when we binarize the images.
To compare the binarized original image with the smoothed ones, there are major differences. The serious one is the noise in the image, without smoothing the image, the grass on the ground is still noticeable. Thus, the noise without smoothing the image is pretty high. After smoothing the image tho, all those noises in the original binarized image is gone.

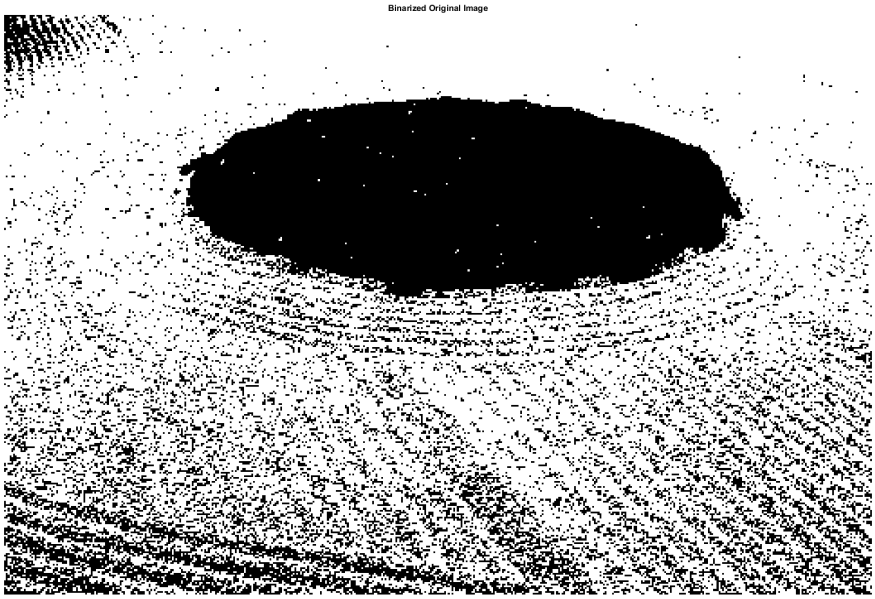


Figure 11: Original image

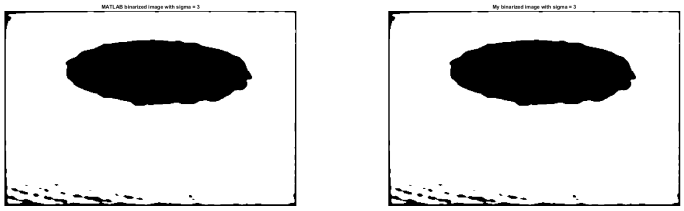


Figure 12: Binarized Image with $\sigma = 3$

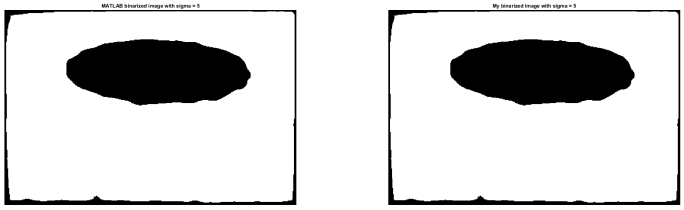


Figure 13: Binarized Image with $\sigma = 5$

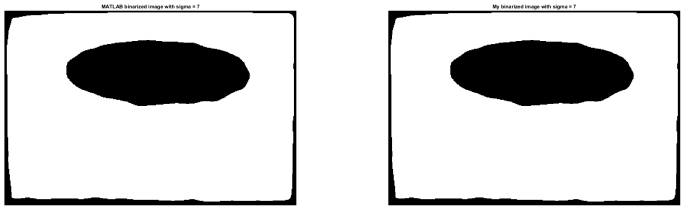


Figure 14: Binarized Image with $\sigma = 7$

Let's see how the pixel boundary value replication padding type works. This type of padding works better for smoothing the image. As we can see, the binarized images don't have those dark spots around the image anymore, and as we increase the σ, we get a pretty good highlight of the circle in the middle.
These images have nothing to do with the original binarized one, there is just a little bit of noise in the image for bigger sigmas, but for low sigmas, the noise of the grass is still noticeable.
All in all, binarizing an image without first smoothing it with a Gaussian filter has a lot of problems, the most significant is the noise. When we use a Gaussian filter for smoothing, the padding type is really important. As we saw before, by smoothing with zero-padding there is a lot of noise in the outer pixels of the binarized image, dark spots. In contrast, by smoothing with the pixel boundary value replication type, this noise in the outer pixels is gone, or almost gone, as we increase the σ. Therefore, the pixel boundary value replication padding type is doing better than the zero-padding.

Figure 15: Binarized Image with $\sigma = 3$

Figure 16: Binarized Image with $\sigma = 5$

Figure 17: Binarized Image with $\sigma = 7$

2.2 Local Standard Deviation and Binarization

In this section we are going to filter the below image with a local standard deviation filter, using a box averaging filter over a 5x5 neighborhood.

Figure 18: Original Image

First of all, let's see how the local standard deviation filter works on the above image, before binarization. As we can see from the filtered images, it feels like we do an edge detection on the original image, the windows, the bars and the balcony are mostly visible in the filtered image.

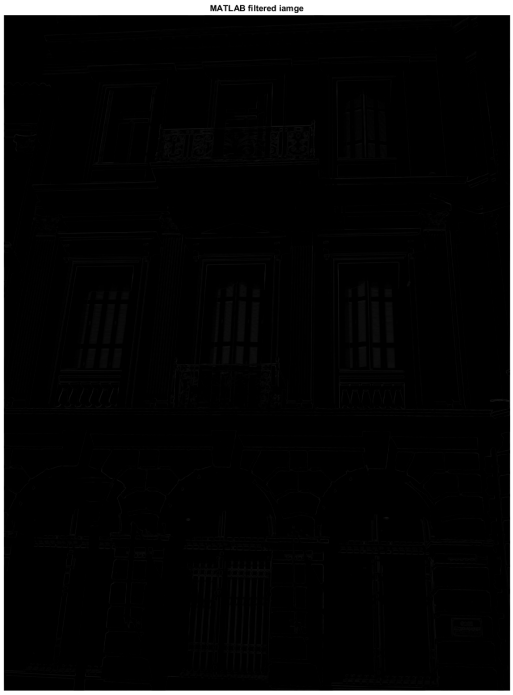


Figure 19: Local Standard Deviation filtered image

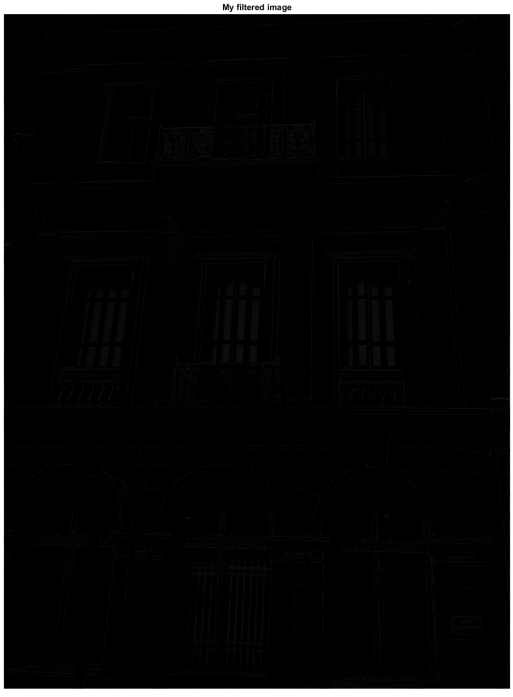


Figure 20: Local Standard Deviation filtered image

Let's take a look at the binarized images now. As we can see, the binarized original image has a lot of differences between the filtered binarized images. First of all, the highlighted parts of the original images is mainly the building unlike the filtered binarized images where the highlighted parts are the windows, the bars and the balconies, as we said before. The original image it's like the "reverse" image of the filtered ones, because where the original image has a white area, in the filtered one there is a dark area. This is not always the case because the bottom part of the image and specifically the three glasses behind the bars is the same.

Figure 21: Binarized Original Image

Figure 22: Local Standard Deviation binarized filtered image

Figure 23: Local Standard Deviation binarized filtered image

2.3 Laplacian and Binarization

In this section we are going to implement an approximation of a Laplacian of Gaussian filter, by taking the difference of two Gaussian filters G1, G2, with the standard deviation σ_1 and σ_2 respectively. We considered that $\sigma_1 = 1.28 \times \sigma_2$ and that σ_2 takes values : $1, \sqrt{2}, 2, 2\sqrt{2}$.
My implementation and MATLAB's have the same output, so there are no differences.
We know that by applying a Laplacian of Gaussian filter, we do an edge detection on the original image, so we expect to see in the filtered image all of the edges of the original image. First of all, let's take a look at the original image.

Figure 24: Original Image

As we can see in the image, there are a lot of vertical and horizontal edges, the letters are black and the background of the image is partially white. Since we apply a Gaussian filter in the original image first, we smoothen the image, so those vertical dark lines won't be visible in the binarized images, unlike the binarized original image.
Before looking at the binarized filtered images, let's first see how the Laplacian of Gaussian filters work on this image.
By increasing the sigmas of the two Gaussian Filters, we can see that the edge detection is becoming more and more visible. The number on the plate is clearly more visible as we increase the standard deviation of the Gaussians.

Figure 25: Laplacian of Gaussian Filter with G1(σ = 1.28×1) and G2(σ = 1)

Figure 26: Laplacian of Gaussian Filter with G1($\sigma = 1.28 \times \sqrt{2}$) and G2($\sigma = \sqrt{2}$)

Figure 27: Laplacian of Gaussian Filter with G1($\sigma = 1.28 \times 2$) and G2($\sigma = 2$)

Figure 28: Laplacian of Gaussian Filter with G1($\sigma = 1.28 \times 2\sqrt{2}$) and G2($\sigma = 2\sqrt{2}$)

Now, let's see the binarized images. By looking at the binarized original image, we can see that the vertical dark lines we talked about before are covering the number on the plate. That is because we didn't apply any smoothing filter on the image yet. Our purpose is to highlight the number on the plate, since it is the only very dark spot on the image. When we apply the Laplacian of Gaussian filter with a low standard deviation number, for example $\sigma = 1$, we can see that those vertical lines are not there. This happens because as we said before, we apply a smoothing Gaussian filter first and then we do the edge detection. All the letters and numbers on the plate are very good depicted, we can understand each and every letter/number. As we increase the σ, we can see that only the number on the plate is being highlighted more and more, the white border covering the plate and the letters that are not in the number are being less highlighted. Not only that, they are less comprehensible when we increase the σ. Lastly, the number on the plate is more and more filled with white colour as we increase the σ, meaning that this is what we want to highlight.

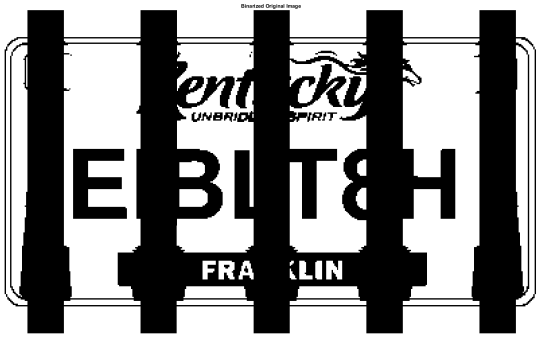


Figure 29: Binarized Original Image

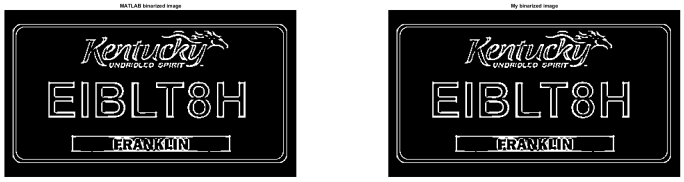


Figure 30: Laplacian of Gaussian Filter with G1($\sigma = 1.28\times1$) and G2($\sigma = 1$)

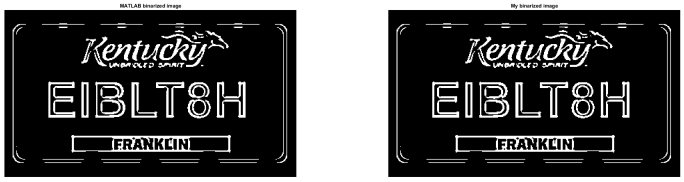


Figure 31: Laplacian of Gaussian Filter with G1($\sigma = 1.28 \times \sqrt{2}$) and G2($\sigma = \sqrt{2}$)

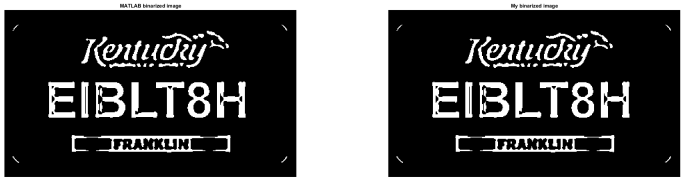


Figure 32: Laplacian of Gaussian Filter with G1($\sigma = 1.28{\times}2$) and G2($\sigma = 2$)

Figure 33: Laplacian of Gaussian Filter with G1($1.28{\times}2\sqrt{2}$) and G2($2\sqrt{2}$)