Image Denoise by Total Variation Minimization, Bilateral Filter and Wavelet Denoising Filter

This paper investigates using total variation minimization(TV) to denoise image and compares it with Bilateral filter(BT) and Wavelet denoising filter(WT). We introduce the algorithm for TV in details and successfully denoise the target image using it. Compared with other two methods, when we try more denoising(narrow the stop criterion) for TV ,we get a better denoised image with less noise comparatively.

Heng Zhou

Hongyang Zheng

Prerna Kaul

Xinyi Ye

I. INTRODUCTION

The problem of image denoising has long been a heated topic and people are always seeking for more efficient methods. Image noise is random variation of brightness or color information in images, and is usually an aspect of electronic noise. A typical model of image noise is Gaussian - additive, independent at each pixel, and independent of the signal intensity, caused primarily by Johnson–Nyquist noise(thermal noise), including that which comes from the reset noise of capacitors ("kTC noise").

There are numerous ways of denoising an image. A few commonly known are - one is using deep neural networks put forward by University of Science and Technology of China. They combined sparse coding and deep networks pre-trained with denoising auto-encoder (DA) to solve low-level vision problems. Moreover, they also used this technique for image inpainting[1]. Another approach is of Non Local Means, which addresses the preservation of structure in a digital image[2]. Other than image denoising, image inpainting is also a popular topic. The difference between the two is that, inpainting focuses more on reconstructing the lost or deteriorated parts of images. An approach replicating the basic techniques used by professional restorators, involves translating the manual inpainting concepts into a mathematical and algorithmic language[3]. The technique is then to select regions to be restored, the algorithm automatically fills-in these regions with information surrounding them.

However, in this paper, we adopted three different methods to conduct image denoising. These methods are: Total Variation Minimization, Bilateral Filter, and Wavelet Denoising Filter. There are two kinds of total variation minimization technique, one is performed using split-Bregman optimization and the other is Linearized Bregman algorithm. We picked the first one to perform denoising, trying to find a image with less total variation under the constraint of the original image. The datasets are downloaded from the Internet, including one 225 by 225 JPG and a 256 by 256 square PNG images. They are all selected and specifically appropriate for image denoising. The results are the successfully denoised images and the performance of denoised results is determined by MSE and estimated standard deviation of Gaussian Noise. MSE gives the similarity between denoised image and original image and Gaussian Noise shows standard deviation of the noise in the image. From MSE results, Wavelet filter gives the most similar denoised image compared to original when it is with parameter *convert2ycbcr*. While Gaussian Noise showed that image after Wavelet filter without the parameter has least noise.

II. DATASET

The dataset has two parts: the first image (dataset1) contains a 225 x 225 pixels image [Fig1] and the second image (dataset2) contains a 256 x 256 pixels [Fig 2] image. These were first converted to a numpy array of pixels using cv2 package of python with the function ‘imread’ and then were explored.

The exploratory data analysis (EDA) was carried out to get an insight into the image dataset. For this analysis, histograms were plotted [Fig3 & Fig4], using the calcHist function of cv2 package, showing the the frequency of pixels in the images corresponding to the pixel values. This gave a general knowledge about image pixels orientation.

Since image is a high dimensional data, a projection of the images using principal component analysis (PCA) was done in order to reduce the dimensionality of the data set consisting of the pixels correlated with each other (either heavily or lightly), while retaining the variation present in the image, up to the maximum extent. The images were first reshaped using the ‘reshape’ function and then PCA for both the images was plotted using PCA() from sklearn.decomposition package, till the highest variation was

retained.[Fig 5 & Fig 6]. The final results of the projection are as follows:

Dataset 1 – We get the final PCA result with 99.2% variation retained from the original image with a compression ratio of 18.51%.

Dataset 2 - We get the final PCA result with 99.2% variation retained from the original image with a compression ratio of 16.27%.

III. TECHNIQUES

The problem we are trying to solve is to remove or decrease the noise in our transferred dataset. We added random noise for dataset1 and Gaussian noise for dataset2. Gaussian noise is evenly distributed over the signal. This means that each pixel in the noisy image is the sum of the true pixel value and a random Gaussian distributed noise value [4].

For this project, we used three techniques, they are:

1. Total Variation Denoising – It is a process is an effective filtering method for recovering piecewise-constant signals thereby reducing the noise. It is based on the principle that signals with excessive and possibly spurious detail have high total variation. According to this principle, reducing the total variation of the image subject to it being a close match to the original image, removes unwanted detail whilst preserving important details such as edges. Unlike a conventional low-pass filter, TV denoising is defined in terms of an optimization problem. The output of the TV denoising 'filter' is obtained by minimizing a particular cost function.

2. Bilateral Filter – It is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. The basic idea behind this algorithm is that two pixels can be close to one another, that is, occupy nearby spatial location, or they can be similar to one another, that is, have nearby values, possibly in a perceptually meaningful fashion. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

The bilateral filter is defined as[5][6]

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

where the normalization term

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

ensures that the filter preserves image energy and

I^{filtered} is the filtered image;

I is the original input image to be filtered;

x are the coordinates of the current pixel to be filtered;

Ω is the window centered in;

f_r is the range kernel for smoothing differences in intensities (this function can be a Gaussian function);

g_s is the spatial kernel for smoothing differences in coordinates (this function can be a Gaussian function).

As mentioned above, the weight W_p is assigned using the spatial closeness and the intensity difference.[6] Consider a pixel located at (i, j) that needs to be denoised in image using its neighbouring pixels and one of its neighbouring pixels is located at (k, l). Then, the weight assigned for pixel (k, l) to denoise the pixel (i, j) is given by

$$w(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}\right),$$

where σ_d and σ_r are smoothing parameters, and I(i, j) and I(k, l) are the intensity of pixels (i, j) and (k, l) respectively.

After calculating the weights, normalize them:

$$I_D(i, j) = \frac{\sum_{k,l} I(k, l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)},$$

where I_D is the denoised intensity of pixel (i, j).

3. Wavelet Denoising Filter - The basic idea behind Wavelet denoising, or Wavelet thresholding, is that the Wavelet transform leads to a sparse representation for many real-world signals and images. What this means is that the Wavelet transform concentrates signal and image features in a few large-magnitude Wavelet coefficients. Wavelet coefficients which are small in value are typically noise and can be "shrunk" or removed without affecting the signal or image quality. Because Wavelets localize features in the data to different scales, the important signal or image features can be preserved while removing noise. After the coefficients have been threshold the data is reconstructed using the inverse Wavelet transform.

Suppose we measure a noisy signal $x = s + v$. Assume s has a sparse representation in a certain wavelet bases, and $v \sim \mathcal{N}(0, \sigma^2 I)$

So $y = W^T x = W^T s + W^T v = p + z$.

Most elements in p are 0 or close to 0, and $z \sim \mathcal{N}(0, \sigma^2 I)$

Since W is orthogonal, the estimation problem amounts to recovery of a signal in independent and identically distributed Gaussian noise. As p is sparse, one method is to apply a Gaussian mixture model for p.

Assume a prior $p \sim a\mathcal{N}(0, \sigma_1^2) + (1-a)\mathcal{N}(0, \sigma_2^2)$, σ_1^2 is the variance of "significant" coefficients, and σ_2^2 is the variance of "insignificant" coefficients.

Then $\tilde{p} = E(p/y) = \tau(y)y$, $\tau(y)$ is called the shrinkage factor, which depends on the prior variances σ_1^2 and σ_2^2. The effect of the shrinkage factor is that small coefficients are set early to 0, and large coefficients are unaltered.

Small coefficients are mostly noises, and large coefficients contain actual signal.

At last, apply the inverse wavelet transform to obtain $\tilde{s} = W\tilde{p}$

V. ANALYSIS

First, we used total variation minimization to denoise dataset1. Total variation minimization (TV) is one of the most popular algorithms that have been used for image denoising. The dataset here is a noisy image with the size of 256 by 256 pixels. The grayscale information of the image is converted into an n*n by 1 array, which makes a 1D problem.

The objective function for 1D regularized problem is[7]:

$$\min_u ||u||_{BV} + \frac{\mu}{2}||u - f||_2^2$$

In order to solve this, we need to split the L1 and L2 parts. Therefore, we introduce a new variable:

$$d = \phi(u)$$

And we wish to solve:

$$\underset{u,x}{argmin}||d||_1 + H(u) \text{ such that } d = \phi(u)$$

Then we add the L2 penalty term to the problem:

$$\underset{u,x}{argmin}||d||_1 + H(u) + \frac{\lambda}{2}||d - \phi(u)||^2$$

We now define the Bregman Distance of this convex function as:

$$D_E^p(u, d, u^k, d^k) = E(u, d) - < p_u^k, u - u^k > + < p_d^k, d - d^k >$$

$$\text{,where } E(u, d) = ||d||_1 + H(u)$$

Hence we recursively solve:

$$(u^{k+1}, d^{k+1}) = arg \min_{u,d} D_E^p(u, d, u^k, d^k) + \frac{\lambda}{2}||d - \phi(u)||_2^2)$$

It is carried out by 3 steps in our program:

1. $u^{k+1} = \underset{u}{argmin} H(u) + \frac{\lambda}{2}||d - \phi(u) - b^k||_2^2$
2. $d^{k+1} = \underset{d}{argmin}|d|_1 + \frac{\lambda}{2}||d - \phi(u) - b^k||_2^2$
3. $b^{k+1} = b^k + \phi(u^{k+1}) - d^{k+1}$

Step 2 can be efficiently solved by $d^{k+1} = shrink(\phi(u^{k+1} + b^k, \frac{1}{\lambda}))$, and b is the Bregman parameter.

The program has three functions: shrinkage operator, sparse discrete difference matrix generation and Bergman iteration:

1. shrink(x,l): The shrink operator in the program, where x is the numpy array which is to be thresholded and returns a numpy array the same shape as x. A threshold parameter controls the number of jumps and the quality of the estimation which is sparse in a sense that TV reconstruction leads to piecewise constant with a small number of jumps. In this program, it is determined by the ratio of the sparsity term and fidelity term.
2. sparse_discrete_diff(n): n is for n by n grid. This function returns a 2*n*(n-1) by n*n matrix.

3. TV_min:(res, x, lam=0.1, gam=10, iter=1000): This is where Bregman iteration is carried out. res is the resolution of square image, x is the grayscale image as a numpy array, lam is the sparsity term, gam is the fidelity term, and iter is the number of iteration. There is no stopping criterion in this function because the number of iteration is determined by us. It returns an n*n by 1 array which will later be rearranged as n by n grayscale array and be displayed with plt.imshow().

In the whole process including inputting datasets, conducting the core algorithm and printing out the results, the following packages have involved:

1. numpy. It is broadly used throughout the program to perform basic calculations. For example, using np.abs() to get absolute values, np.array() conversion to array, and np.zero() to obtain zero values.
2. scipy.sparse. This is the 2D sparse matrix package for numeric data and mostly used in the sparse_discrete_diff function to create sparse discrete difference matrics. We used sparse.diags() and sparse.identity() to build diagonal and identity sparse matrices, and sparse.vstack() to stack the matrices row-wise.
3. scipy.sparse.linalg. It is used for solving the sparse linear system Ax=b, where b may be a vector or a matrix. In this program, it is used in the function TV_min to solve for φ.
4. matplotlib.pyplot. The most common and most used package for graphs, and like in most cases, we used it in our program to plot the result from arrays with plt.imshow(), and also setting titles, hiding axes, and saving figures.
5. PIL. It is used to open the .png.

After investigating the application of total variation minimization on image denoising, we then employed other two approaches: Bilateral filter and Wavelet denoising filter as well as total variation to denoise a new image (dataset2) and compared the denoising results. From the previous part, we know that different from total variation, Bilateral filter replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels,, and Wavelet denoising filter can remove noise with preserving major features in the image. Our dataset2 is a gray 256 by 256 picture, which is added random Gaussian-distributed additive noise and then converted into a 256 by 256 by 3 numpy array.

Next, we changed the value of parameter(s) in each method to see whether denoising process perform better or not. For total variation method, denoising increases with the increase in weight or smaller eps (stop criterion), so we first found the optimal weight (discussed later) and then changed *eps* from 0.02 to 0.01. For bilateral filter method, a larger value in *sigma_color* results in averaging of pixels with larger radiometric differences. Since the image has be converted into float64, the standard deviation is in the range [0,1]. Therefore, we changed *sigma_color* from 0.05 to 0.1. For Wavelet, we let *convert2ycbcr* equal true. This parameter and multichannel together be true can do the Wavelet denoising in the YCbCr color space instead of the RGB color space. This typically results in better performance for RGB images. Although our image is gray, we want to try and see how it works[8].

For this analysis, we mainly used a python package called skimage, which is useful for image processing in python, cv2, and matplotlib.pyplot. We import the following parts:

1. skimage.restoration, which includes denoise_tv_bregman (denoise image using total variation and return an image), denoise_bilateral (denoise image using bilateral and return an image), denoise_Wavelet (denoise image using Wavelet and return an image) and estimate_sigma (calculate standard deviation of Gaussian noise in the image)
2. skimage.util, which includes random_noise (add random Gaussian noise on the image)

3. skimage import img_as_float (convert uint8 into float64 data type)
4. using cv2.imread to read the image into a 256 by 256 by 3 array
5. using matplotlib.pyplot to plot the images

We used two metrics to measure the quality of images after denoising. The first one is the standard deviation of Gaussian noise for each image, which can be considered to indicate the level of noise remained in the image. The bigger the standard deviation is, the bigger the variation of noise is and the noisier the image is. The second metric is the mean squared error (MSE)[9], which compares the similarity between a denoised image and the original image. The mean squared error equation is:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_{(i,j)} - K_{(i,j)}]^2$$

In our code, I and K represent pixel intensities for the original image and denoised image; m and n represent the number of pixels in each image. This method calculates the difference of pixel intensities between the two images and then sums up the squared differences. Next, it divides this sum by the total number of pixels of each image and then we got MSE. The smaller the MSE is, the greater similarity is between denoised image and the original image. By looking at MSE, we can see how similar the two images are.

VI. RESULTS AND DISCUSSION

The result of TV algorithm consists of two parts: the images and the values of error and total variation norm[10]. Error is the L2 distance of the new array from the original one, TV is the total variation norm of the new array. In a total of 1000 iterations, the error and total variation norm is calculated for every 100 iterations, which is shown in Fig.8. From it we can see that the error has been stable around 79.64 since the 300th iteration after decreasing, while the total variation norm is increasing and is not stable around 1590.69 until the 600th iteration. Therefore, we concluded that the largest possible difference of the L2 distance between the original image and denoised image is 79.64, while the largest possible total variation norm is 1590.69.

During the in-depth analysis, we want to both compare the denoising effect of different methods and the effect of same method under different conditions. We know that the smaller the weight is, the more denoising total variation does, but it is at the expense of less similarity to the original image. We firstly need to find an optimal weight for total variation, so we can fix the weight at the best condition and let the eps as the only parameter we changed. As a result, we used MSE as a metric to find the weight that has the smallest MSE score (most similar to the original) and then investigates the effect of changing eps on decreasing noise. We first tried weight from 1 to 10 and got the minimum value when weight is around 3. Then we tried weight from 2.5 to 4.0, taking each step as 0.1, and found the optimal weight is still 3. Therefore, for the following analysis for total variation, we fixed *weight* = 3. Fig.9 shows the change tendency between weight and MSE score.

At the beginning, we compared the results between different denoising methods under fixed parameters. Fig.10 is the results after denoising the noisy image using total variation (TV) with *eps*=0.02, Bilateral (BT) with *sigma_color* = 0.05, Wavelet (WT) in RGB color space. After simple looking at these results, we can see that the total variation and Wavelet denoising seemly gave better restorations. After calculating the two metrics, which is shown in Table.1, we can see that total variation gave us the minimum MSE, which indicated that the denoised image by total variation is more similar to the original

one. However, Wavelet has the minimum standard deviation, which means it has smaller level of noise than the denoised image from total variation. Wavelet denoising has smaller level of noise at the cost of losing similarities.

From the MSE of Table.1, we can see that when we narrowed the stop criterion for TV, the denoised image is more similar to the original one (smaller MSE), and at the same time getting less noise variance. For BT, after decreasing *sigma_color*, we found that both MSE and noise variance decreased, which indicated that BT performed better. However, when we set *convert2ycbcr* as TRUE, more denoised image had more similarities with the original one, while its noise variance increased. There might be a tradeoff between similarity and noise for using WT method: the image with high similarity is more likely to have more noise. This is similar to the discovery found by Mohsen Ghazel, George H. Freeman, and Edward R. Vrscay. They implemented different levels of resolution and found that a low resolution results in smoothing most of the noise at the expense of blurriness of edges and blockiness artifacts. On the other hand, a higher resolution results in a greater reconstruction of the noise, while preserving the high-frequency content (edges) of the image [11].

Observing 6 process, they all successfully denoised the image and restored it to a great extent. All of them had smaller MSE and much less noise than the noisy image had. Among these 3 methods and 2 levels, the Wavelet with *convert2ycbcr* is the method that most likely to restore the noisy one to original image and Wavelet without *convert2ycbcr* has the least noise. From this prospective, maybe Wavelet is a more efficient approach to denoise image than using total variation minimization. Nevertheless, TV and BT are much more stable since when changing the parameter in the direction of decreasing noise, they also keep a good similarity level.

APPENDIX

Fig.1 :- Dataset 1

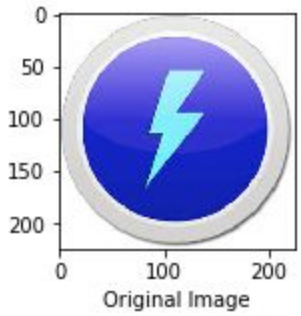


Fig.2 :- Dataset2

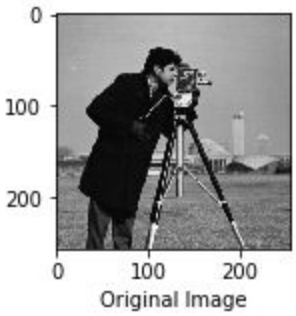


Fig.3 :- Histogram of dataset1 showing the frequency of each pixel contained in the image

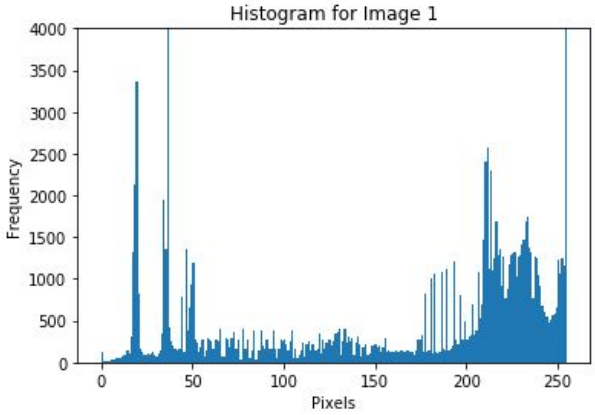


Fig.4 :- Histogram of dataset2 showing the frequency of each pixel contained in the image

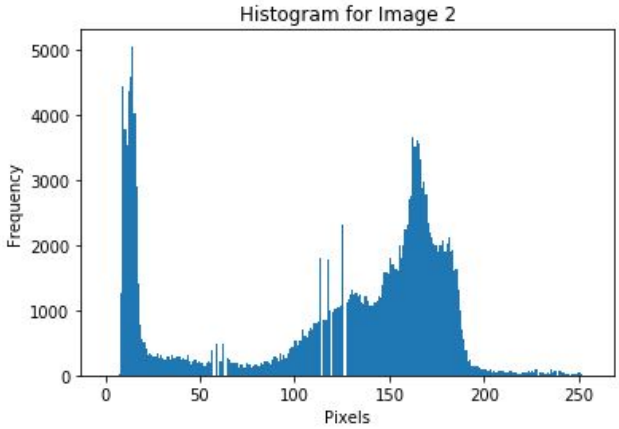
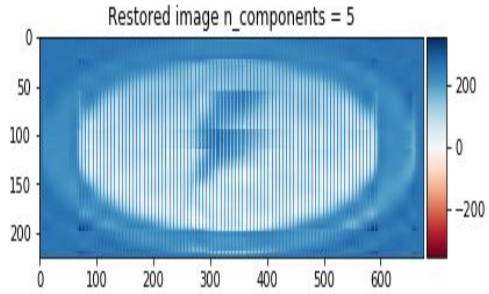
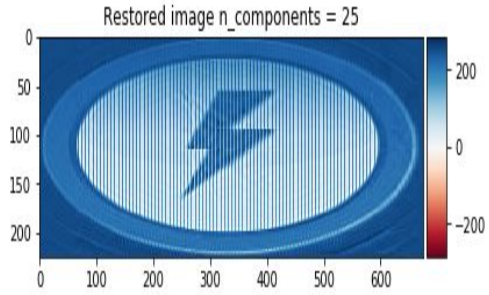


Fig.5 :- PCA results for dataset1



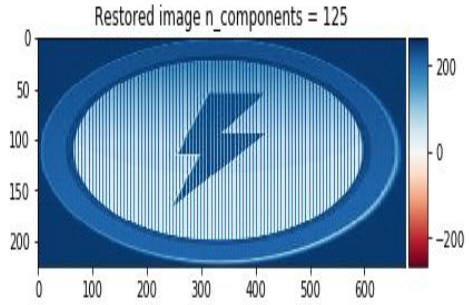
Variance retained 82.17642013975333%

Compression Ratio 0.7407407407407408%



Variance retained 96.07665920738255%

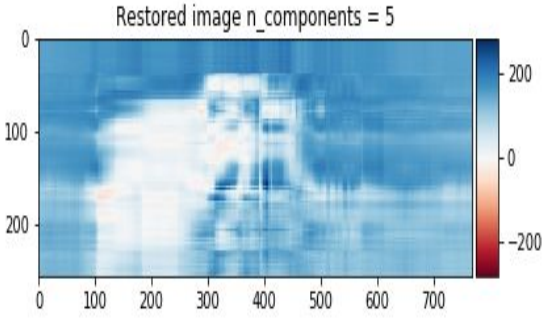
Compression Ratio 3.7037037037037033%



Variance retained 99.20030910177884%

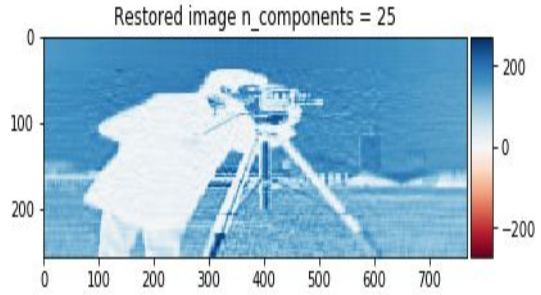
Compression Ratio 18.51851851851852%

Fig.6 :- PCA results for dataset2



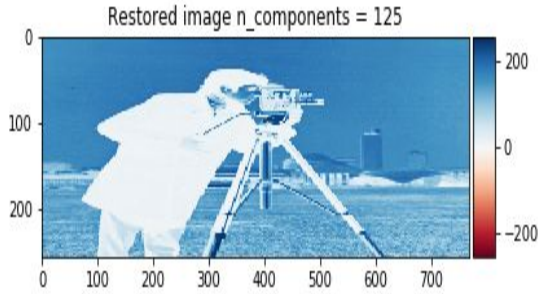
Variance retained 85.04292889057821%

Compression Ratio 0.6510416666666667%



Variance retained 96.24201764008698%

Compression Ratio 3.2552083333333335%



Variance retained 99.20109958323904%

Compression Ratio 16.276041666666664%

Fig.7: -Denoised image by total variation minimization

Fig.8:-Error term with iteration(left) and total variation with iteration(right)

Fig.9:-MSE Change tendency with weight from 1-9(left) and with weight from 2.5-4.0(right)

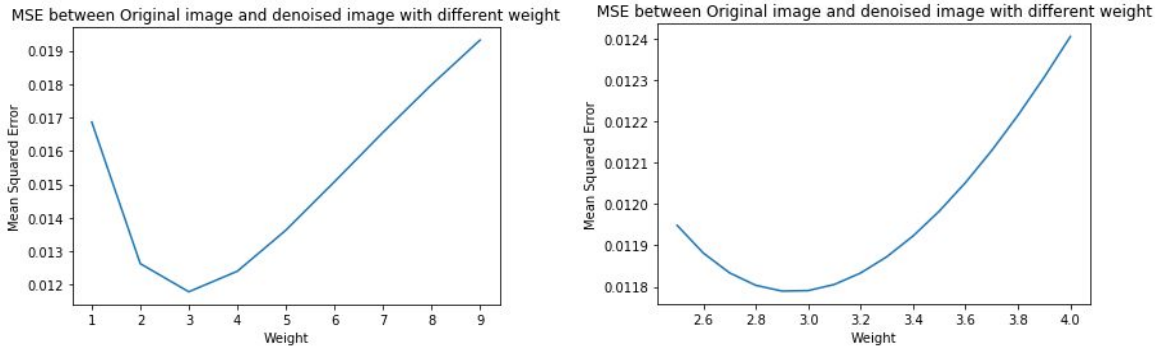


Fig.10-Denoised images by TV, BT, WT, more TV, more BT, more WT.

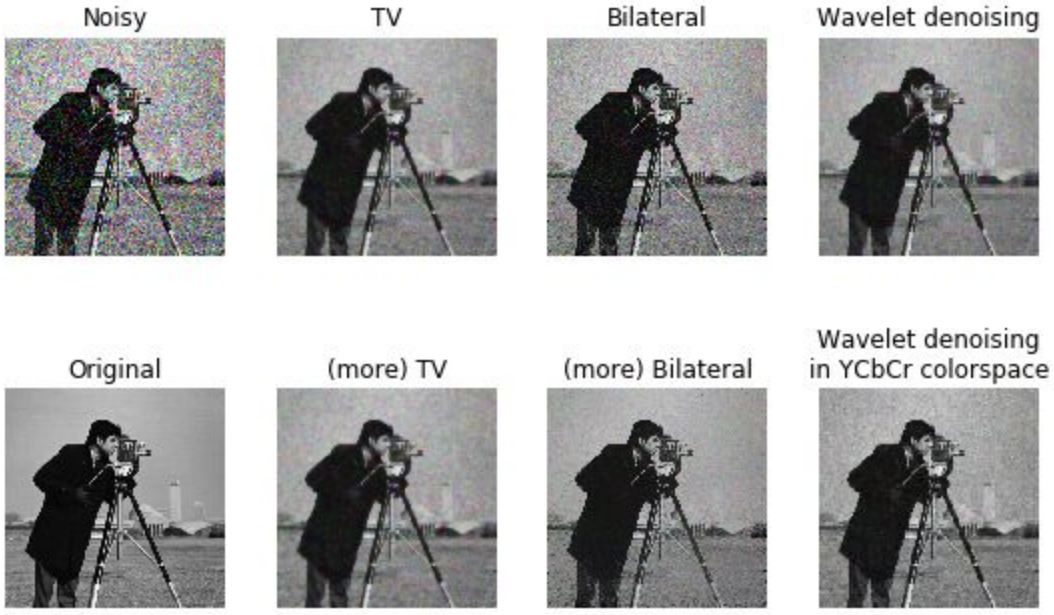


Table.1:- MSE and SD of Gaussian Noise for TV, BT, WT, more TV, more BT, more WT

Metrics	Noisy	TV	BT	WT	More TV	More BT	More WT
MSE	0.0637	0.0121	0.0217	0.0144	0.0118	0.0115	0.0095
Estimated SD of Gaussian Noise	0.1434	0.0131	0.0776	0.0038	0.0095	0.0370	0.0081

REFERENCES

[1] Junyuan X, Linli X & Enhong C. *Image Denoising and Inpainting with Deep Neural Networks*. Retrieved from https://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks.pdf

[2] Antoni B, Bartomeu C & Jean-Michel M. (2005). *A review of image denoising algorithms, with a new one.* SIAM Journal on Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal, 2005, 4 (2), pp.490-530.

[3] Marcelo B & Guillermo S. *Image Inpainting*. Retrieved from https://www.dtic.upf.edu/~mbertalmio/bertalmi.pdf

[4] A.K. Jain," Fundamentals of Digital Image Processing", Englewood Cliffs, NJ: Prentice Hall, 1989.

[5]Tomasi, C; Manduchi, R (1998). Bilateral filtering for gray and color images (PDF). Sixth International Conference on Computer Vision. Bombay. pp. 839–846

[6]Banterle, F.; Corsini, M.; Cignoni, P & Scopigno, R. (2011). "A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain". Computer Graphics Forum. 31 (1).

[7] T. Goldstein & S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343. Retrieved from ftp://ftp.math.ucla.edu/pub/camreport/cam08-29.pdf

[8] scikit-image. *Module: restoration*. Retrieved from http://scikit-image.org/docs/dev/api/skimage.restoration.html#skimage.restoration.denoise_tv_chambolle

[9] Adrian R. (2014). pyimagesearch. *How-To: Python Compare Two Images.* Retrieved from https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/

[10] Antonio M & Stanley J. O. (2008). *Image Super-Resolution by TV-Regularization and Bregman Iteration.* Retrieved from https://www.researchgate.net/profile/Antonio_Marquina/publication/230604847_Image_Super-Resolution_by_TV-Regularization_and_Bregman_Iteration/links/54c0b4900cf28eae4a696e70.pdf

[11] Mohsen G, George H. F & Edward R. V. (2003). IEEE TRANSACTIONS ON IMAGE PROCESSING. *Fractal Image Denoising.* Retrieved from https://pdfs.semanticscholar.org/f267/7bac38ffdf4639f6eb21e0e4684584851d1b.pdf