Performing Convolutions with Different Kernels

Bilal Ahmad

February 01, 2020

1 Introduction

I implemented mean, Gaussian, and median kernels to perform convolutions on two provided images, *NoisyImage1* and *NoisyImage2*. The purpose of the exercise was to explore how the different kernels perform on noisy images and how their parameters affect the outcomes of the convolutions. All code was written using MATLAB.

2 Mean Filtering

Mean filters work by replacing a pixel value with the average value of the surrounding pixels. I made two mean kernels, one a 3x3 and the other a 9x9. I used MATLAB's *conv2* function to perform the convolution on the images. The code used to implement the convolutions is shown in Listing 1. The results of the convolutions can be seen in Fig 1 and Fig 2.

Listing 1: MATLAB Code for Mean Filtering

```
% Make the kernels
lanKernel1 = ones(3,3)/9;
lanKernel2 = ones(9,9)/81;

% Convolve keeping size of the images
larK1_Image1 = conv2(lanNoisyImage1, lanKernel1, 'same');
larK2_Image1 = conv2(lanNoisyImage1, lanKernel2, 'same');
larK1_Image2 = conv2(lanNoisyImage2, lanKernel1, 'same');
larK2_Image2 = conv2(lanNoisyImage2, lanKernel2, 'same');
```

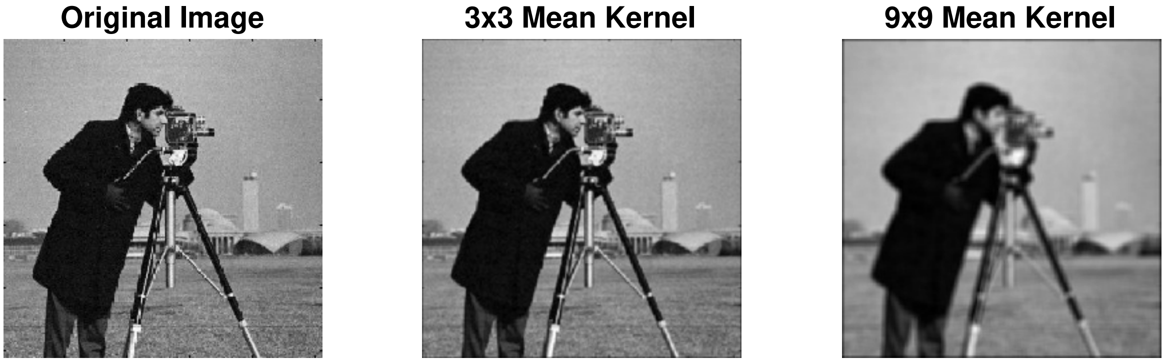


Figure 1: Applying Mean Filtering to *NoisyImage1*

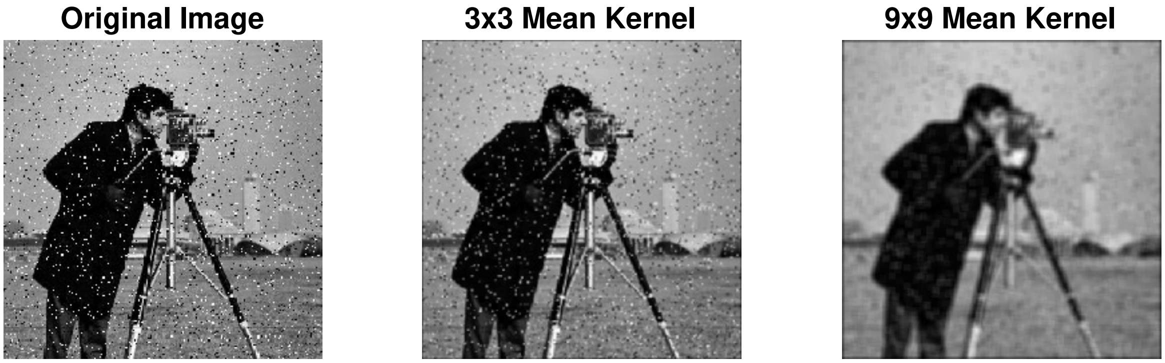


Figure 2: Applying Mean Filtering to *NoisyImage2*

For both images, the 3x3 kernel smoothed the image without reducing the edges too drastically. For the larger 9x9 kernels, the images have been blurred too much. This result was expected since a larger kernel size means that smoothing is occurring over a larger area for each pixel. Neither kernel sizes perform particularly well at eliminating the salt and pepper markings in *NoisyImage2*.

3 Gaussian Filtering

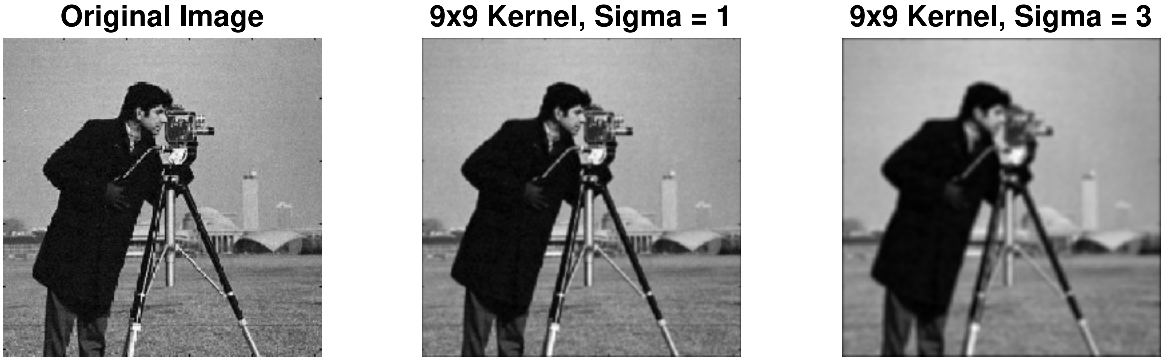
Gaussian kernels are weighted so that values closer to the center pixel have more weight than values further away. The weights of the values in the kernel are described by Eq 1. The sigma value in Eq 1 determines how wide the Gaussian curve is. Larger simga values will produce kernels whose values decrease less drastically the further away they are from the center pixel than they would with a smaller sigma value.

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{1}$$

For this assignment, I made two Gaussian kernels. They are both 9x9 kernels but one has a sigma value of 1 while the other has a sigma value of 3. I chose these sigma values because I wanted to see how skinnier Gaussian curves compare to thicker curves. With a 9x9 kernel, sigma values of 1 and 3 create enough discrepancy to show the relationship between sigma size and smoothing. To implement the Gaussian kernel, I did a loop over each item in a 9x9 matrix and applied Eq 1 to the value. I did not use the constant factors in Eq 1 because I normalized the kernel after looping through the kernel. The MATLAB implementation of constructing the kernels can be seen in Listing 2. After building the kernels, I used the *conv2* function similarly to how it was used for mean filtering. The results of the convolutions can be seen in Fig 3 and Fig 4.

Listing 2: MATLAB Code for Gaussian Kernel

```
lrSum = 0.0;
for i = 1:9
    for j = 1:9
        % for 9x9 kernel the center is (5,5)
        sq_dist = (i-5)^2 + (j-5)^2;
        larKernel1(i,j) = exp(-1 * (sq_dist)/ (2 * lnSigma1^2));
        lrSum = lrSum + larKernel1(i,j);
    end
end
larKernel1 = larKernel1/lrSum;
```



Original Image

9x9 Kernel, Sigma = 1

9x9 Kernel, Sigma = 3

Figure 3: Applying Gaussian Filtering to *NoisyImage1*

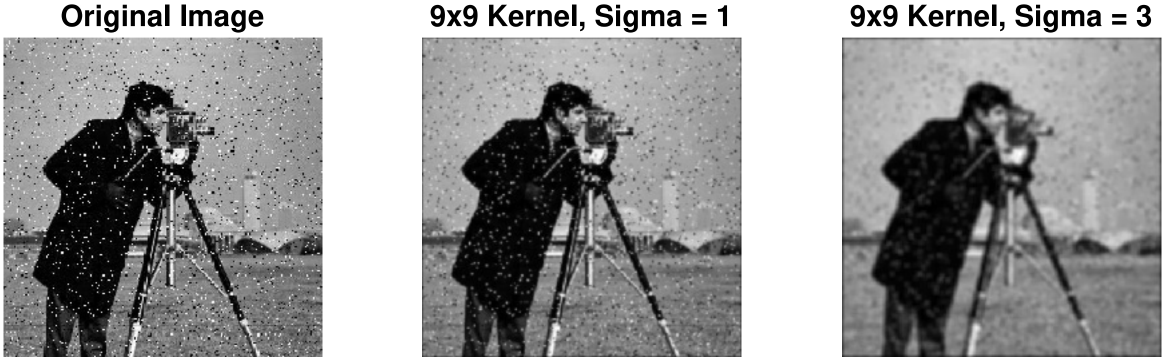


Figure 4: Applying Gaussian Filtering to *NoisyImage2*

Gaussian filtering with the sigma values used performed similarly well to mean filtering. The smaller sigma of 1 did a better job of smoothing the image without sacrificing too much edge clarity. This is expected since larger sigma values place less emphasis on closer pixel values and begin to resemble mean filters of larger kernels. Gaussian kernels are better low pass filters than mean kernels and upon close inspection you can see that there are fewer artifacts with the images produced from Gaussian filtering with a sigma value of 1 than the images produced from mean filtering with a 3x3 kernel. Again, the salt and pepper markings are still prevalent in *NoisyImage2* after smoothing has been applied.

4 Median Filtering

Median filtering works by replacing the center pixel with the median value of all the pixels in the kernel. MATLAB has a built in *medfilt2* function for median filtering that I used. I performed the filtering using a 3x3 and 9x9 kernel. The code for implementing the convolutions is shown in Listing 3 and the results of the convolutions are shown in Fig 5 and Fig 6.

Listing 3: MATLAB Code for Median Filtering

```
% Use medfilt2 to perform convolution
larK1_Image1 = medfilt2(lanNoisyImage1, [3 3]);
larK2_Image1 = medfilt2(lanNoisyImage1, [9 9]);
larK1_Image2 = medfilt2(lanNoisyImage2, [3 3]);
larK2_Image2 = medfilt2(lanNoisyImage2, [9 9]);
```

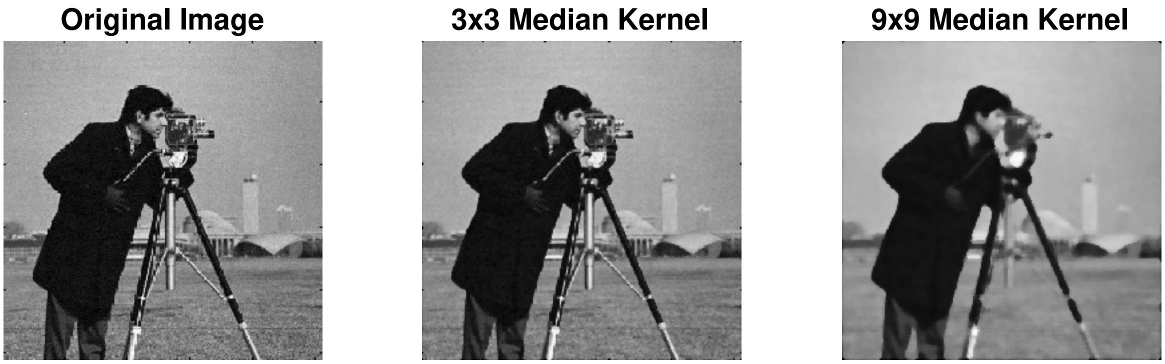


Figure 5: Applying Median Filtering to *NoisyImage1*

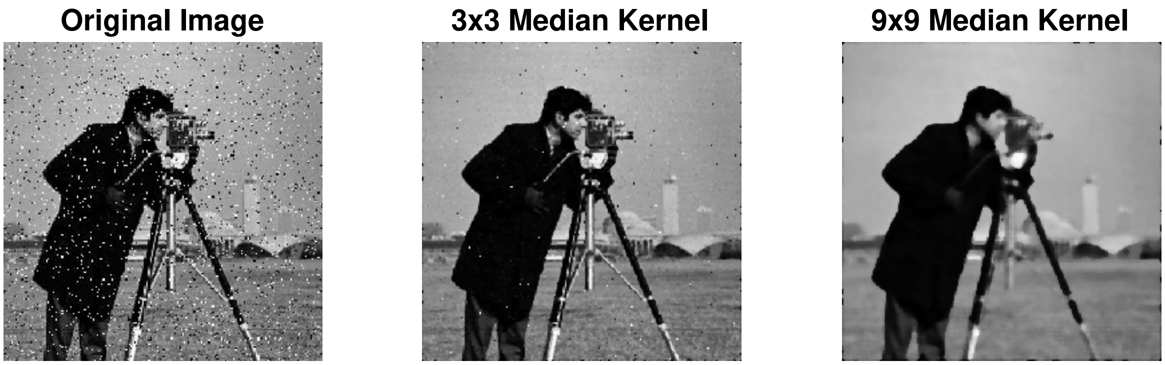


Figure 6: Applying Median Filtering to *NoisyImage2*

Median filtering the smaller 3x3 kernel produced better results than the 9x9 kernel. The larger kernel produced images that were smoother but almost all edges were lost. This was expected since the larger kernel size means that edges will not be as localized and they will get washed out in the convolution.

5 Comparing Kernels

For *NoisyImage1*, the 9x9 Gaussian kernel with a sigma value of 1 performs better than the other kernels. As mentioned earlier, the Gaussian kernel is a better low pass filter than the mean kernel so it produces fewer artifacts. In addition, the Gaussian kernel produced a smoother image than the 3x3 median kernel especially along the edges of the man's coat. For *NoisyImage2*, the median 3x3 kernel clearly outperformed the other kernels as it was able to eliminate most of the salt and pepper markings while also smoothing the image.