

CMSC498D Project 3

by: Jacob Lin, William Lin

Introduction

In this project, we will test and explore a few different types of denoising algorithms. We will be testing the algorithms on the following image:



For each algorithm, we will be tuning its parameters to get an idea of how these parameters affect the denoising process.

PSNR

This section was fairly straightforward, we created a function that will be used in the future to get a general measure of how good our denoising algorithms will be. The PSNR function takes in two parameters, the original image and the denoised image.

Add Noise

For this section, we had to consult some numpy documentation and we came across `np.random.normal` which was exactly what we are looking for to add some noise in our original image to test out denoising algorithms.

Our noisy image is shown below:

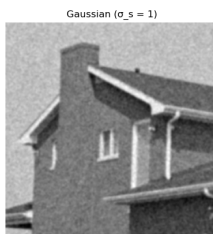


PSNR: 22.10

Gaussian Filtering

For this section, the most difficult part was understanding and implementing $i_{denoised}(x)$. We first defined the ranges of the 25 x 25 window and then separately calculated the numerator and denominator. Instead of having an x value that has the coordinates of a pixel, we instead had an i, j value which allowed for some easier implementations.

As for the Gaussian filter itself, we were able to use the given Gaussian weight function and passed it into $i_{denoised}(x)$ to be used. We first used the actual pixel values in the implementation of the function but quickly realized after looking at the Bilateral weight function that this weight function relies on the spatial distances of a particular pixel and its neighboring pixels. The results of each σ_s value is shown below:



PSNR: 29.64



PSNR: 22.49

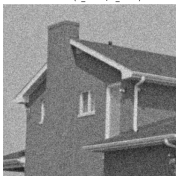
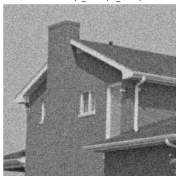
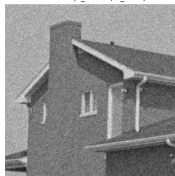
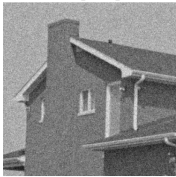
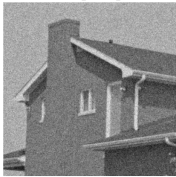
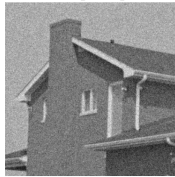

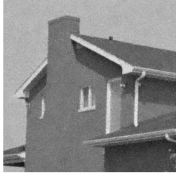









PSNR: 20.50

As we can see here, Gaussian denoising does in fact remove the noise but also causes the images to be blurred which simply just creates another offputting quality. Also notice that a σ_s value of 1 seemed to result in the highest PSNR value out of the three.

Bilateral Filtering

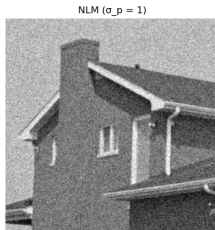
Bilateral filtering is very similar to Gaussian filtering. Since we already implemented a general denoising function, we only had to implement the bilateral weight function. The resulting images are given below:

$\sigma_i \backslash \sigma_s$	1	5	25
1	 <p>Bilateral ($\sigma_s = 1, \sigma_i = 1$)</p>	 <p>Bilateral ($\sigma_s = 5, \sigma_i = 1$)</p>	 <p>Bilateral ($\sigma_s = 25, \sigma_i = 1$)</p>
	PSNR: 22.14	PSNR: 22.26	PSNR: 25.99
5	 <p>Bilateral ($\sigma_s = 1, \sigma_i = 5$)</p>	 <p>Bilateral ($\sigma_s = 5, \sigma_i = 5$)</p>	 <p>Bilateral ($\sigma_s = 25, \sigma_i = 5$)</p>
	PSNR: 29.27	PSNR: 29.67	PSNR: 22.15
25	 <p>Bilateral ($\sigma_s = 1, \sigma_i = 25$)</p>	 <p>Bilateral ($\sigma_s = 5, \sigma_i = 25$)</p>	 <p>Bilateral ($\sigma_s = 25, \sigma_i = 25$)</p>
	PSNR: 22.49	PSNR: 27.25	PSNR: 27.88
50	 <p>Bilateral ($\sigma_s = 1, \sigma_i = 50$)</p>	 <p>Bilateral ($\sigma_s = 5, \sigma_i = 50$)</p>	 <p>Bilateral ($\sigma_s = 25, \sigma_i = 50$)</p>
	PSNR: 22.60	PSNR: 22.16	PSNR: 22.51
500	 <p>Bilateral ($\sigma_s = 1, \sigma_i = 500$)</p>	 <p>Bilateral ($\sigma_s = 5, \sigma_i = 500$)</p>	 <p>Bilateral ($\sigma_s = 25, \sigma_i = 500$)</p>
	PSNR: 26.75	PSNR: 26.17	PSNR: 20.60

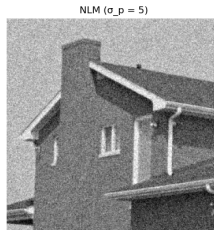
Notice that a σ_s and σ_i value of 25 yielded the highest PSNR value.

Non-Local-Means

After looking at the documentation for `cv2.fastNlMeansDenoising`, we called the function to denoise our image with a few different sigma values and got the following results:



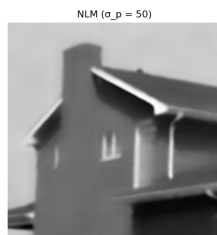
PSNR: 28.70



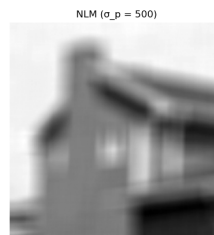
PSNR: 28.70



PSNR: 33.12



PSNR: 31.72

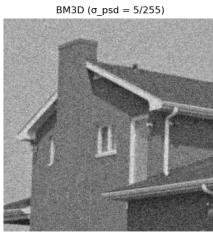


PSNR: 30.51

With this algorithm, we have gotten the highest PSNR values so far with a σ_p value of 25 being the highest. We noticed that we have fairly gotten rid of most of the noise in the image. However, many of the details in the edges have been removed.

Block Matching 3D Collaborative Filtering

Similar to above, we looked at the documentation for `bm3d.bm3d` and passed in the noisy image and varying sigma values. We got the following results:



PSNR: 22.33



PSNR: 33.62



PSNR: 31.80

We see that a σ_{psd} value of 20/255 yielded the highest PSNR of 33.62. However, we noticed that a lot of the details in the brick is lacking in this denoised images which is understandable since it can be mistaken as noise.