Translation Invariance (TI) based Novel Approach for better De-noising of Digital Images

Ranganadh Narayanam*

**Assistant Professor, Electronics & Communications Engineering, ICFAI Tech School, IFHE*

Abstract –Image de-noising is an important image processing task, both as a process itself, and as a component in other processes. Image de-noising means removing unwanted noised form a corrupted image to restore the original image. There are many ways to de-noise an image or a set of data of images, exist. Image De-noising has remained a fundamental problem in the field of image processing. Reducing noise from the original image is still a challenging task for researchers. There are several algorithms and methods each having assumptions, advantages, and disadvantages. Noise can be introduced by transmission errors and compression. In this paper we have introduced an innovative method of Translation Invariant (TI) algorithmic approach for de-noising images to improve SNR. We have applied this algorithm for various Digital Image Processing filters: convolution filter, imaging filter, wiener filter, cross correlation filter, Gaussian filter, order-static filter, wavelet de-noising filter, sliding neighborhood filter. We de-noised the images with all these filters without TI and with TI and we found with TI, the filters are working with better performance than without TI. It is also verified using edge detection by observing the preservation accuracy of edges. In this research we made some useful novel observations, specified as conclusions.

Key terms – Image de-noising, edge detection, Translation-Invariance, Image filtering, edge detection, SNR.

1. Introduction to the image de-noising filters: Translation Invariance Approach

Digital images play an important role both in daily life applications such as satellite television, magnetic resonance imaging, computer tomography as well as in areas of research and technology such as geographical information systems and astronomy. Data sets collected by image sensors are generally contaminated by noise. Imperfect instruments, problems with the data acquisition process, and interfering natural phenomena can all degrade the data of interest. Furthermore, noise can be introduced by transmission errors and compression. Thus, de-noising is often a necessary and the first step to be taken before the images data is analyzed. It is necessary to apply an efficient de-noising technique [1,2] to compensate for such data corruption. Image de-noising still remains a challenge for researchers [3]. This paper describes different methodologies for noise reduction (or de-noising) giving an insight as to which algorithm should be used to find the most reliable estimate of the original image data given its degraded version. Here we have introduced an innovative approach called Translation Invariant algorithm, for time domain of the images. The basic algorithm has the following steps

1) Shift the noised image for a number of shifts
2) Apply the de-noising using the given filter for each shift
3) Un-shift the de-noised image for each shift
4) Average the results

5) This gives the Translation invariant de-noised image using that filter under consideration

Without TI means de-noise the noised image without shifting using the given filter under consideration.

Here we did the de-noising of the images using the following filters without TI, and with TI for 3 shifts & 5 shifts.

1.1 Convolution filter

Convolution is the treatment of a matrix by another one which is called "kernel" [12]. The Convolution Matrix filter uses a first matrix which is the Image to be treated. The image is a bi-dimensional collection of pixels in rectangular coordinates. The used kernel depends on the effect we want. We can use 5x5 or 3x3 matrices. We will consider only 3x3 matrices, they are the most used and they are enough for all effects we want. If all border values of a kernel are set to zero, then system will consider it as a 3x3 matrix. The filter studies successively every pixel of the image. For each of them, which we will call the "initial pixel", it multiplies the value of this pixel and values of the 8 surrounding pixels by the kernel corresponding value. Then it adds the results, and the initial pixel is set to this final result value. With the Convolution Matrix filter, you can build a custom filter. The filters with convolution are relatively simple. More complex filters, that can use more fancy functions, exist as well, and can do much more complex things , for example the Colored Pencil filter in Photoshop.
For example take the following

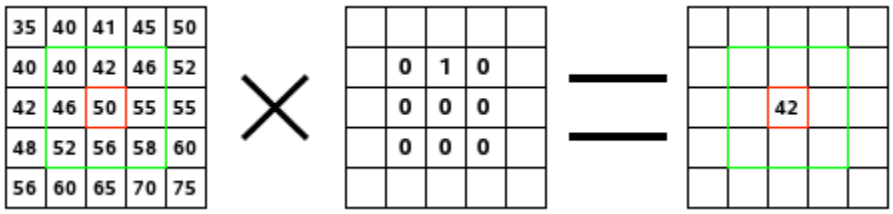

Figure 1. Example showing convolution on imaging

Here is what happened: the filter read successively, from left to right and from top to bottom, all the pixels of the kernel action area. It multiplied the value of each of them by the kernel corresponding value and added results. The initial pixel has become 42: (40*0)+(42*1)+(46*0) + (46*0)+(50*0)+(55*0) + (52*0)+(56*0)+(58*0) = 42. (The filter doesn't work on the image but on a copy).

There is a MATLAB function to explore this filter: **"conv2()"**

1.2 Imaging filter:

It is an averaging filter for 2D images in image processing for which there is a MATLAB function **"imfilter()"**.

1.3 Wiener filter:

Pixel wise adaptive Wiener method [13] based on statistics estimated from a local neighborhood of each pixel. The Wiener filtering executes an optimal tradeoff between inverse filtering and noise smoothing. It

removes the additive noise and inverts the blurring simultaneously. The Wiener filtering is a linear estimation of the original image. The approach is based on a stochastic framework. The orthogonality principle implies that the Wiener filter in Fourier domain can be expressed as follows:

$$W(f_1, f_2) = \frac{H^*(f_1, f_2) S_{xx}(f_1, f_2)}{|H(f_1, f_2)|^2 S_{xx}(f_1, f_2) + S_{\eta\eta}(f_1, f_2)}$$

$S_{xx}(f_1, f_2)$, $S_{\eta\eta}(f_1, f_2)$ power spectra of original image and additive noise. $H(f_1, f_2)$ is the blurring filter. Wiener filter has two separate parts, an inverse filtering part and a noise smoothing part. It not only performs the de-convolution by inverse filtering (high-pass filtering) but also removes the noise with a compression operation (low-pass filtering).

To explore this filter there is a MATLAB function: **"wiener2()"**

1.4 Cross correlation:

Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other [14]. Its present-day applications are almost innumerable and include image analysis, image compression, velocimetry, and strain estimation [3, 4].
In discrete form, the cross correlation can be defined as

$$f(x,y) \circ k(x,y) = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} f(i,j)\, k(x+i, y+j)$$

W & H are the width and height of the image. If f = k, it is auto correlation [15].

To explore this filtering approach there is a MATLAB function: **"xcorr2()"**

1.5 Gaussian filtering:

The idea of Gaussian smoothing [17] is to use this 2-D distribution as a `point-spread' function. The Gaussian filter is a non-uniform low pass filter. The Gaussian smoothing is used to `blur' images and remove detail and noise. It uses a different type of kernel with special properties that represents the shape of a Gaussian (`bell-shaped') hump. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. Following box shows a suitable integer-valued kernel that approximates a Gaussian with a **σ** of 1.0. It is not obvious how to pick the values of the mask to approximate a Gaussian [5]. One could use the value of the Gaussian at the centre of a pixel in the mask, but this is not accurate because the value of the Gaussian varies non-linearly across the pixel. The values of the Gaussian over the whole pixel (by summing the Gaussian at 0.001 increments) are integrated. The integrals are not integers: rescaled the array so that the corners had the value 1. Finally, the 273 is the sum of all the values in the mask. A further way to compute a Gaussian smoothing with a large standard deviation is to convolve an image several times with a smaller Gaussian. While this

is computationally complex, it can have applicability if the processing is carried out using a hardware pipeline.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 2. Box showing discrete approximation to Gaussian function with **σ** =1.0

The Standard deviation of the Gaussian function [16] plays an important role in its behavior. In the following figure, the values located between +/- σ account for 68% of the set, while two standard deviations from the mean (blue and brown) account for 95%, and three standard deviations (blue, brown and green) account for 99.7%. Account for 99.7%. This is very important when designing a Gaussian kernel of fixed length.

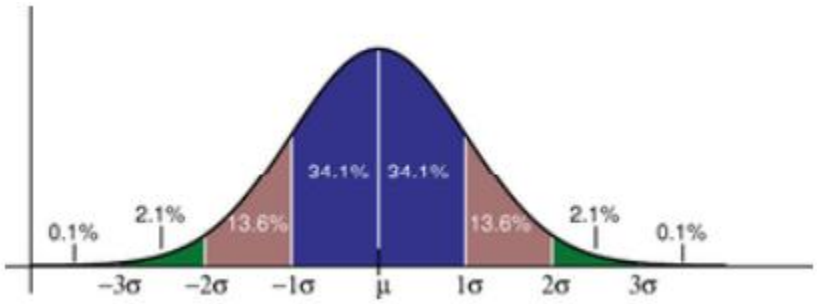

Figure 3. The kernel is rotationally symmetric with no directional bias. Gaussian kernel is separable which allows fast computation. Gaussian kernel is separable, which allows fast computation.

To explore this filtering approach there is a MATLAB function: **"imgaussfilt()"**

1.6 Order static filters:

Order Statistics filters are nonlinear spatial filters [11] which are based on ordering the pixels contained in an image. Usually, sliding window technique is employed to perform pixel-by-pixel operation in a filtering algorithm. The local statistics obtained from the neighborhood of the center pixel gives a lot of information about its expected value. If the neighborhood data are ordered (sorted), then ordered statistical information is obtained. If this order statistics vector is applied to a finite impulse response

(FIR) filter, then the overall scheme becomes an order statistics (OS) filter. They are differentiated based on how they choose the values in the sorted list. Minimum and maximum filter, the minimum filter selects the smallest value within the pixel values and maximum filter selects the largest value within of pixel values. This is accomplished by a procedure which first finds the minimum and maximum intensity values of all the pixels within a windowed region around the pixel. If the intensity of the central pixel lies within the intensity range spread of its neighbors, it is passed on to the output image unchanged. However, if the central pixel intensity is greater than the maximum value, it is set equal to the maximum value; if the central pixel intensity is less than the minimum value, it is set equal to the minimum value. The minimum and maximum filters are represented as follows:

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

Median filter

The median filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.)

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$

Alpha Trimmed Mean Filter

The alpha-trimmed mean (ATM) filter is based on order statistics and varies between a median and mean filter. It is so named because, rather than averaging the entire data set, a few data points are removed (trimmed) and the remainders are averaged. The points which are removed are most extreme values, both low and high, with an equal number of points dropped at each end (symmetric trimming). In practice, the alpha-trimmed mean is computed by sorting the data low to high and summing the central part of the ordered array. The number of data values which are dropped from the average is controlled by trimming parameter alpha which is being expressed as:

$$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

To explore this filtering approach there is a MATLAB function: **"ordfilt2()"**

1.7 Wavelets de-noising:

In many applications, image de-noising is used to produce good estimates of the original image from noisy observations. The restored image should contain less noise than the observations while still keep sharp transitions (i.e. edges). Wavelet transform, due to its excellent localization property [10], has rapidly become an indispensable signal and image processing tool for a variety of applications, including compression and de-noising. Wavelet de-noising attempts to remove the noise present in the signal while preserving the signal characteristics, regardless of its frequency content. It involves three steps: a linear forward wavelet transform, nonlinear thresholding step and a linear inverse wavelet transform. Wavelet thresholding (first proposed by Donoho) is a signal estimation technique that exploits the capabilities of wavelet transform for signal de-noising. It removes noise by killing coefficients that are insignificant relative to some threshold, and turns out to be simple and effective, depends heavily on the choice of a thresholding parameter and the choice of this threshold determines, to a great extent the efficacy of de-noising. Researchers have developed various techniques for choosing de-noising parameters and so far there is no "best" universal threshold determination technique. The system is expressed in the following figure 4.

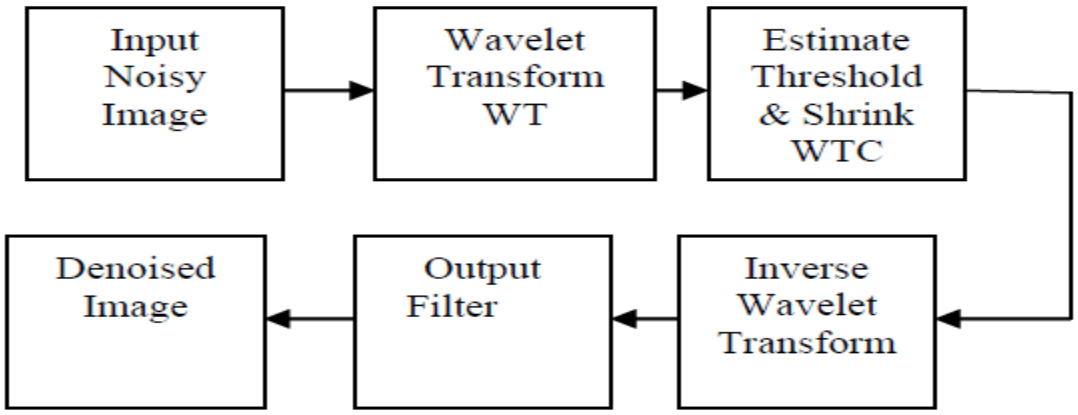

Figure 4. De-noising using wavelets thresholding

1.8 Sliding neighborhood operations filter

This filtering is available in MATLAB function **"nlfilter()".**

B = nlfilter(A, [m n], fun) applies the function fun to each m-by-n sliding block of the grayscale image A. fun is a function that accepts an m-by-n matrix as input and returns a scalar result. c = fun(x), fun must be a function handle. c is the output value for the center pixel in the m-by-n block x. nlfilter calls fun for each pixel in A. nlfilter zero-pads the m-by-n block at the edges, if necessary. B = nlfilter(A, 'indexed',...) processes A as an indexed image, padding with 1's if A is of class single or double and 0's if A is of class logical, uint8, or uint16.

2. Result Analysis

Here we did research using the Translation Invariance (TI) technique for all the filters under consideration. We did De-noising using all the filters without TI, and then with TI for 3 shifts & 5 shifts. MATLAB programming is done using MATLAB 2016a. Some of the results are provided in the following Figure 5,6,7,8. The results are obtained for various noise conditions of variances 0.001, 0.01,0.05, 0.1 & 0.5. The Signal to Noise Ratio values are provided in the Table 1. It is clearly observable that Signal to Noise Ratio values of each filter are higher in the case of TI based de-noising than without TI de-noising. So it is clear that the approach of Translation Invariance for all the filters surely improves the image quality than without TI. It is also observed that with more number of shifts of TI gives better de-noising of the image. This is observable in the case of 5 shifts to 3 shifts, where SNR is improved in the case of 5 shifts when compared to 3 shifts. Edge detection is also done for all the cases for the efficiency confirmation of our novel TI approaches for all filters. Some of the edge detection results images are given in the Figure 9, 10, 11. Canny edge detection is the most prominent edge detection and is recognized as excellent edge detection operator [7,8,9]. So, we applied canny edge detection operator. It is clear that de-noised with TI is preserving edges in a better way. Wavelets de-noising filters are working with better performance than all the remaining filters. When we go through the specific observation some filters are working better than some other filters in low noise conditions, but when noise level increases their performance deteriorating and the other filters are working with better performance.

If we observe the performance of conv and imfilter with all the remaining filters (except wavelet filters), it is found that even though the remaining filters are performing better than these two filters in low noisy conditions, when it comes to highly noisy conditions these two filters are performing better than the remaining filters, and the remaining filters' performances are largely going down when compared to these two filters. So, when it comes to higher and higher noisy conditions these two filters are highly useful when compared to the remaining all filters under observation. In this paper it is proved that we can improve the SNR, for improving the quality of the image by using the proposed Translation Invariant algorithm. It is also proved that if we increase the number of shifts of this TI algorithm from 3 to 5 also the SNR improving. Out of all the filters it is clearly proved that wavelet de-noising filtering procedure is highly performing in both low and high noisy conditions than any filter under observation. With TI algorithm application for wavelet de-noising it is further performing with high SNR improvements, and also if number of shifts increased from 3 to 5 also its performance is increasing in improving the image quality in terms of SNR. Correlation filter did the least de-noising performance, as it is clear from the table that its image quality improvement SNR values are less than all the filters.

3. Conclusions

As a conclusion, in this research we proved the following:

1) Translation Invariant (TI) algorithm does better de-noising than without TI for any filter for improving the quality of the image.

2) 'Convolution' and 'imfilter' filtering are performing less in low noise conditions, but when it comes to highly noisy conditions these two are performing better than all the remaining filters under observation (except wavelets)
3) Wavelets based de-noising filter (without TI), wavelets based TI de-nosing filter are working better than all the remaining filters, and it is performing better in low and also high noisy conditions, than all the remaining filters under observation.
4) Least de-noising is done by correlation filter than all the filters.

In addition to these conclusions, we confirmed that where ever there is improvement of SNR values, there is better preservation of edges, which further confirms that filters with TI based de-noising works better than without TI. This edge preservation confirms that wavelets do the best de-noising than all; and with TI it is better edge preserver than all.

Figure 5 De-noising performances of Translation Invariance Algorithm for Convolution filter under various noise conditions. (a) Original Image (b) Noised Image (c) Without TI De-noised image (d) With TI De-noised image.

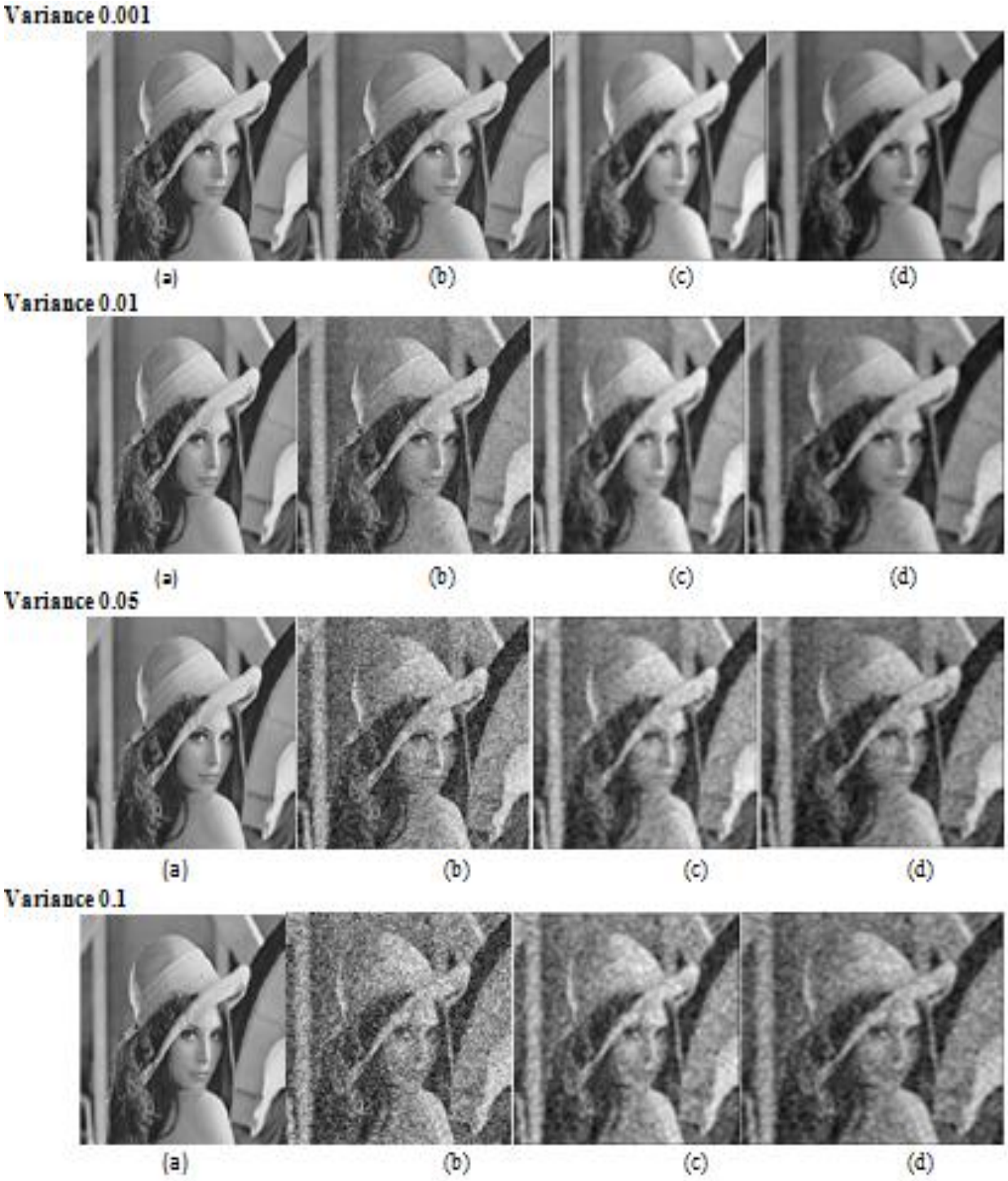

Figure 6 De-noising performances of Translation Invariance Algorithm for image filter "imfilter" under various noise conditions. (a) Original Image (b) Noised Image (c) Without TI De-noised image (d) With TI De-noised image.

Figure 7 De-noising performances of Translation Invariance Algorithm for wiener filter under various noise conditions. (a) Original Image (b) Noised Image (c) Without TI De-noised image (d) With TI De-noised image.

Figure 8 De-noising performances of Translation Invariance Algorithm for wavelets filter under various noise conditions. (a) Original Image (b) Noised Image (c) Without TI De-noised image (d) With TI De-noised image.

	Without TI	With TI – 3 shifts	With TI – 5 shifts
Variance	SNR for "conv2" filter		
0.001	17.4501	19.6982	24.6541
0.01	16.9234	19.3466	23.6785
0.05	16.8724	18.0041	22.5430
0.1	15.9821	16.6197	20.0087
0.5	11.8024	13.1368	19.1109
Variance	SNR for "imfilter" filter		
0.001	19.1967	19.7018	23.1265
0.01	18.7347	19.3334	22.9860
0.05	17.0561	17.9285	21.0096
0.1	15.6594	16.6921	20.1256
0.5	11.7793	13.1107	18.0098
Variance	SNR for "wiener2" filter		
0.001	24.7124	25.4725	28.2321
0.01	21.2689	22.6969	26.1209
0.05	17.0916	18.4940	21.9987
0.1	15.1589	16.4386	18.9987
0.5	11.4841	12.7124	17.9007
Variance	SNR for "xcorr2" filter		
0.001	9.9466	12.4417	14.5078
0.01	9.8496	12.3863	13.9989
0.05	8.9087	12.1176	13.5609
0.1	7.9466	11.9215	13.0009
0.5	7.0789	10.7493	12.1100
Variance	SNR for "imgaussfilt" filter		
0.001	28.1594	26.8764	29.0897
0.01	18.0285	19.5698	24.9008
0.05	11.6502	13.2217	20.0098
0.1	9.2913	10.8652	18.6750
0.5	5.6435	7.2212	16.0078
Variance	SNR for "ordfilt2" filter		
0.001	20.4410	21.9622	28.9002
0.01	18.1918	19.8580	24.1207
0.05	14.3278	16.0595	19.0123
0.1	12.2250	13.9176	17.1023
0.5	7.1363	8.7280	13.0123
Variance	SNR for "wavelets" filter		
0.001	39.0012	48.0192	58.1002
0.01	38.7901	47.0987	54.0112
0.05	34.1020	44.0998	52.0987
0.1	30.1203	41.2890	50.0098
0.5	29.0998	37.0112	45.0098
Variance	SNR for "nlfilter" filter		
0.001	24.4288	25.9384	30.0789
0.01	19.9662	21.5250	28.0098

0.05	**14.4142**	**15.9797**	**18.0101**
0.1	**11.7991**	**13.3475**	**17.4907**
0.5	**5.9810**	**7.5310**	**12.1987**

Table 1 Table showing the performances of various Digital Image Processing filters in various noise conditions. TI - Translation Invariance. All filters are improving their performance in terms of SNR from Without TI de-noising to With TI de-noising. SNR performance also improving from 3 shifts to 5 shifts in with TI. Wavelets de-noising filters are working with better performance than all the remaining filters. When we go through the specific observation some filters are working better than some other filters in low noise conditions, but when noise level increases their performance deteriorating and the other filters are working with better performance.

(a) (b) (c) (d)

Figure 9 de-noising using Conv2 filter. (a) Original image (b) noised image variance 0.001 (c) de-noised image without TI technique (d) de-noised image with TI technique. It is clearly observable that with TI is de-noising better.

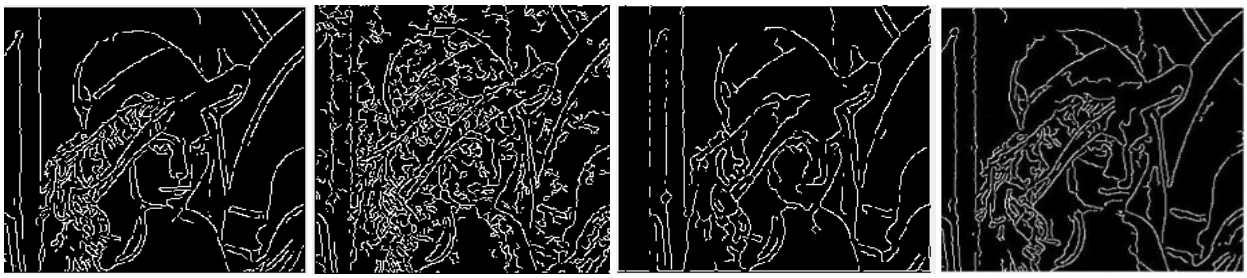

Figure 10 de-noising using wiener filter. (a) Original image (b) noised image variance 0.05 (c) de-noised image without TI technique (d) de-noised image with TI technique. It is clearly observable that with TI is de-noising better.

Figure 11 de-noising using wavelets. (a) Original image (b) noised image variance 0.01 (c) de-noised image without TI technique (d) de-noised image with TI technique. It is clearly observable that with TI is de-noising better.

References

[1] Rafael C Gonzalez and Richard E Woods, "Digital Image Processing", third edition, Pearson Education,2007

[2] R. Y. Xia and C. C. Bao, "Wiener filtering based speech enhancement with weighted denoising auto-encoder and noise classification," Speech Communication, vol. 60, pp. 13–29, 2014.

[3] M. Boix and B. Canto, "Using wavelet denoising and mathematical morphology in the segmentation technique applied to blood cells images," Mathematical Biosciences and Engineering, vol. 10, no. 2, pp. 279–294, 2013.

[4] T. Saba, A. Rehman, A. Al-Dhelaan, and M. Al-Rodhaan, "Evaluation of current documents image denoising techniques: a comparative study," Applied Artificial Intelligence, vol. 28, no. 9,
pp. 879–887, 2014.

[5] B. K. K. Shreyamsha, "Image denoising based on gaussian/bilateral filter and its method noise thresholding," Signal, Image and Video Processing, vol. 7, no. 6, pp. 1159–1172, 2013.

[6] R. M. Yan, L. Shao, L. Liu, and Y. Liu, "Natural image denoising using evolved local adaptive filters," Signal Processing, vol. 103, pp. 36–44, 2014.

[7] Q. Xu, S. Varadarajan, C. Chakrabarti, and L. J. Karam, "A distributed canny edge detector: algorithm and FPGA implementation," IEEE Transactions on Image Processing, vol. 23, no.
7, pp. 2944–2960, 2014.

[8] R.R. Coiman and D.L. Donoho, "Translation-Invariant De-noising", wavelets and statistics, springer-verlag, New York 1995.
[9] Jun Li, Sheng Ding, "A Research on Improved Canny Edge Detection Algorithm", CCIS, ICAIC 2011, Springer-Verlag Berlin Heidelberg 2011

[10] S. mohideen, S. A. perumal, M.M. sathik, "Image de-noising using discrete wavelet transform", IJCSNS, Vol 8, 2008.

[11] I. Pitas, A.N. Venetsanopoulos, "Order statistics in digital image processing", Proceedings of the IEEE (Volume: 80, Issue: 12, Dec 1992)

[12] O. Fialka, C Martin, "FFT and Convolution Performance in Image Filtering on GPU", IEEE 10th international conference on information visualization, 2006.

[13] P.Biswas, AS sarkar, M. Mohemmed, "Deblurring Images using a Wiener Filter",IJCA, January 2015

[14] Beckett, J. and Bancroft, J.C., "Event detection in prestack migration using matched filters", CREWES Research Report, 14, 2002

[15] C Yuganya , A V Khirthana , and Udayakumara pandian "2D cross correlation multi-modal image recognition", JGRCS, April 2013.

[16] Elhanan Elboher, Michael Werman "Efficient and Accurate Gaussian Image Filtering Using Running Sums",

[17] G Deng, LW Cahill, "An adaptive Gaussian filter for noise reduction and edge detection", Nuclear Science Symposium and Medical Imaging Conference, 1993., 1993 IEEE Conference Record.

Author's Biography

Prof. Ranganadh Narayanam is an Assistant Professor in the department of Electronics & Communications Engineering in IFHE Deemed University, Hyderabad, India. Mr.Narayanam, graduated with his research master degree from University of Texas at San Antonio. He worked on the research of DSP software and hardware design and implementations in UTSA. He was a foreign student research grant holder while studying at UTSA, from the government of Texas state, USA. He worked as a research trainee in University of Ottawa in Auditory Neural Signal Processing. He worked in the area of Brain Imaging in University of California Berkeley. Mr. Narayanam's research interests include "DSP & DIP techniques and algorithms software & Hardware design and implementations and their specific applications into neuroscience". He has 7 years of Teaching & 11 years of research experience.