A Generic Transfer Function based Technique for Estimating Noise from Images

Krishnan Kutty
Associate Technical Fellow
KPIT Cummins Infosystems Ltd.
Pune, India

Swati Ojha
Research Associate
KPIT Cummins Infosystems Ltd.
Pune, India

ABSTRACT
Estimation of noise from an image continues to be a challenging area of research in the field of image processing. However, noise estimation from images that inherently contains very fine details or which have textured regions is still a challenging task. This paper attempts to estimate noise content from images corrupted by Gaussian and Speckle noise. The noise estimation technique proposed here is based on deriving a generic transfer function. This transfer function attempts to map the median value of the local noise standard deviation that is calculated on overlapping sub-images of the noisy image to the overall noise deviation in the image. The results obtained show that the proposed algorithm performs well for different types of images and over a large range of noise deviation. Comparison with other known standard techniques in literature is also presented in the paper, which confirms that proposed method provides better noise estimation. The approach has been proven to work on images affected by speckle noise as well. Results for estimation of speckle noise are also presented in this paper.

General Terms
Image Processing, Noise Removal

Keywords
Gaussian noise, Standard deviation, Noise estimation, Estimation error, Speckle noise

1. INTRODUCTION
Noise removal is a very important processing step for most of the digital imaging applications [1]. Some of the well known noise removal algorithms require prior knowledge about the content of noise in the image [2] [3].The noise content for standard type of noise (Gaussian\speckle) can be expressed in terms of noise variance or noise standard deviation. In 1993, Olsen gave a complete description and comparison of various noise estimation algorithms that were available then. In general, these algorithms are classified into two different categories: filter-based and block-based [4]. Since 1993, many approaches have been presented in order to obtain a good estimate of noise content in an image.

J. Immerkaer's [6] fast noise estimation method is a very simple way of calculating the noise standard deviation. He proposed that the standard deviation of the noisy image 'N' of size m x n can then be computed as:

$$\sigma_n = \sqrt{\frac{\pi}{2}} \quad \frac{1}{6mn} \sum_{i,j=1}^{mn} \left|(N * Mask)_{i,j}\right| \qquad (1)$$

Where,

$$Mask = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad (2)$$

This method only requires convolution and averaging and is therefore computationally less expensive. It works well for images corrupted with noise of high noise standard deviation. However, it overestimates noise in case of low standard deviation; since, it considers fine details inherent to the image to be noise.

In 1999, Rank, Lendl, and Unbehauen [5], proposed another method for noise variance estimation. Their algorithm estimates the noise variance in three steps. As a first step, the original image structure is suppressed by cascaded vertical and horizontal difference operators. The second step involves computation of histogram of local standard deviation. Finally, an iterative evaluation of the histogram is performed to estimate noise. Since it involves iterative histogram analysis, it is computationally expensive. Moreover, a-prior knowledge about the histogram is required. This method also suffers from the problem of over estimation due to poor suppression of smooth regions. Bilcu and Vehvilainen [7] proposed a method in order to improve the work by Rank et al. The algorithm involves following steps: Laplacian convolution, Edge detection, Local variance computation, Histogram computation and statistical averaging. Bilcu et al have overcome the problem of overestimation by using Laplacian convolution which suppresses image structure more efficiently. However, they continued to use histogram based statistics which makes it computationally intensive. Bosco et.al [8] explained a noise estimation method based on histogram approximation to overcome inaccuracy of averaging. The main weakness of this method is that it takes more time and storage to accumulate differences. Amer et al [9] introduced a new approach to find homogeneous regions using special masks. The method includes use of eight high-pass operators and special masks for corners to stabilize the homogeneity estimation for each block. Shin et al [10] explained noise estimation algorithm based on both block-based and filtering-based approaches. It selects homogeneous blocks by a block-based approach and then filters the selected homogeneous blocks using a filtering-based approach.

In 2008, Shen-Chuan Tai and Shih-Ming Yang [11] modified J. Immerkaer's method by introducing edge detection block before Laplacian convolution. In order to obtain edge map, they have used Sobel edge detector with adaptive threshold selection. After finding the edge map, they follow the same approach as [6], but exclude the edge pixels. This method gives good results for low noise deviation but tends to overestimate noise for images with fine details or textured regions.

The proposed method in this paper is a further improvement to the approach proposed by Tai et al. It estimates noise more accurately over large range of noise variances for a variety of images. The rest of the paper describes our proposed method in detail and is organized as follows. Section 2 describes the noise model. Section 3 describes our proposed method. Section 4 presents the results and discussion. Section 5 reports on conclusion and scope for future work.

2. NOISE MODEL

2.1 Additive/Gaussian Noise

All In the case of additive white Gaussian noise (AWGN), all the image pixels deviate from their original values following the Gaussian curve. The probability density function (PDF) for a zero mean Gaussian distribution is given as:

$$P_g(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-n^2}{2\sigma^2}} \quad (3)$$

For each image pixel with intensity value I_{ij} ($1 \le i \le m$, $1 \le j \le n$; for an m x n image), the corresponding pixel of the noisy image N_{ij} is given by,

$$N_{ij} = I_{ij} + G_{ij} \quad (4)$$

Where, each noise value 'G' is drawn from a zero-mean Gaussian distribution.

2.2 Multiplicative/Speckle Noise

This type of noise can be expressed as a unity-mean normal random process with PDF given by

$$P_s(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(n-1)^2}{2\sigma^2}} \quad (5)$$

For each image pixel with intensity value I_{ij} ($1 \le i \le m$, $1 \le j \le n$; for an m x n image), the corresponding pixel of the noisy image N_{ij} is given by,

$$N_{ij} = I_{ij} + I_{ij}S_{ij} \quad (6)$$

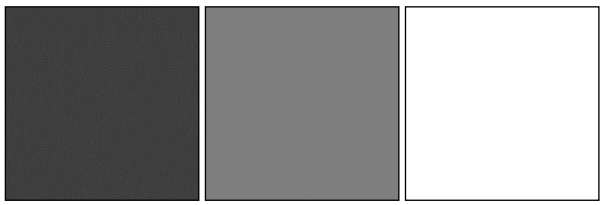
Where, each noise value *'S'* is drawn from unity-mean normal random distribution.

3. PROPOSED ALGORITHM

The proposed algorithm works in two phases: generating the transfer function and noise estimation. The transfer function is generated only once a-priori; and then it is used to estimate noise from any noisy image.

3.1 Generating the Transfer Function

In order to obtain transfer function, we have used three images, each having all the pixels of same gray values, as shown in Fig.1.These images are of size 256x256; and have uniform intensities of 64,128 and 255 respectively.



(a) (b) (c)

Fig 1: Source images

We first convolve the image with a Laplacian mask as given in (1). Because of this, a majority of the smooth regions in the image is suppressed, leaving behind mostly the noisy regions. From this 'noise only image', we compute the local standard deviation of the noisy pixels. For this, we have tested with several different window sizes and found that the proposed algorithm works best for a window size of 11 x 11. We then find the median value of local standard deviation for the entire image as described below:

$$std_{med} = median(S_{11}, S_{12}, \ldots\ldots\ldots S_{qp}) \quad (7)$$

Where, $S_{11}, S_{12}, \ldots\ldots\ldots S_{qp}$ are the standard deviation values of the overlapping 11 x 11 blocks of image convolved with mask given in (2). This is depicted in Fig.2, where the red, pink and the blue boxes are the illustrative overlapping 11x11 boxes; the pixels within which are used to compute the local standard deviationS_{ij}.

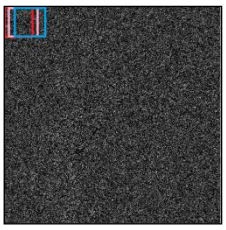


Fig 2: Overlapping Boxes considered for calculating median of local standard deviation

As a next step, we find a regression between the local characteristic computed from the noisy image against the standard deviation of the input image. We have repeated all the above steps for all three source images by varying the noise deviation.

In case of additive noise (Gaussian), as shown in Fig.3, a graph is plotted between the input standard deviation and median value of the local standard deviation for that image. Here, the input standard deviation is varied from 1 to 40 for all the three source images.

For multiplicative (speckle), noise, as shown in Fig.4, a graph is plotted between the input standard deviation and median value of the local standard deviation divided by mean of that image. Here, the input standard deviation is varied from 1 to 10 for all the three source images.

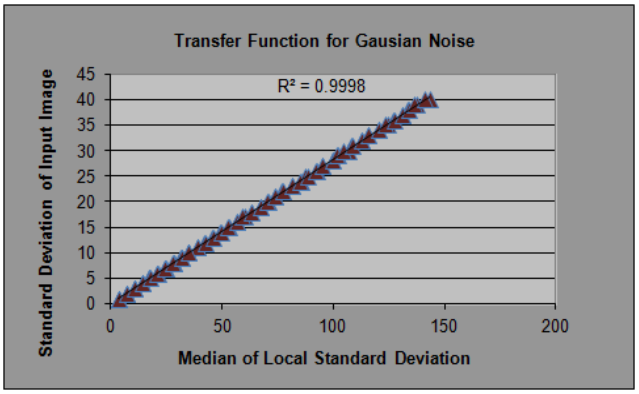


Fig 3: Graph between input standard deviation and median of the local standard deviation for source images (additive noise)

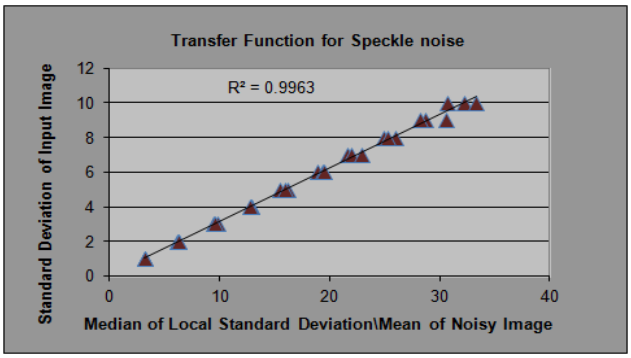


Fig 4: Graph between input standard deviation and median of the local standard deviation divided by mean for source images (multiplicative noise)

As seen from Fig. 3 and Fig. 4, for a liner regression, the R-square value was more than 0.99 considering all the three source images. We then derive a transfer function that relates the image standard deviation to the local characteristic as calculated from individual images. The transfer functions are as given below:

For additive noise (Gaussian),

$$\sigma_n = (0.2833 * std_{med}) + 0.0155 \quad (8)$$

For multiplicative noise (Speckle),

$$\sigma_n = \left(0.3101 * \frac{std_{med}}{mean\_val}\right) + 0.05 \quad (9)$$

Where,

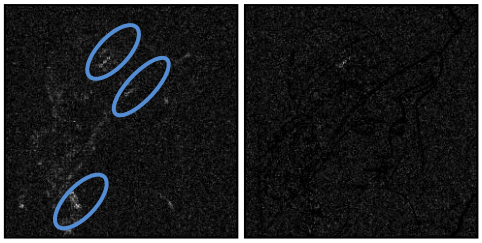
std_{med} is the median of local standard deviations

$mean\_val$ is the mean of the entire image

3.2 Noise Estimation

In order to estimate noise, we first convolve the image with a Laplacian mask. This is to suppress the smooth regions of the image. However, some edge information still remains even after applying the Laplacian mask, as shown in Fig.5(c). These edges will act as noise and will result in overestimation. It is therefore desirable to further filter the image in order to remove these edges as well. For this, we have used Sobel operator. After obtaining the edge map, we suppress the edge pixels from output of Laplacian convolution. The resultant image is now assumed to contain all the noisy pixels as shown in Fig.5 (d).

(a) **(b)**



(c) **(d)**

Fig 5: (a)original image (b) Noisy image with σ=5 (c) Output of Laplacian convolution with edges highlighted (d) Output after suppressing edges from (c)

The detailed algorithm for calculating the noise standard deviation is summarized below:

STEP 1: Convolve noisy image *'N'* with Laplacian mask.

$$N_{lap} = |N * Mask| \quad (10)$$

Where,

N_{lap} is the laplacian convolved image

$Mask$ is given in (2)

'*' denotes convolution

STEP 2: Perform edge detection on noisy image *'N'* using Sobel edge Detector.

$$N_{gx} = |N * gx| \quad (11)$$

$$N_{gy} = |N * gy| \quad (12)$$

Where,

$$gx = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (13)$$

$$gy = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (14)$$

STEP 3: Suppress edges from Laplacian Convolved image to obtain noise only image(N_n). In order to suppress edges from N_{lap}image, apply threshold on N_{gx} and N_{gy} images.

The threshold can be obtained as:

$$T_{hx} = p \times N_{gx(max)} \quad (15)$$

$$T_{hy} = p \times N_{gy(max)} \quad (16)$$

Where,

$N_{gx(max)}$ is the maximum value of N_{gx}

$N_{gy(max)}$is the maximum value of N_{gy}

We have experimentally found value of 'p' as,

$$p = 0.1$$

STEP 4: Compute local standard deviation (11x11 blocks) of noise only image(N_n). Also compute median std_{med} of local standard deviation values as explained in previous section.

STEP 5: Compute noise standard deviation using (17) for additive noise and using (18) for multiplicative noise.

$$\sigma_n = (0.2833 * std_{med}) + 0.0155 \quad (17)$$

$$\sigma_n = \left(0.3101 * \frac{std_{med}}{mean\_val}\right) + 0.05 \quad (18)$$

Where,

std_{med} is the median of local standard deviations

$mean\_val$ is the mean of the entire image

4. RESULTS

The proposed algorithm (PA) is tested on a variety of images with varying noise variances.

We have depicted results on test images as shown in Fig.6.

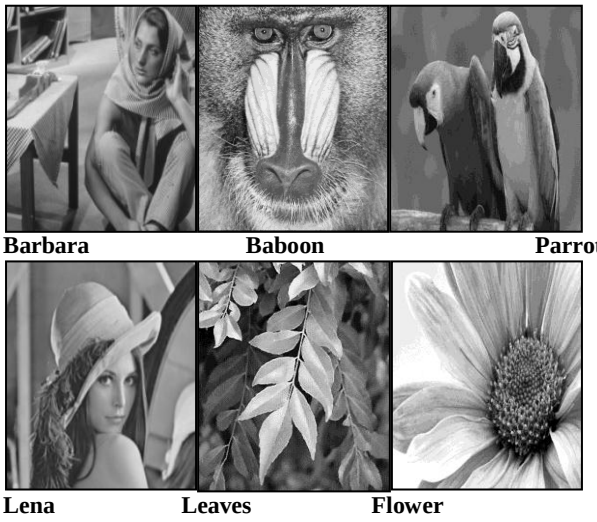


Fig 6: Test images

In order to evaluate the performance, we define the error metric (percentage estimation error) as:

$$\epsilon = \frac{abs(\sigma_{estimated} - \sigma_{original})}{\sigma_{original}} \times 100 \quad (19)$$

Where,

$\sigma_{original}$ is the input standard deviation value

$\sigma_{estimated}$ is the estimated value of standard deviation by proposed algorithm

4.1 Additive/Gaussian Noise

The proposed algorithm is compared with noise estimation algorithms by Rank et al [5], J. Immerkaer [6] and Tai et al [11].

Fig.7 and Fig.8 shows the performance comparison of algorithms presented in literature against our method in a graphical format. When the noise deviation is low, most of the algorithms tend to overestimate noise. The noise estimation becomes more erroneous when the image inherently contains fine details (Baboon), as shown in Fig.7 or if the image has textured regions (Barbara), as shown in Fig.8.

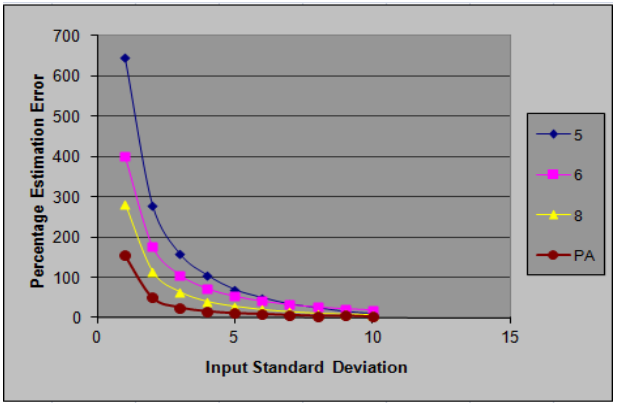


Fig 7: Performance comparison of [5], [6], [8] and PA for Baboon image

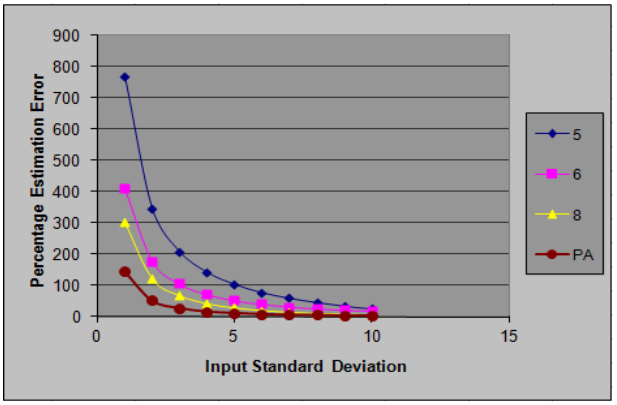


Fig 8: Performance comparison of [5], [6], [8] and PA for Barbara image

The algorithm described in [5] makes use of cascaded vertical and horizontal difference operator which does not suppresses complete original image structure. The algorithm described in [6] does not employ edge detection. It therefore overestimates noise because it considers strong edges that are left out after Laplacian convolution as noise. The algorithm described in [11] has achieved significant improvement over [6] by performing edge detection before Laplacian convolution to reduce their contribution to noise deviation.

However, the proposed algorithm estimates noise more accurately as compared to the other techniques, since it considers a windowed local standard deviation after suppressing the image structure and subsequently maps it on to transfer function that is obtained using linear regression on about 120 noisy images with known noise content. The proposed method is in contrary to [11], which estimates noise using simple arithmetic averaging.

Table 1-6 shows comparison of our proposed algorithm with other algorithms over range of noise deviation (standard deviation varied between (1 to 40)) for the test images. The results indicate that we have achieved significant improvement in estimation as compared to [5] and [6].

When compared with [11], our algorithm performs better for images which inherently contains fine details (Baboon) or if the image has textured regions (Barbara).

Table 1. Table .Comparison of Percentage Estimation Error of [5] and PA for Lena, Leaves and Flower Images

Input	Lena		Leaves		Flower	
Sigma	[5]	PA	[5]	PA	[5]	PA
1	246.0766	**90.61942**	414.7067	**135.1575**	146.9127	**16.44746**
2	91.18157	**28.38375**	175.5225	**49.79838**	48.85889	**7.751754**
3	44.73336	**15.38671**	97.98543	**25.21303**	15.92741	**4.036443**
4	22.01068	**7.8438**	59.90803	**15.54918**	1.386717	**2.149644**
5	7.748332	**3.127081**	38.48388	**10.88212**	6.714989	**2.056355**
10	11.81142	**0.562428**	1.810009	**2.331771**	20.9436	**0.031152**
15	9.102033	**0.047087**	14.345	**0.803266**	7.282224	**0.029218**
20	7.217195	**0.355755**	10.18791	**0.840695**	6.603097	**0.168274**
25	6.356624	**0.313539**	8.12894	**0.363798**	5.936832	**0.009656**
30	6.578002	**0.952072**	8.204516	**0.55519**	5.696142	**0.354247**
35	5.962036	**0.919199**	6.611554	**0.000623**	5.539308	**0.57942**
40	5.788263	**0.444797**	6.178112	**0.25175**	5.394642	**0.229642**

Table 2. Comparison of Percentage Estimation Error of [6] and PA for Lena, Leaves and Flower Images

Input	Lena		Leaves		Flower	
Sigma	[6]	PA	[6]	PA	[6]	PA
1	121.8111	**90.61942**	181.1906	**135.1575**	58.25452	**16.44746**
2	48.17914	**28.38375**	77.66575	**49.79838**	21.36461	**7.751754**
3	26.82657	**15.38671**	46.36538	**25.21303**	11.72606	**4.036443**
4	17.27163	**7.8438**	31.08359	**15.54918**	7.313163	**2.149644**
5	12.10757	**3.127081**	22.20889	**10.88212**	4.858636	**2.056355**
10	3.656405	**0.562428**	8.472363	**2.331771**	1.425793	**0.031152**
15	1.84218	**0.047087**	4.01898	**0.803266**	0.780602	**0.029218**
20	1.452171	**0.355755**	2.580883	**0.840695**	0.891377	**0.168274**
25	1.012447	**0.313539**	1.697151	**0.363798**	0.695642	**0.009656**
30	0.870218	**0.952072**	1.461313	**0.55519**	0.78265	**0.354247**
35	0.367477	**0.919199**	1.214785	**0.000623**	0.297564	**0.57942**
40	0.239807	**0.444797**	0.661409	**0.25175**	0.525374	**0.229642**

Table 3. Comparison of Percentage Estimation Error of [8] and PA for Lena, Leaves and Flower Images

Input	Lena		Leaves		Flower	
Sigma	[8]	PA	[8]	PA	[8]	PA
1	89.92697	**90.61942**	137.9139	**135.1575**	29.62207	**16.44746**
2	31.24417	**28.38375**	54.44179	**49.79838**	6.599154	**7.751754**
3	14.66121	**15.38671**	28.9007	**25.21303**	5.988749	**4.036443**
4	7.459863	**7.8438**	18.71167	**15.54918**	2.097235	**2.149644**
5	3.755893	**3.127081**	12.07837	**10.88212**	2.96269	**2.056355**
10	0.399039	**0.562428**	3.066724	**2.331771**	3.473988	**0.031152**
15	0.853554	**0.047087**	0.152831	**0.803266**	2.55488	**0.029218**
20	0.415108	**0.355755**	0.359542	**0.840695**	1.57366	**0.168274**
25	0.317968	**0.313539**	0.902565	**0.363798**	0.943094	**0.009656**
30	0.251242	**0.952072**	0.279332	**0.55519**	0.468132	**0.354247**
35	0.500283	**0.919199**	0.303236	**0.000623**	0.648495	**0.57942**
40	0.556762	**0.444797**	0.1234	**0.25175**	0.216702	**0.229642**

Table 4. Comparison of Percentage Estimation Error of [5] and PA for Barbara, Baboon and Parrot Images

Input	Barbara		Baboon		Parrot	
Sigma	[5]	PA	[5]	PA	[5]	PA
1	769.8471	**146.8865**	646.0693	**155.1575**	148.7154	**16.44746**
2	346.807	**52.15717**	278.8593	**49.79838**	51.13482	**5.751754**
3	209.4257	**28.46191**	157.6192	**25.21303**	20.25417	**1.036443**
4	143.4596	**17.09611**	104.6429	**15.54918**	5.755091	**1.149644**
5	104.7179	**12.67136**	68.93633	**10.88212**	5.052154	**1.056355**
10	25.98556	**3.913414**	9.26356	**2.331771**	19.80468	**0.031152**
15	30.16295	**0.903954**	25.55113	**0.803266**	7.144256	**0.029218**
20	19.9217	**0.531821**	17.17686	**0.840695**	6.253884	**0.168274**
25	14.43825	**0.624575**	13.33233	**0.363798**	5.891558	**0.009656**
30	11.7743	**1.989**	10.5492	**0.55519**	5.410982	**0.354247**
35	10.54998	**1.013714**	8.782948	**0.000623**	5.684262	**0.57942**
40	9.192806	**0.709705**	8.049578	**0.25175**	5.099526	**0.229642**

Table 5. Comparison of Percentage Estimation Error of [6] and PA for Barbara, Baboon and Parrot Images

Input	Barbara		Baboon		Parrot	
Sigma	[6]	PA	[6]	PA	[6]	PA
1	410.8644	**146.8865**	399.6944	**155.1575**	50.95108	**16.44746**
2	178.3732	**52.15717**	176.2082	**49.79838**	20.43818	**5.751754**
3	105.8167	**28.46191**	105.4272	**25.21303**	12.21382	**1.036443**
4	72.04735	**17.09611**	71.78469	**15.54918**	8.418802	**1.149644**
5	52.53264	**12.67136**	52.54316	**10.88212**	5.973641	**1.056355**
10	18.88556	**3.913414**	18.76859	**2.331771**	2.055965	**0.031152**
15	9.489261	**0.903954**	9.524114	**0.803266**	1.207521	**0.029218**
20	6.124627	**0.531821**	6.048058	**0.840695**	0.955541	**0.168274**
25	4.449004	**0.624575**	3.887225	**0.363798**	0.93931	**0.009656**
30	3.108202	**1.989**	2.652316	**0.55519**	0.430595	**0.354247**
35	2.621014	**1.013714**	1.922063	**0.000623**	0.621716	**0.57942**
40	1.435858	**0.709705**	1.539392	**0.25175**	0.161122	**0.229642**

Table 6. Comparison of Percentage Estimation Error of [8] and PA for Barbara, Baboon and Parrot Images

Input	Barbara		Baboon		Parrot	
Sigma	[8]	PA	[8]	PA	[8]	PA
1	303.5422	**146.8865**	281.8836	**155.1575**	24.47914	**16.44746**
2	122.6405	**52.15717**	114.7857	**49.79838**	5.604214	**5.751754**
3	68.78756	**28.46191**	63.71258	**25.21303**	1.876959	**1.036443**
4	43.01536	**17.09611**	39.91468	**15.54918**	1.270465	**1.149644**
5	29.38696	**12.67136**	27.67495	**10.88212**	0.832711	**1.056355**
10	6.786776	**3.913414**	6.883378	**2.331771**	1.540141	**0.031152**
15	1.992469	**0.903954**	2.962148	**0.803266**	1.169223	**0.029218**
20	1.343108	**0.531821**	1.856242	**0.840695**	0.76375	**0.168274**
25	1.164786	**0.624575**	1.056816	**0.363798**	0.416216	**0.009656**
30	1.117793	**1.989**	0.766227	**0.55519**	0.667554	**0.354247**
35	1.415842	**1.013714**	0.586011	**0.000623**	0.288644	**0.57942**
40	0.760923	**0.709705**	0.50759	**0.25175**	0.612469	**0.229642**

We have also tested our algorithm on image with poor contrast and image corrupted by motion blurr. The test images are as shown in Fig. 9.

(a) **(b)**

Fig 9: (a) Low contrast image (b) Motion blurred image

Table 7 shows the results of proposed algorithm on Fig. 9(a) and Fig. 9(b) .Numbers in the table are indicative of robustness of the proposed algorithm.

Table 7. Percentage Estimation Error of PA for low contrast and motion blurred image

Input Sigma	Low Contrast	Motion Blurred
1	55.15902	43.45009
2	47.04004	11.32496
3	39.51478	4.840586
4	24.60847	2.914138
5	18.55516	1.209949
10	6.272656	0.113518
15	3.031616	0.880237
20	1.4435	0.251628
25	0.891858	0.591109
30	0.42945	0.793
35	0.789072	0.591699
40	0.43075	0.4685

4.2 Multiplicative/Speckle Noise

Table 8 shows percentage estimation errors for test images mentioned earlier when contaminated with speckle noise of varying sigma. Numerical values indicate that the proposed algorithms can directly be used to estimate speckle noise accurately.

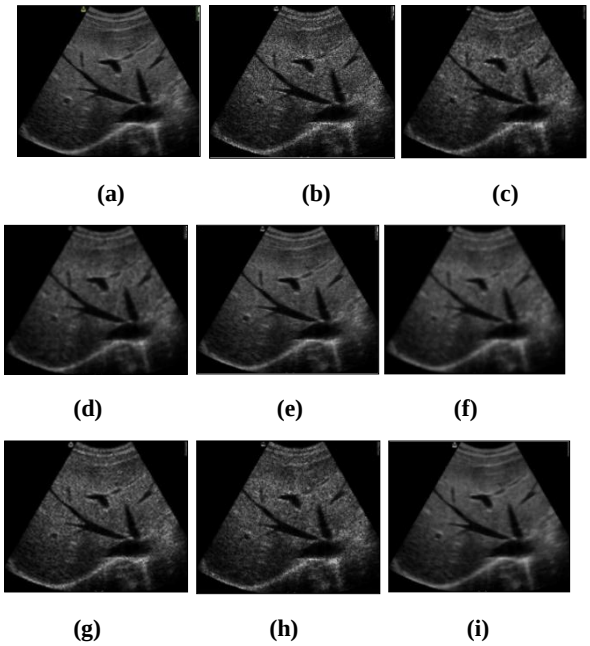
Table 8. Percentage Estimation Error of PA for Lena, leaves, Flower, Barbara, Baboon and Parrot Images contaminated with speckle noise

Input Sigma	Lena	Leaves	Flower	Barbara	Baboon	parrot
1	5.120569	3.904314	4.782746	5.537384	3.415984	5.568478
2	1.338299	0.172472	1.766814	1.058733	0.082526	2.41383
3	0.903992	0.272407	0.554344	1.138297	0.39536	1.715693
4	0.653649	1.381287	0.580278	0.730151	0.60695	1.799519
5	0.066056	0.357709	0.069406	0.001824	1.761497	1.136971
6	0.259513	1.536425	0.598546	0.511529	1.771299	1.559265
7	0.338274	1.123878	0.749356	0.667131	2.068638	0.769614
8	0.038073	1.054579	0.093119	0.660432	1.032009	1.415282
9	0.617181	0.578122	0.158542	0.438396	2.033482	0.660947
10	0.270584	0.917614	1.525212	0.353528	2.014808	0.380595

4.3 Noise Verification of Proposed Algorithm on Medical Ultrasound Images

Our algorithm can also be used for performance comparison of noise removal algorithms quantitatively. The well known error metrics e.g. RMSE, PSNR, UQI and SSIM require original image for reference. However, the proposed algorithm does not require reference image and can give noise content which actually matches with visual comparison.

Banazier et al [12] has proposed algorithm for speckle noise reduction from clinical ultrasound images. Fig. 9 shows the results of different de-speckle filters [12].



(a) **(b)** **(c)**

(d) **(e)** **(f)**

(g) **(h)** **(i)**

Fig 9: (a) Ultrasound image (liver image) (b) Noisy image(σ=5) (c) Median filter output (d) Speckle reducing anisotropic diffusion filter output (e) Geometric de-speckle filter output (f) Mean and variance local statistics de-speckle filter output (g) Wavelet Shrinkage filter output (h) Total Variation de-speckle filter output (i) Output of method proposed in [9]

Table 9 shows speckle noise estimation for images shown in Fig. 9.

Table 9. Estimation of speckle noise content for images shown in Fig.9

Image	Description	Estimated Speckle Noise
Fig.9(a)	Ultrasound image (liver image)	**0.1153**
Fig.9(b)	Noisy image	**0.5167**
Fig.9(c)	Median filter output	**0.1101**
Fig.9(d)	Speckle reducing anisotropic diffusion filter output	**0.0605**
Fig.9(e)	Geometric de-speckle filter output	**0.1326**
Fig.9(f)	Mean and variance local statistics de-speckle filter output	**0.059**
Fig.9(g)	Wavelet Shrinkage filter output	**0.1226**
Fig.9(h)	Total Variation de-speckle filter output	**0.2542**
Fig.9(i)	Output of method proposed in [9]	**0.0638**

As given in above table, speckle noise content is highest for Fig.9 (b) which is quite obvious as it is noisy image. Fig.9 (f) has lowest estimated value; this is due to over smoothing of image. Values for Fig.9 (d) and Fig.9 (i) indicates that these algorithms remove noise while preserving image details.

5. CONCLUSION

In this paper, a robust algorithm for estimation of noise standard deviation from images is proposed. This method estimates noise based on transfer function which is obtained experimentally from three source images on a-priori basis. The algorithm performs well for images with wide range of

noise deviation. It also gives encouraging results for noisy images that have inherent high frequency (fine details) or textured regions in it. We have reduced noise estimation error approximately by 50% for such images. The proposed algorithm is robust enough to estimate noise from poor contrast or blurred images with estimation error of 6% or less for noise deviation of 10 or more. The same algorithm can be used without any modification for estimating speckle noise as well. Results for speckle noise estimation indicate that proposed algorithm performs equally well for multiplicative noise with estimation error of 5% or less. The proposed algorithm can also be used to quantify noise content of an image and hence can be used as metric to compare performance of different noise removal algorithm. Owing to the repetitive nature of windowing and calculating local standard deviation thereof, this method can be further exploited for parallelization and hardware implementation as well.

6. REFERENCES

[1] Gonzalez and Woods, Digital Image Processing, Prentice Hall, 3rd edition, 2004.

[2] J. S. Lee, "Digital Image Smoothing and the Sigma Filter", Computer Graphics Image Processing, Vol. 24, No.2, pp. 255-269, Nov.1983.

[3] D. Barash,"A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation", IEEE Trans. Pattern Anal. Machine Intel, Vol. 24, No.6, pp. 844-847, June 2002.

[4] S. I. Olsen, "Estimation of noise in images: An evaluation", Computer Vision Graphics Image Process. Graphic Models and Image Process, Vol.55, No. 4, pp. 319-323, 1993.

[5] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," Proc. IEE Vis. Image Signal Processing, Vol. 146, No. 2, pp.80-84, Apr. 1999.

[6] J. Immerkær, "Fast Noise Variance Estimation", Computer Vision and Image Understanding, Vol. 64, No. 2, pp. 300-302, Sep. 1996.

[7] R. C. Bilcu and M. Vehvilainen, A New Method for Noise Estimation in Images, Proc. IEEE EURASIP International Workshop on Nonlinear Signal and Image Processing, Sapporo, Japan, May 18-20, 2005.

[8] A. Bosco, K. Findlater, S. Battiato, A. Castorina – " A Noise Reduction Filter for Full-Frame Imaging Devices" – IEEE Transactions on Consumer Electronics – Vol. 49, Issue 3, August 2003.

[9] A. Amer, A. Mitiche, and E. Dubois, "Reliable and Fast Structure-Oriented Video Noise Estimation," in Proc. IEEE Int. Conf. Image Processing, Montreal, Quebec, Canada, Sep. 2002.

[10] D.-H. Shin, R.-H Park, S. Yang, J.-H. Jung, Block-Based Noise Estimation Using Adaptive Gaussian Filtering, in IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, February, 2005.

[11] Shen-Chuan Tai and Shih-Ming Yang, A Fast Method for Image Noise Estimation Using Laplacian Operator and Adaptive Edge Detection, ISCCSP 2008, pp.1077-1081, March 12-14 2008.

[12] Banazier A. Abrahim, Yasser Kadah, Speckle noise reduction method combining total variation and wavelet shrinkage for clinical ultrasound imaging, Middle East conference on Biomedical Engineering (MECBME) , pp.80-83, Feb. 2011.