ECE 590 Image Processing Final Project

Ting Chen - tc233 - 20190504

1 Introduction

In this project I'll mainly focus on 2 potential improvements for non-local means algorithms. Since it does not involve any model training or clustering, this project will not deal with extra datasets.

1.1 Non-local means with equalized block

As we discussed in class, non-local means (NLM) is an image denoising algorithm. Instead of averaging a pixel with its neighbors, NLM calculates the means of all the pixels/blocks in the image, and result in better preserved details, especially for repetitive patterns. However, the method introduced in class use a Gaussian weigh function, which might lead to large difference for similar patterns with different brightness in terms of grayscale images, as Webster pointed out. This will be discussed in next section in detail.

For this topic, I want to find an alternative weighting function that put greater weight for similar patterns regardless of their brightness. For example, 1-D arrays/blocks from [1, 2, 3, 2, 1] to [127, 128, 129, 128, 127] will have smaller distance than [1, 2, 3, 4, 5]. By making this change, I'm assuming that the image might have repetitive patterns, but difference in exposure among blocks are considerably huge. I'm not sure about the outcomes for this assumption, and I'll discuss with several examples and test cases. Also, if there is still time, I will discuss how does the window size and standard deviation parameters affect the result metrics. All the experiments for this topic will be conducted with grayscale images. Without further declaration, each example image will be 64*64 pixels, the window/block size will be 5*5 pixels and the sigma value will be 5e4. All test images are either taken from our class, from internet or from the original paper [1].

1.2 Applying non-local means to color images (planned but not conducted)

Another topic is to apply NLM to a color image. In class we only did NLM to grayscale images. When doing NLM to color images, it's not explicitly discussed which color representation (e.g. RGB, HSV, YC_bC_r, etc.) we should use. For this topic I will first try to apply NLM on all 3 channels of an image and see if there are chromatic aberration and other artifacts, and then try to find better ways of doing NLM for color images.

1.3 How to run the code

The code will be run in a "one-click" fashion. After you run the main test script, it will pop up several figures concerning the results. Also, all the result figures will be provided in this report. The original code for NLM is taken from our class, and changes are based on that. Please change the original noise-free image in the script to test with different images.

2 Methods

In this section I will discuss the basic definitions and expected approaches for the experiments.

2.1 Basic definitions and assumptions for NLM image denoising

In class, the image noise model is described as follow.

$$I(x,y) = S(x,y) + N(x,y)$$

Where S is the original signal, and N is a 2-D Gaussian noise.

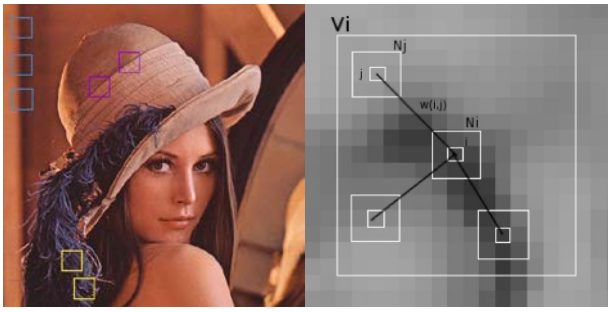

FIGURE NLM SIMILAR PATTERNS EXAMPLES. SOURCE: DSVISION.GITHUB.IO; [2]

NLM assumes the image contains self-repetitive patterns. For example, in the left image above, blocks with same color are similar to each other, which will result in higher weight in the NLM algorithm. In the right image, we can have a general feeling about how this algorithm works: for example, if there is a sharp edge or line with one pixel in width, then all the pixels on this line will have greater weight for each other.

2.2 Original weighting function for non-local means algorithm

The original weighting function f is described as follow.

$$f(p,q) = e^{-\frac{|B(q)-B(p)|^2}{h^2}}$$

Where p and q are two pixels in the image. $B(p)$ is the window/block surrounding center pixel p with a given size, and h acts as a degree of filtering, which controls the decay of the weighting function.

2.3 Potential change to the weighting function

2.3.1 Brightness average

The first thought is to set the brightness of all other blocks that compare to current block to the same level. That is to say, to shift the vector till they have the same mean value. By doing this, the brightness difference between blocks is mitigated, and the trend or the pattern of the block is taken more into consideration. The shift process can be described as follow. If we want to shift vector v2 to align with vector v1.

$$v_2 = v_2 + d_{shift} = v_2 + mean(v_1) - mean(v_2)$$

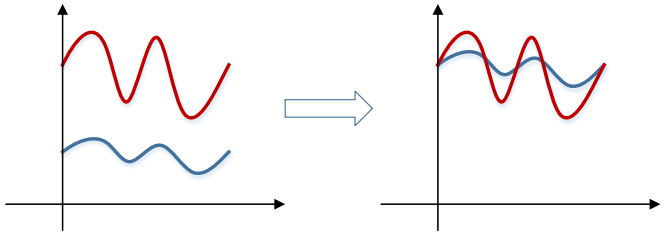

FIGURE EXAMPLE OF "EQUALIZED" SIGNALS

As the figure shown above, both 1-D signals have similar trends along x axis, whereas the blue signal is lower in average comparing to the red signal. After moving the blue signal up, the difference or the distance between each point is smaller, and similar patterns could contribute more to the overall weight.

One problem of this change to the weighting function is that we cannot directly apply the weight to the original image, since we calculated the distance for pixels that were shifted in "brightness". One potential solution is to keep a copy of the shifted value of each center pixel of the block, and apply the weighting function to that value. In next section, I'll further elaborate on this approach.

2.3.2 Brightness normalization

Another method is to align the vectors according to their peaks, or to match the minimum and maximum values. By doing this, both of the signals will have the same range in brightness, and similar patterns with different brightness level will also benefit it.

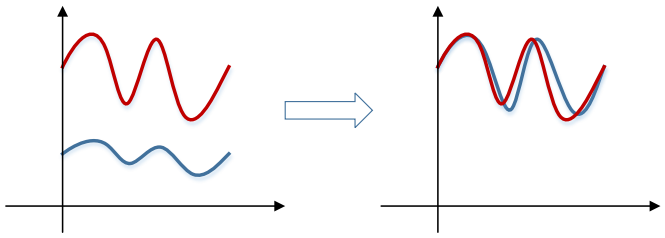

FIGURE EXAMPLE OF "NORMALIZED" SIGNALS

For vector v1 and v2, this assumes for each element i in vector v_2,

$$v_{2i}^{'} = \left(v_{2i} - \min\left(v_2\right)\right)\frac{\max\left(v_1\right) - \min\left(v_1\right)}{\max\left(v_2\right) - \min\left(v_2\right)} + \min\left(v_1\right)$$

This method could also lead to problems. One is that it could make less similar blocks have higher weight. For the original weighting function, if the blocks are significantly different, it will give the compared-to blocks small weights. But with this method, even a not similar compared-to block can also have a rather high weight, which is undesirable. Also, this method need to keep a copy of the "normalized" center pixel value to calculate the final denoised image.

2.3.3 Euclidean distance

For the original algorithm, I found the distance between two blocks is the square of their Euclidean distance. I'd like to change the distance to square root of it. That is to say, change the weighting function to

$$f(p,q) = e^{-\frac{|B(q)-B(p)|}{h^2}}$$

This might result in a worse denoising effect, since e^{-x} decays slower than e^{-x^2}, and will make weight values less different from each other.

2.4 Naïve approach to apply non-local means filter to a color image

Basically, the idea of applying NLM to color image is to apply NLM to each channel of that image and combines the channels to get the denoised image. However, this could be problematic considering different types of noise and images. For example, if the noise is grayscale only, or black and white "salt and pepper", this method won't work so well for it; also, if the image are biased in channels, say it does not contains much red channel signal, then applying NLM to that channel might even worsen the quality of it.

2.5 Metrics for results

2.5.1 RMSE

Root Mean Square Error (RMSE) is the most common metric for the difference between images that we used in class. RMSE is defined as follow.

$$RMSE = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[I_1(i,j) - I_2(i,j)\right]^2}$$

For comparing the results, I'll calculate the RMSE 5 times for both the original method and the improved, and average them to get the final results

2.5.2 PSNR

Peak signal-to-noise ratio (PSNR) is also a commonly seen metric for measuring how much is an image affected by noise. PSNR is defined as follow and the result is dB in unit.

$$PSNR(I_0, I) = 20\log_{10}\frac{MAX_I}{RMSE(I_0, I)}$$

PSNR will also be calculated for each case as a reference.

3 Results and discussion

In this section, I will discuss the results for the 3 potential changes to NLM denoising algorithm.

3.1 Weight distribution comparison

The weight distribution of a certain pixel can reflect how the other pixels contribute to the denoising step. For example, if the weight distribution have the highest weight exactly for those similar/repetitive patterns, then the denoising result will be expectedly great, since no artifacts will be introduced. Therefore, I investigate the weight for different methods first to have a general perception of their potential performances. In this section the test images are listed below.

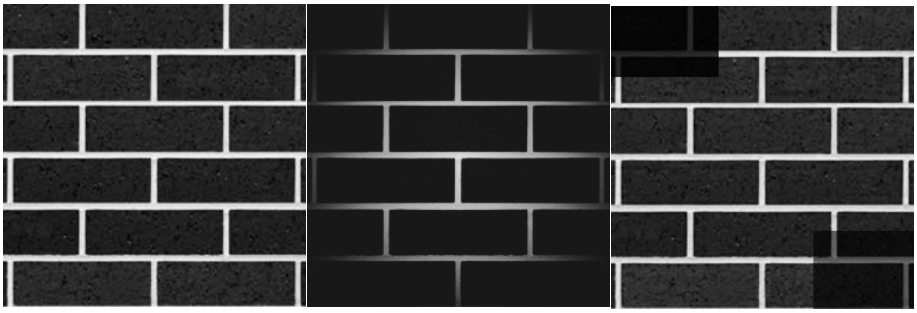

FIGURE THE TEST IMAGES. FROM LEFT TO RIGHT ARE TA, TB, TC

The test image TA is the original image, which is a wall with bricks. TB is TA with a radial gradient shadow added to it. TC is TA with brightness of upper left and lower right corners adjusted.

3.1.1 Original weighting function

The original weight distribution of NLM should be similar to figure 2 in the original paper. The highest weighted pixel should appear at the center of the most similar compared-to block, and the pixels with higher value should have similar brightness to the original pixel, due to the trait of an exponential function.

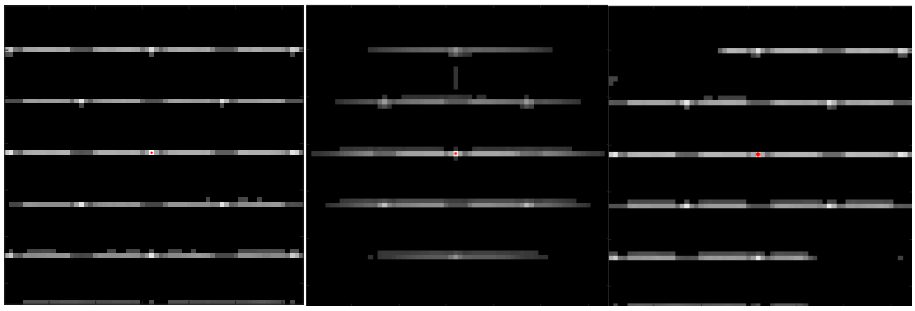

FIGURE EXAMPLES FOR ORIGINAL WEIGHTING FUNCTION. FROM LEFT TO RIGHT ARE GRAPH A, B, C

Above is an example of the weight distribution for the original function. The target pixel is at the center of the image, as the red dot marked in A, B and C. Most of the highest weight pixels are at the intersection of “T” shaped patterns among the blocks, and the horizontal white lines also have considerably high weights. Although the patterns are exactly the same for TA, TB and TC, the original weighting function cannot give higher weight for the corners in TB and TC since their brightness/distance are much different than the center pixel.

3.1.2 The average method

According to my assumption, the average method should perform similarly to the original weighting function for the original image, and should be slightly better for the shadowed image TB and TC.

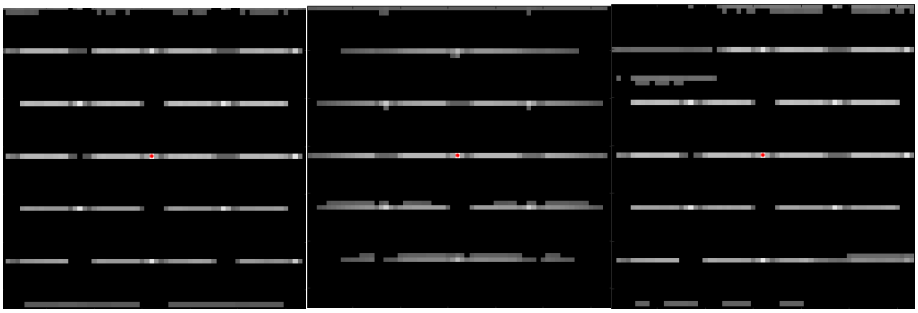

FIGURE EXAMPLES FOR AVERAGE METHOD. FROM LEFT TO RIGHT ARE GRAPH A, B, C

As it can be seen in the graphs, in addition to a similar result for TA, the result B and C are slightly better than that of original weighting function. However, for the normalization method, it also introduced some artifacts between the rows of bricks. By aligning the average value of compared-to block, all the pixels in it will change in brightness. In my case, since TC is poorly contrasted in corners within a rectangle region, the edge of that rectangle has a similar effect as the white lines between each row of bricks in the original image TA, which causes some artifacts. Because the rectangle region is rather darker than original image, the original weighting function won't give those pixels high weights and would just lead to the result like figure C in 3.1.1.

3.1.3 The normalization method

This method performs pretty similar to the average method, but is worse in some aspects.

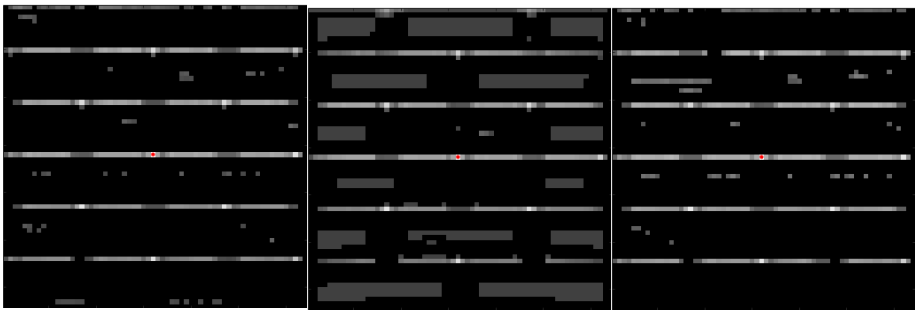

FIGURE EXAMPLES FOR NORMALIZATION METHOD. FROM LEFT TO RIGHT ARE GRAPH A, B, C

For TA, this method also has a similar output to its peers. And for TC, it yields a better result than the original weighting function. For TB, this method does not perform well, and introduced lots of artifacts to the original image. Since this method will normalize a compared-to block according to its target block, the problem described in section 2.3.2 could happen.

3.1.4 The square root method

After experiments, I found that this method have same behaviors as the original method, only different in the values of weight. Therefore, I will not elaborate on this method.

3.2 Denoising metrics

	Metric	Original	Average	Normalization
Cameraman	RMSE	9.64	9.84	34.45
	PSNR	28.34	28.15	17.29
Lena	RMSE	10.11	11.37	28.87
	PSNR	27.32	26.30	18.21
Brick brightness 2	RMSE	8.27	8.16	44.98
	PSNR	28.80	28.92	14.10

TABLE COMPARISON OF METRICS BETWEEN METHODS

Above is the table for all the experiments and their metrics. All the sigma for weighting function is picked as 3e3.

As we can see from the table, the average method performs really closed to the original weighting function, whereas the normalization method failed all the cases. This result is surprising to me, as I expected the average method to perform slightly better than the original method at first.

For the average method, further rigorous conduction should take place. By shifting the mean value of a block, not only the signal got shifted, but also the noise. As my analysis for weight distribution was conducted for noise free images, it might not be the same for noisy images. I'll inspect this result later.

For the normalization method, it makes the weight distribution more balanced, which means that each pixel depends more on every other pixel. And that lead to a more blurry and obscure image.

3.3 Denoising results discussion

3.3.1 The shifted value problem proposed in 2.3.1

In section 2.3.1 I proposed a problem for average method. That is, if we multiply the weight matrix directly to the original image, the shifted value of the center pixel in a compared-to block can be significantly different with its original value. This could lead to some unusual effects in the denoised output.

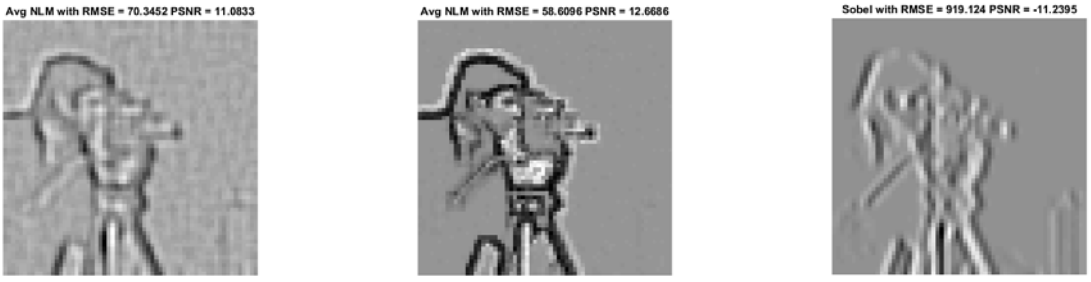

FIGURE STRANGE EFFECT ON THE SHARP EDGE OF CAMERAMAN IMAGE. LEFT TO RIGHT - SIGMA = 5E4, SIGMA = 3E3, SOBEL

When applying average method to the cameraman test image, I got this result, which remind me of the Sobel operator/filter for edge detection. Therefore, I plotted the original noise-free cameraman image through Sobel operator in both directions. Clearly, although the average method displayed sort of the edge detection effect on the denoised images when not properly set, it does not share the same mechanism with Sobel operator. Generally speaking, Sobel operator is an approximation for the derivatives in both directions. And average method on original image has this sort of edge effect because of the sharp contrast on edges. If we take all the edges blocks into

consideration, and if a block happens to have most pixels black/white, then it would contribute to the overall weighted image and make the reconstructed edge black/white. In the cameraman case, since the man's coat is very dark, then the edge around it is mainly black. And for Lena, the edge is either dark or white, since both type of edges exists in the original image.

3.3.2 How to pick sigma for weighting function

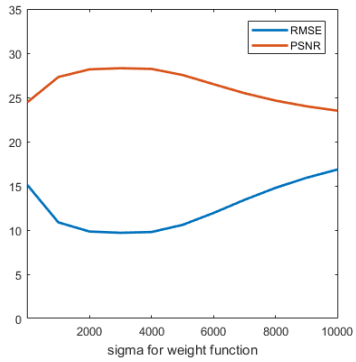

FIGURE SIGMA FOR CAMERAMAN

Instead of following the original sigma value that I pick previously, I wrote a script to find out the best sigma for each image. Despite the images all have the same dimensions, the best sigma to minimize their RMSE can vary due to different. After testing several images, I decided to use sigma = 3e3 in the results above. Also, as we can see in the figure, both metrics share a similar trend, only different in the sign. Therefore, for the discussion, I mainly talk about the RMSE for a denoised image.

3.3.3 Visual comparison between denoised result

All the result denoised images can be found along with my code.

FIGURE AVERAGE METHOD VS ORIGINAL WEIGHTING FUNCTION VS NORMALIZATION METHOD FOR BRICK TEST CASE

Generally speaking, the results from average method and original weighting function are really similar to each other, and I cannot distinguish them. The result from normalization method totally failed for all test cases, with patch-like mosaic all around the denoised images.

4 Conclusion

This experiment yielded some unexpected results in contrast to my previous expectation. Overall, the average method works, but does not have a noticeable advantage on the original weighting function. And the normalization methods does not work. More rigorous conduction of average method should be done to find the conditions for it to prevail.

This project as well as the class was interesting and inspiring. During the semester I learned a lot of methods to deal with 2-D signals, especially low frequency images. Hopefully I'll have more chance to dig into signal processing in the future.

5 References

[1]. Buades, A., Coll, B., & Morel, J. M. (2005, June). A non-local algorithm for image denoising. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 2, pp. 60-65). IEEE.

[2]. Nguyen, T. A., Nakib, A., & Nguyen, H. N. (2016). Medical image denoising via optimal implementation of non-local means on hybrid parallel architecture. Computer methods and programs in biomedicine, 129, 29-39.

[3]. An approach to Non-Local-Means denoising: http://dsvision.github.io/an-approach-to-non-local-means-denoising.html

[4]. Wikipedia - Non-local means: https://en.wikipedia.org/wiki/Non-local_means

[5]. Wikipedia - Peak signal-to-noise ratio: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[6]. Wikipedia - Sobel operator: https://en.wikipedia.org/wiki/Sobel_operator