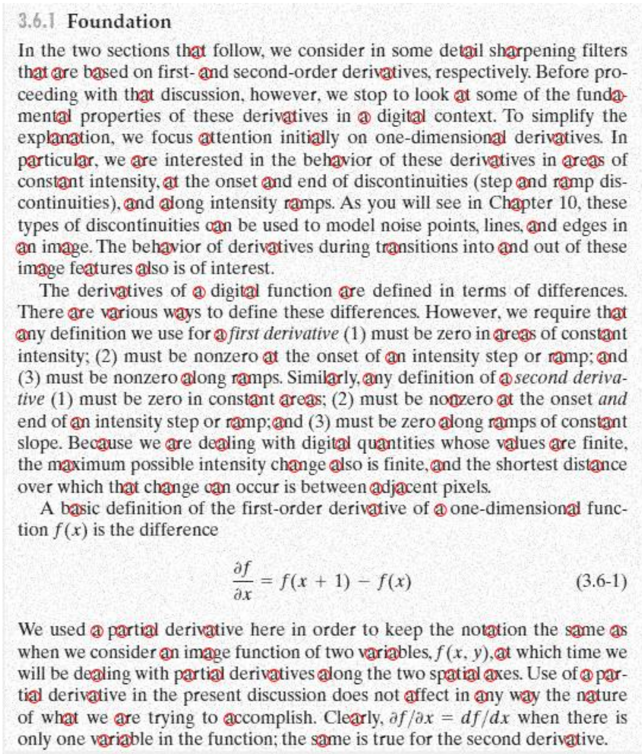
EE 475 HW#3 Sezen Perçin

Object recognition/detection by cross-correlation:



3.6.1 Foundation

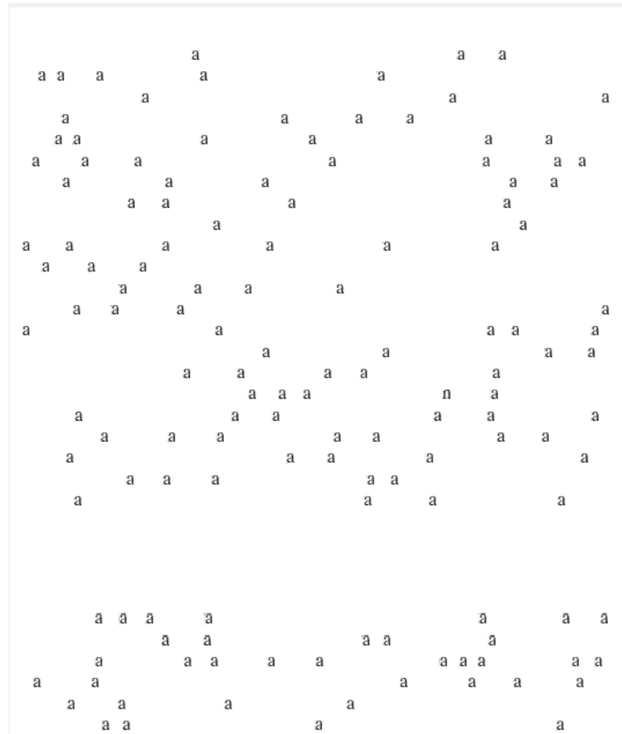
In the two sections that follow, we consider in some detail sharpening filters that are based on first- and second-order derivatives, respectively. Before proceeding with that discussion, however, we stop to look at some of the fundamental properties of these derivatives in a digital context. To simplify the explanation, we focus attention initially on one-dimensional derivatives. In particular, we are interested in the behavior of these derivatives in areas of constant intensity, at the onset and end of discontinuities (step and ramp discontinuities), and along intensity ramps. As you will see in Chapter 10, these types of discontinuities can be used to model noise points, lines, and edges in an image. The behavior of derivatives during transitions into and out of these image features also is of interest.

The derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, we require that any definition we use for a *first derivative* (1) must be zero in areas of constant intensity; (2) must be nonzero at the onset of an intensity step or ramp; and (3) must be nonzero along ramps. Similarly, any definition of a *second derivative* (1) must be zero in constant areas; (2) must be nonzero at the onset *and* end of an intensity step or ramp; and (3) must be zero along ramps of constant slope. Because we are dealing with digital quantities whose values are finite, the maximum possible intensity change also is finite, and the shortest distance over which that change can occur is between adjacent pixels.

A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{3.6-1}$$

We used a partial derivative here in order to keep the notation the same as when we consider an image function of two variables, $f(x, y)$, at which time we will be dealing with partial derivatives along the two spatial axes. Use of a partial derivative in the present discussion does not affect in any way the nature of what we are trying to accomplish. Clearly, $\partial f/\partial x = df/dx$ when there is only one variable in the function; the same is true for the second derivative.



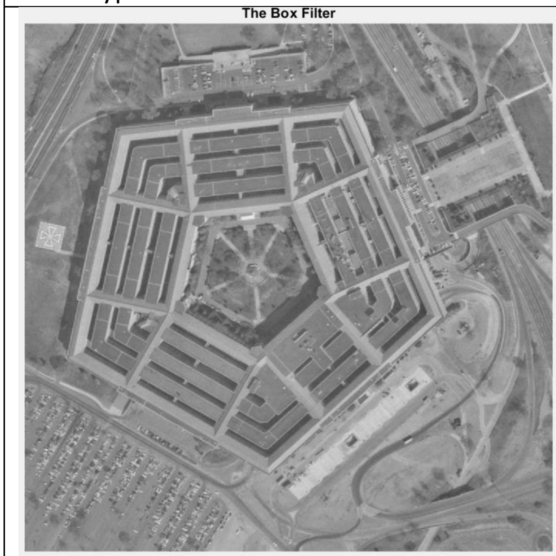
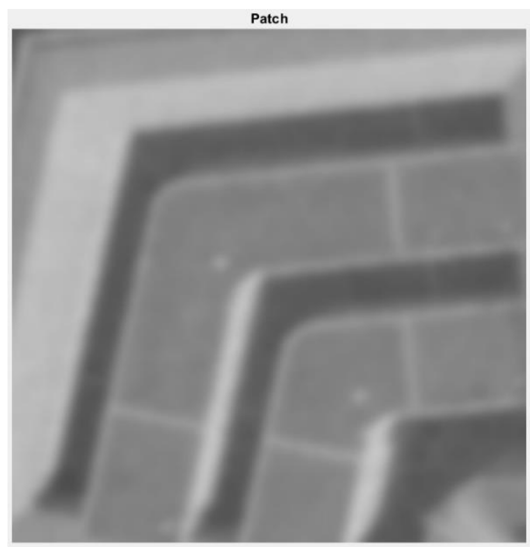
There are in total 141 lower case “a” letter in the text and 138 of them are found with the threshold of 0.752. Additional 1 “n” letter is captured as a false alarm therefore:

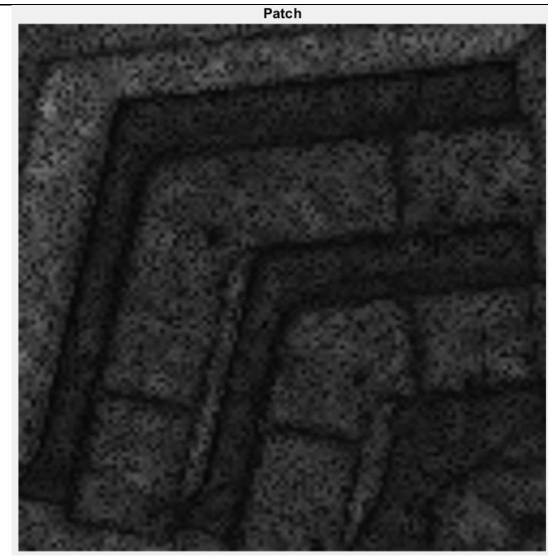
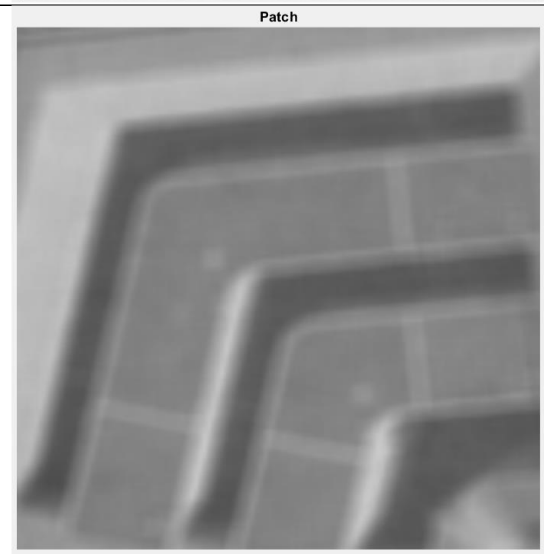
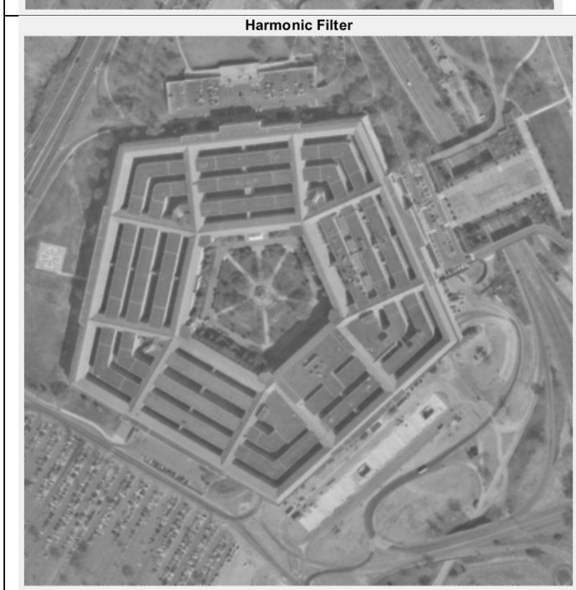
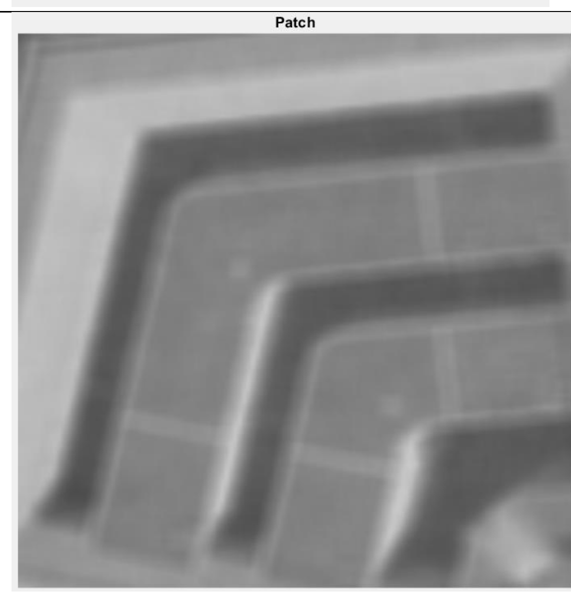
Comission Error: 1/139*100=%0.71

Omission Error:3/141*100=%2.13

Averaging Filters:

The differences among the filter results are more evident when they are examined through the MATLAB interface. Image with gaussian noise of variance 18.05 (as calculated from the formula) (I divided this value by 255*255 since the imnoise function has variance in range 0-1 but the image values are between 0-255):

Filter Type	Patch	SSIM
		0.6954
		0.6137

		0.1273
		0.5623
		0.5592

I implemented these filters by developing algorithms that runs the calculations of the filter formulas and applies it by looping over the pixels (or scanning with a frame, in this case: the mask). Box filter has the most SSIM value.

The box filter is a filter that takes the average of all the pixels in the mask. In this way all the pixels contribute to the target pixels equally.

Gaussian filter weighted more on the pixels of the mask that are closer to target pixel. It made sense to me since the relation of the target pixel is weaker with the far away pixels (their intensity and value). It is more possible that the target pixel is a component of a structure with the nearest pixels.

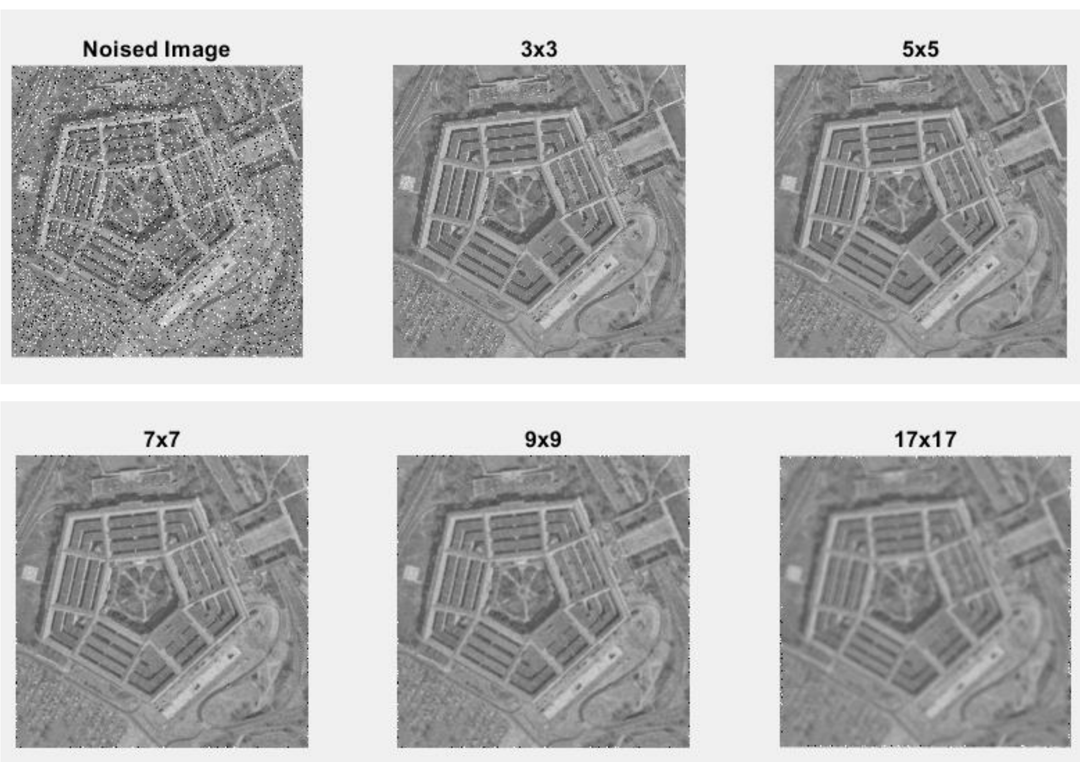
In geometric filter every pixel contributes to target pixel equally. Since every pixel has the operation of multiplication an increase or decrease of the pixel contributes more to the target value than box filter. For this reason, may the SSIM value of the geometric filter less than box filter.

Harmonic filter is similar to geometric filter but instead of multiplication, division operation is performed. Their SSIM values are similar.

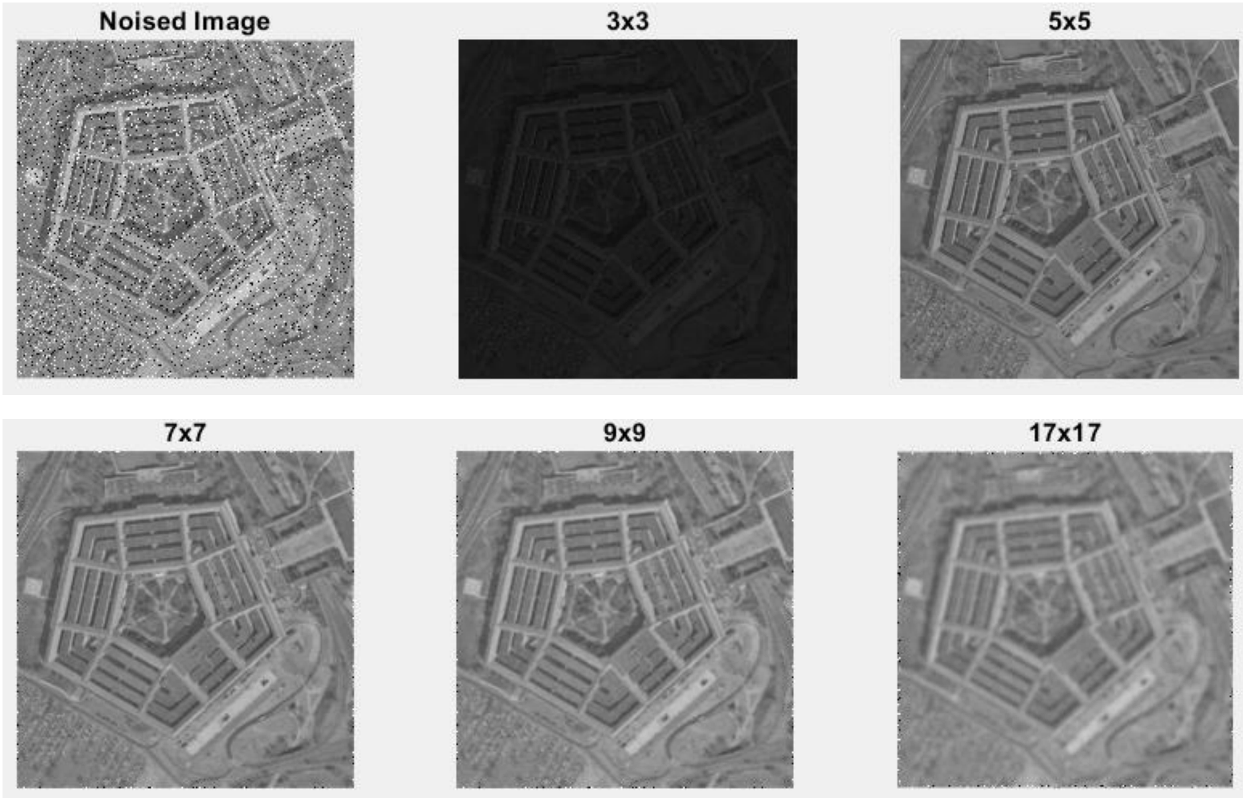
Inverse gradient filter in some way took the negative form of the image which enables us too examine whiter areas explicitly. But noise is still evident on the image. Some of the details that are present in the image before (e.g. cars) are no longer distinctively perceivable. I think this is the reason why its SSIM score is the lowest.

Rank order filtering:

For median filter:



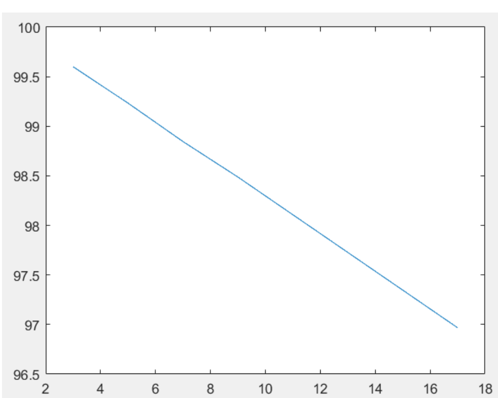
For alpha-trimmed-mean filter:

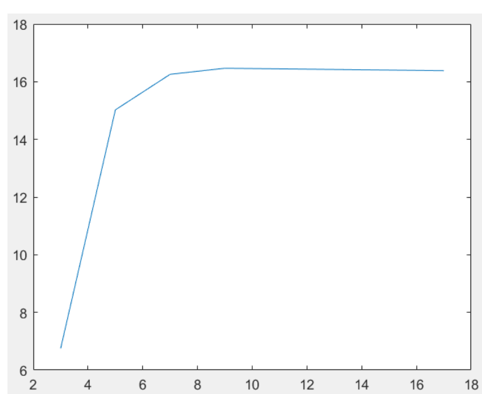


After using the original image with no salt and pepper noise I observed that with the mask size increasing, more pixels that are far away to the target pixel are used. Since this is an image that has many structures in short range of pixels, these pixels that are far away may carry infromation that belongs to different regions. Therefore, we observe that the image got blurred and became inaccurate hence it lost fidelity as the mask size increases. For median filter and alpha-trimmed filter respectively:

While experimenting with alpha I found the overall lowest PSNR value at alpha=8.

Window size- % of S&P noise removed || Window size - PSNR value (for the best alpha trim value)





Gaussian noise and salt-and-pepper noise:

1. For the first part I calculated the mean and standard deviation with the usual way by using the pixels in the window simultaneously. The result:

2. For the second part first I calculated the median of the pixels in the window and use it as an estimate of the mean robustly. Then I calculated the MAD (median absolute deviation) and use it along with the scale factor k=1.4826 to calculate standard deviation robustly. The result then was:


Problem 3.28 of G&W 4th Edition:

a)Since convolution with a Gaussian is a linear operation, convolution of the Gaussian filters will be again Gaussian.

b)New filter's standard deviation can be calculated as σ^2=1.5^2+2^2+4^2 then σ=4.72

c)The new filter will be of the size ceil(6*sigma). In this case ceil(4.72*6)= ceil(28.32)=29.