Fuzzy Edge Preserving Smoothing Filter Using Robust Region Growing

Amirshahed Mehrtash, *Student Member, IEEE*, Shahab Vahdat, and Hamid Soltanian-Zadeh, *Senior Member, IEEE*

Abstract—**Smoothing, while preserving edges, has always been a major challenge in image processing. In this paper, we propose a new approach that uses segmentation in order to avoid inter-region smoothing thus preserving the edges. It is common to smooth the image prior to region growing. The opposite procedure does not work properly in the presence of noise since region growing is very noise sensitive. To overcome this difficulty we adapted a robust region growing algorithm. Since region growing is very resource consuming, we do not perform it for every pixel. Instead, we divide the image into a number of overlapping blocks for which we carry out the segmentation. Then, we use the results for some of the core pixels in that block. Now that the regions are known, we proceed to smooth each region by a fuzzy approach emphasizing the pixels residing in it. The performance of this filter is compared to that of the bilateral filter and it is shown that the new method generates superior results.**

I. Introduction

Many applications require the image to be smoothed in the first stage. The very idea behind smoothing is to make pixels similar to their surroundings by using the information of other pixels in the vicinity. One simple method is to replace each pixel by a weighted average of its neighboring pixels. Although this method can reduce noise and smooth the image, it tends to blur the edges. Many techniques have been proposed to preserve the edges while smoothing [1]-[17]. In the following, we briefly describe some of the most important methods related to our work.

By solving partial differential equations in anisotropic diffusion [11], image is smoothed locally but averaging across the edges is prohibited. This iterative approach tends to promote intra-region smoothing while preventing inter-region smoothing. At first the structure of the regions is unknown (If the true positions of the edges were known the problem would already be solved.) but the current best estimate of the boundaries gets more and more precise with each iteration. The very idea of supporting smoothing within a region and penalizing smoothing across the edges also exists in our method but the way we construct an estimate of the boundaries is rather different. In anisotropic diffusion the estimate of the region boundaries is achieved iteratively throughout the whole process while in our method we do it in one run with a robust region growing algorithm. For an improved version of anisotropic diffusion see [21].

Median filtering is a simple and effective method for noise (especially speckle) reduction that replaces a pixel with the median of the pixels in a local window. This filter usually faces problems around corners or features that are less than half width of the filter window. Topological median filtering [16] uses fuzzy connectedness of pixels to enhance the performance of the median filter, using four filters: TD, TL, TDL, and TLD. The drawback of this approach is that for light features on dark background TD or TLD but for dark features on light background TL or TDL should be used. This filter and our filter share the use of the topology of the image for filtering but for that, topological median filter uses the concept of α-connectivity. Moreover, in our filter we replace a pixel with a weighted average of its neighbors while in topological median filter the pixel is replaced by one of the pixels in the vicinity (sometimes the median of them).

Bilateral filter proposed by Tomasi and Manduchi [14] is a non-iterative filter that combines pixel intensities based on both their geometric closeness and their photometric similarity. In bilateral filter each pixel is replaced by a weighted average of some neighboring pixels. These weights are generated by multiplying two kernels: one based on the geometric closeness of the pixels and one based on the similarity of their intensities. The further a pixel from the center of the mask and the more different its intensity from that of the central pixel, the lower its weight in averaging. It is common to use two Gaussian kernels for range and similarity functions in bilateral filters. We too use two kernels in our filter. The range kernel is like that of the bilateral filter but the similarity kernel is constructed rather differently. Although we used a Gaussian kernel for this cause, the different definition of similarity is the origin of this difference.

Noise changes the intensity of the pixels. Therefore, basing the similarity of the pixels only on their intensities would be prone to errors. Instead, we compute the similarity of the region each pixel resides in to the central pixel and assume that all the pixels in that region have the same similarity value. We will give more details on this later.

Elad [20] illustrated the theoretical connection of the bilateral filter to the Bayesian approach with a novel penalty function. He also suggested some improvements for the bilateral filter. Another filter which is based on bilateral filter is trilateral filter [15].

The new filter we propose here is based on segmentation.

First, we segment the image using a robust region growing scheme then we use a Gaussian mask for smoothing but in order to avoid averaging across the edges we multiply our Gaussian mask by another mask built based on the similarity of the regions.

In Section II, we describe our method for region growing which is robust to noise. Section III discusses how we use the result of region growing for fuzzy segmentation and then smoothing the image with emphasis on intra-region information. Experimental results are presented in Section IV and conclusions in Section V.

II. Robust Region Growing

For being robust, our algorithm checks two sets of conditions; if at least one of them holds, the investigated pixel is added to the region.

The expansion of the region is done in four directions. In other words, starting from a pixel, we check four pixels: right, left, upward, and downward pixels. If the pixel from which we want to expand the region is *x,* we denote these four neighbors by: x^{Left}, x^{Right}, x^{Up}, and x^{Down}, respectively. For brevity, we define x^D as the neighbor pixel in direction *D*, where *D* can be left, right, up, or down. x^{DD} is the next pixel in the same direction. x_{SP} is the seed point pixel in region *R*.

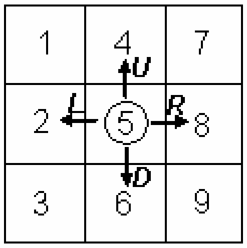


Fig. 1. Directions of expansion for the central pixel.

A. First Condition

If the gray value of a neighbor pixel, $I(x^D)$, and the gray value of the next pixel in the direction of expansion, $I(x^{DD})$, are less than a threshold different from the original pixel, the investigated pixel is added to the region:

$$\left|I(x_{SP})-I(x^D)\right| \leq thresh) \ \&(\ \left|I(x_{SP})-I(x^{DD})\right| \leq thresh$$

The threshold is calculated based on the local variance of the image. Due to our simulations, for best results, the threshold should be between σ/2 and σ/4.

This condition prevents a noisy pixel to be considered in the region. The only fallacious situation is when noise affects the investigated pixel and its neighbor in the direction of expansion in a way that the intensity of both moves into the vicinity of the seed point's intensity. This situation is highly unlikely considering pixels' noise being uncorrelated.

While this condition guards against intruders, it may not count some correct pixels in the region, especially when the noise level is high or when we are examining edge pixels. To avoid this problem, we introduce the second condition.

B. Second Condition

The investigated pixel ($I(x^D)$) is joined to the region if the similarity of the original pixel ($I(x)$) to its neighborhood, is more than the similarity of the next pixel in the direction of expansion ($I(x^{DD})$) to its neighborhood. The neighborhood (*N*) of each pixel, as shown in Figure 2, is defined as a six pixel rectangle that contains that pixel. To find the similarity of a pixel to a neighborhood, we calculate the sum of absolute difference between the pixel's intensity and the intensity of the neighboring pixel. The smaller this number, the higher the similarity.

For example, in Figure 3, we want to find out if pixel 5 belongs to the same region as pixel 6. The direction of expansion is upwards. We investigate the similarity of pixel 6 to its neighborhood pixels (2, 5, 8, 3 and 9) and pixel 4 to its neighborhood (1, 7, 2, 5 and 8). Pixel 5 joins the region if:

$$\sum_{i=2,3,5,8,9}\left|I(i)-I(6)\right| < \sum_{i=1,2,5,7,8}\left|I(i)-I(4)\right|$$

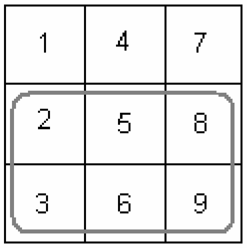


Fig. 2. Neighborhood of pixel 6 when the direction of expansion is upwards from 6 to 5. When expanding in other directions just rotate the figure.

Near the corners, this procedure may face problems. Therefore, we should emphasize the effect of the investigated pixel in the summation. To do this, we assign to the investigated pixel, the weight *n* which is greater than 1. Our simulations showed that for best result, *n* should be 2 or 3.

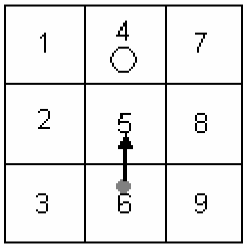


Fig. 3. The investigated pixel is pixel 5; we reach this pixel from pixel 6 with upward direction of expansion. We compare the similarity of pixels 6 and 4 to their neighbors.

Performing region growing takes a lot of time, especially when the window size is large. To achieve a reasonable delay for our filter we do not carry out region growing for every pixel we want to smooth. Instead, we divide the image into some overlapping blocks and run the region growing algorithm once for each block (Here we choose the blocks to be squares and we refer to their sizes as window size of the filter). By this approach, we obtain the region structure of each block, so we can proceed to smooth the pixels inside it with its region information. As for the pixels residing near the corners or ends of each block, the result of smoothing would not be accurate, since we can only use the information of neighboring pixels of one side of them (The pixels on the other sides have been outside the block and were not considered when region growing). To overcome this undesired effect, we only smooth some of the central pixels of each block, the *core* pixels, and leave the other pixels untouched at this juncture (Again, here we use a square core and call its size the *core size* of the filter). The pixels that were not smoothed in a specific block will be smoothed in an overlapping neighboring block. To smooth all pixels in the image, the overlap of two blocks should be equal to window size minus core size so that each pixel belongs to the core of one block (Figure 4). In other words, with this condition, the cores of the overlapping blocks partition the image.

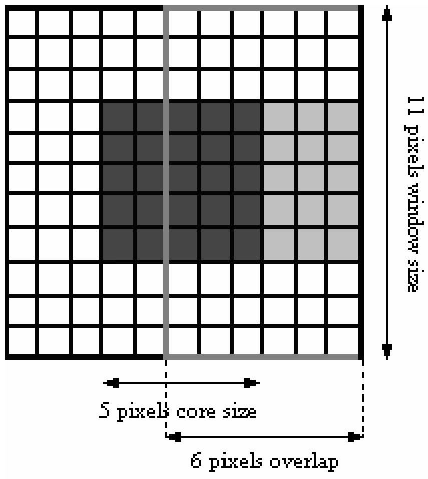


Fig. 4. One block of a filter with window size of 11 and core size of 5 pixels. Region growing is performed on all the 121 pixels but only the dark gray pixels are smoothed at this step, using all and nothing but the 121 pixels in this block. The remaining pixels in this block are smoothed when we are treating the neighboring blocks; for example the light gray pixels belong to the core of the block to the right (Half of its outline is painted in light gray).

III. Fuzzy Smoothing

Now that we have segmented the image, we can smooth it by only using the pixels in each region for its members. For example, we can use a Gaussian mask but set the weight of pixels outside the region to zero. This method not only preserves the edges but also exaggerates them. However, by using fewer elements in smoothing, we may not be able to reduce the noise very well. Thus, a better approach would be to use the neighboring regions in smoothing the intended region as well.

A. Fuzzy Segmentation

A classical set wholly includes or wholly excludes an element; the choice is binary. In fuzzy sets, membership becomes a matter of degree. To be more precise, in fuzzy logic, we map each member to a value in the interval [0,1] that defines its degree of membership (DOM). Zero meaning that the element is not a member, one defining absolute membership and a value in between denotes partial membership. The function used for this is called a membership function.

B. Fuzzifying the Result and Smoothing

Region growing gives us a number of classical sets but we can use this information to build fuzzy sets for our cause. In our study, we used the mean gray value of pixels in a region to define the DOM of its elements in a set so all pixels in a region have the same DOM. Let S be the classical set of pixels which are in the region we want to smooth and S' a classical set of pixels in another region which ζ' is a member of. Also, let the mean gray value of the pixels in S and S' be m and m', respectively. Now we define a fuzzy set F for region indicated by classical set S. This fuzzy set has a membership function denoted by μ_F.

$$\mu_F(\zeta') = \exp(-\frac{(m-m')^2}{2\sigma_f}) \qquad (1)$$

Here, σ_f is a parameter that defines the sharpness of the membership function.

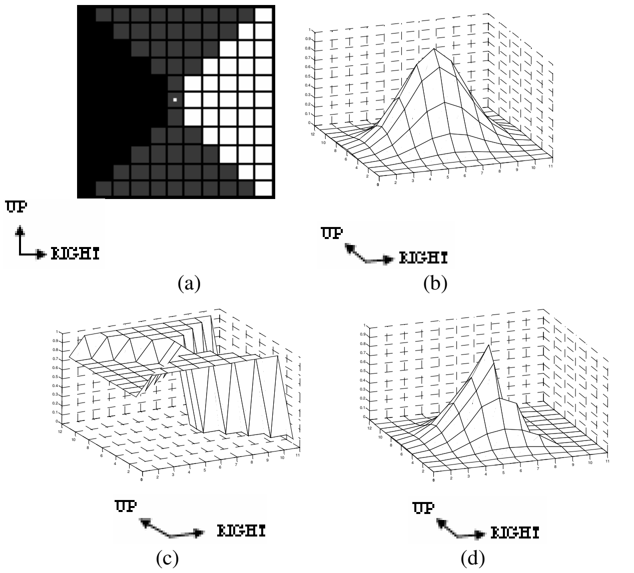


Fig. 5. (a) a block of image containing three regions. The weights are to be computed for finding the intensity of the pixel with the white dot in the smoothed image; (b) distance weights of neighboring pixels for computing the output value of the central pixel; (c) degree of membership of each pixel in the set for the central pixel; (d) final weights by multiplying distance weights by the corresponding degree of memberships.

The DOM of a pixel in S is 1 for the fuzzy set F because a pixel is an absolute member of the region it resides in. The more alike a region is to our intended region, the higher the DOM of its members.

To find the value of a pixel in the smoothed image, we first have to build the membership function for the region containing that pixel. The same membership function can be used for all of the pixels in a region. The output of the filter is attained by the following formula:

$$I_{out}(x) = \frac{1}{k(x)} \sum_{\zeta \in N_x} I_{in}(\zeta) c(x,\zeta) \mu_{F_x}(\zeta) \quad (3)$$

Here, $I_{out}(x)$ is the intensity value of pixel x in the output smoothed image, $I_{in}(\zeta)$ is the intensity value of pixel ζ in the input image, and $c(x,\zeta)$ is the domain kernel that designates weights to the pixels based on their positional distance from the intended pixel (Geometrical distance between x and ζ). It is common to choose a Gaussian function with standard deviation σ_c for this function. μ_{Fx} is the membership function of the fuzzy set F_x which is an extension of the classical set that includes pixel x. N_x is a neighborhood of pixel x. In our work, we defined it as the whole block in which region growing is performed. To maintain the original image's intensity, we normalize the sum by:

$$k(x) = \sum_{\zeta \in N_x} c(x,\zeta) \mu_{F_x}(\zeta) \quad (2)$$

An example of the weights of the neighboring pixels for calculating the output value of the central pixel (marked by a white dot) is shown in Figure 4.

IV. Experimental Results

In this section, we evaluate the performance of the proposed filter by evaluating the bias and variance of the output image and its quality. We also compare the proposed filter to bilateral, wiener, and median filters.

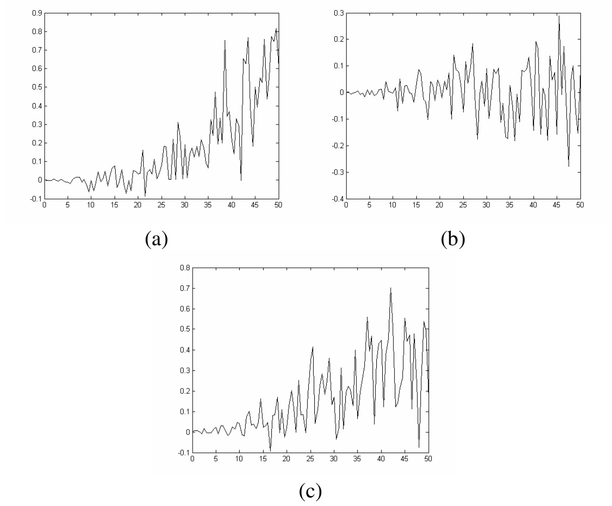


Fig. 6. Mean of output image versus input image standard deviation for filter with window and core size of (a) [7,3]; (b) [9,5]; (c)[11,7] .

A. Bias

First, we investigate the response of our filter to Gaussian variables. To this end, we build a number of 100×100 images composed of pure Gaussian white noise with the mean of 128 and different variances, 100 images for each. Then we pass these images through our filter and average over the outputs intensity.

For investigating the filter's bias, the mean intensity of the output image versus the standard deviation of the input image is plotted for different window and core sizes. The filter's bias is less than 1 (for window and core size of 9 and 5, it is less than 0.3) so it is absolutely negligible.

B. Standard Deviation

Figure 7 shows the standard deviation of the output image versus that of the input image. As this figure shows there is a linear relation between them. The slope of the interpolated lines are 0.40025, 0.34838, and 0.37525 for window and core size of [7,3], [9,5], and [11,7], respectively; σ_f and σ_d are both 1.

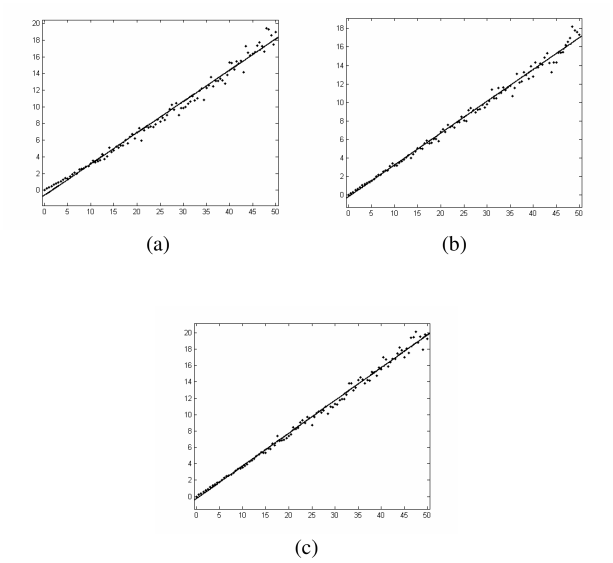
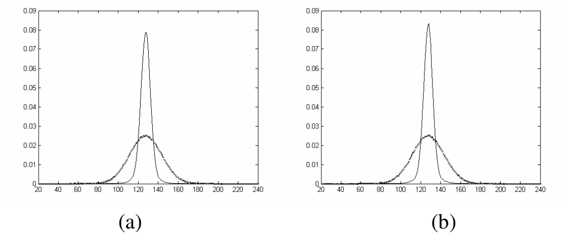


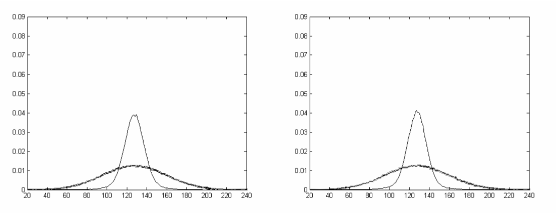
Fig. 7. Standard deviation of output image versus standard deviation of input image for a filter with window and core size of (a) [7,3]; (b) [9,5]; (c) [11,7] ; averaged over 100 image for 101 points.

We achieve maximum smoothing when we use a 9×9 window and 5×5 core. Increasing the core size and window size beyond that will result in more computations and less smoothing.

C. Histogram Analysis

Figure 8 shows the normalized histogram of the output images for different window sizes and various input standard deviations. As it can be seen, the response is symmetric averaged around 128, which means the filter does not change the image intensity (σf and σd are both 1).





(c) (d)

Fig. 8. Histogram of input (dotted line) and output (solid line) for input Gaussian noise of 128 mean and standard deviation of (a) 16 for window and core size of [9,5] (b) 16 for window and core size of [11,7] (c) 32 for window and core size of [9,5] (d) 32 for window and core size of [11,7].

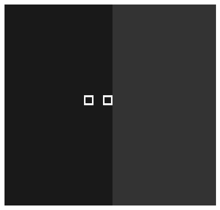
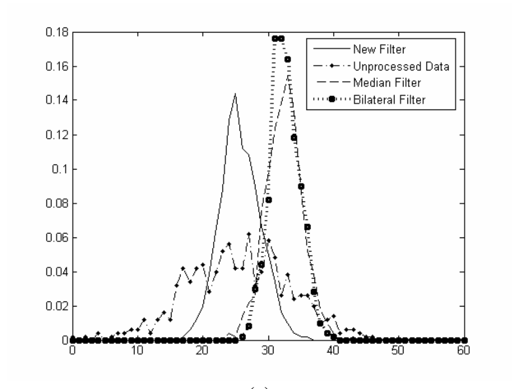
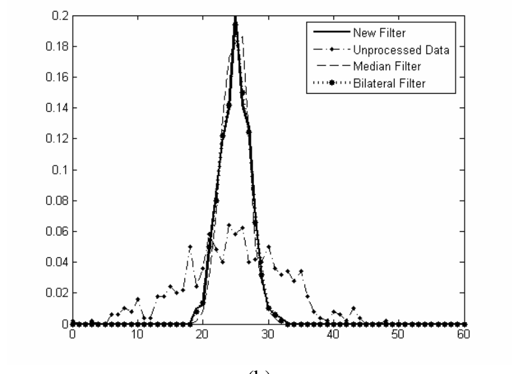


Fig. 9. Original image used for examining the edge preserving characteristics of our filter. One pixel on the edge and one two pixels away from the edge are marked.



(a)

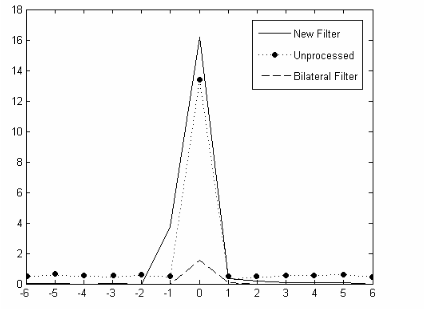


(b)

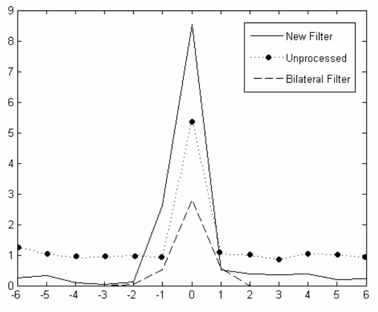
Fig. 10. The histogram of a pixel (a) on the edge; (b) two pixels far from the edge.

D. Edge Preserving Characteristics

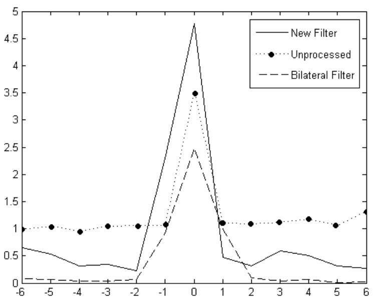
The most important characteristic of this filter is its preserving edges. We built an image with two areas one with intensity of 50 and another with 25 (Fig. 8). We added noise to this image and constructed 500 images with noise standard deviation of 16 and 500 images with 32. We then passed these images through our filter and investigated two pixels, one on the edge and another two pixels from the edge in the region with intensity 25. The histogram of this pixel intensity after passing through median, bilateral, and our new filter is plotted in Figure 10.



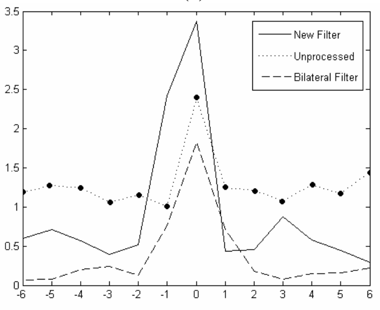
(a)



(b)



(c)



(d)

Fig. 11. Statistics of Sobel edge detector's output to an image with additive Gaussian noise with: (a) σ=25; (b) σ=50; (c) σ=75; (d) σ=100.

The histogram should be symmetric around 25. As Figure 11 illustrates, in the case of the pixel near the edge but not on it, all filters are almost the same with median filter having a slightly better performance. However, for the pixels on the edge, our new filter gives the best performance and is the only filter whose output histogram averages around 25.

In another set of simulations, we added white Gaussian noise to Figure 9, filtered it, and determined the edge pixels by a Sobel edge detector. The number of pixels determined as edge pixels versus their offset from the edge for unprocessed noisy image, image filtered with bilateral, and image filtered with our new filter are plotted in Figure 11. Each edge consists of two sides, therefore, in the result of the edge detector, the two pixels on each side of the edge are considered as one pixel. The window and core size for the new filter are 9 and 5 and the window size of the bilateral filter is 5. For both filters, the σ's of the Gaussian kernels are 2. The results are averaged over 100 images. They show the superiority of the proposed method.

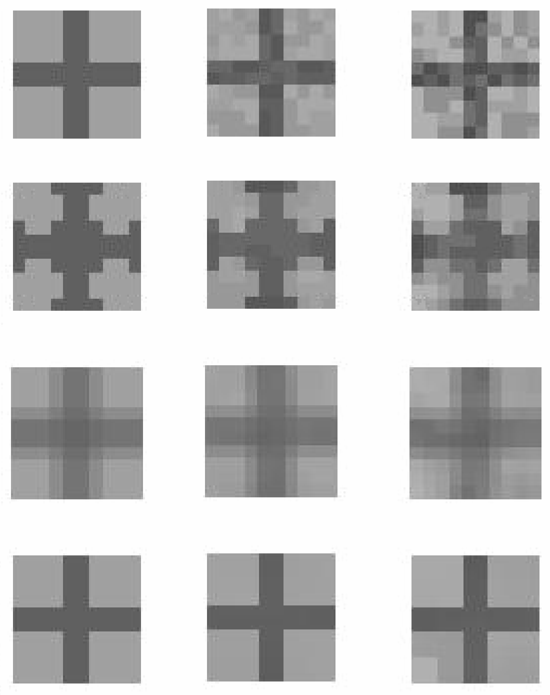


Fig. 12. First row is the input image, second row is the output of median filter, third row is the output of bilateral filter and the fourth row is the output of our new filter. In the first column, no noise is added, in the second column noise is {-1 0 1}, and in the third column noise is {-2 0 2}.

Figure 12 shows a cross with the width of two pixels. The gray values are normalized to 12. Intensity of the background is 10 and that of the cross is 2. Additive noise is chosen uniformly from the set {-1,0,1} for the second column and {-2,0,2} for the third column. The output of our filter is compared to wiener and median filters, showing its superiority.

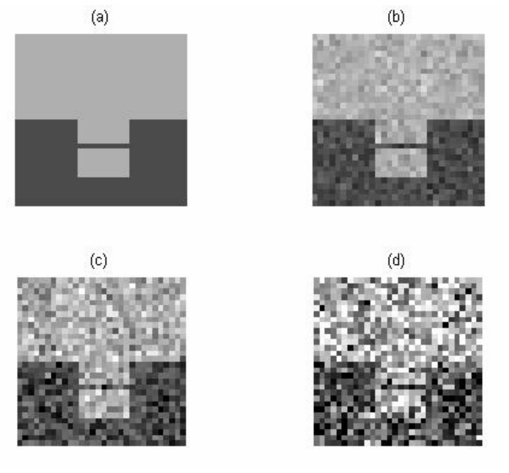


Fig. 13. (a) Original image; (b) Additive white Gaussian noise σ=16; (c) Additive white Gaussian noise σ=32; (d) Additive white Gaussian noise σ=64.

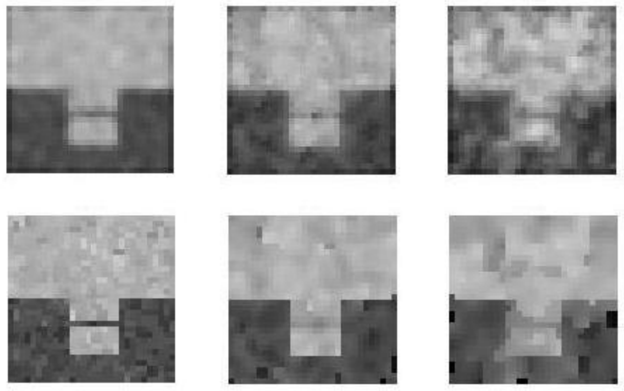


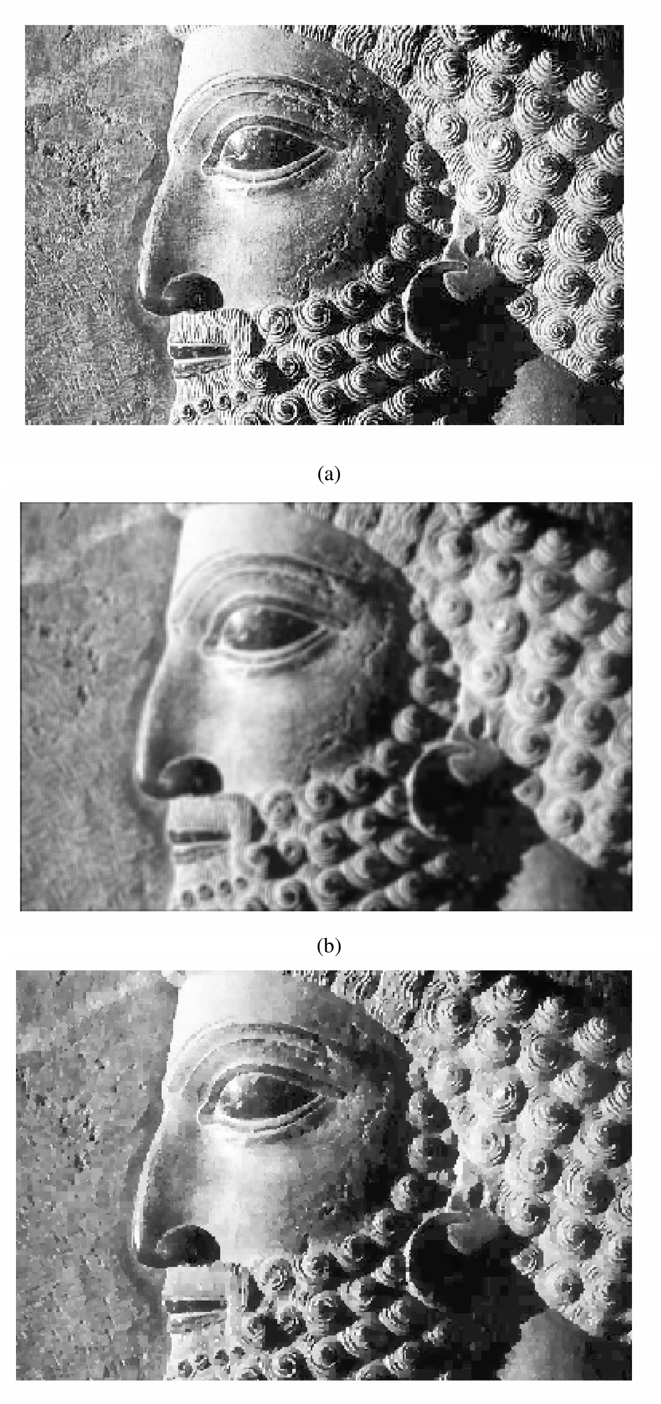
Fig. 14. First row is the output of bilateral filter and the second row output of our new filter. Image noise are 16, 24 and 32 for first, second, and third columns, respectively.

Our filter can delicately preserve corners and narrow features which are more than 1 pixel wide (Figs. 12-13). The filter cannot distinguish between two features that are as close as one pixel if the noise level is high (variance of additive noise is higher than 1.5 times that of the original image).

E. *Qualitative Smoothing Results*

When performed on noiseless images, the new filter can preserve the main edges and smooth each region independently. In order to prevent smoothing across the edges, here σ_f should not be large. Since the image's noise level is low, we do not need to use many pixels in our weighting to achieve an acceptable result.

Figure 15 shows an image smoothed using the new filter. Note that the main edges of the face are preserved while some small features like tiny fluctuations in the stone are smoothed.


(a)

(b)

(c)

Fig. 15. Persian soldier, Perspolis. Please note that the figure has undergone histogram equalization so the details be more noticable, Iran; (a) Original; (b) Smoothed image using bilateral filter with σ_c=2, σ_f=10 and window size of 5; (c) Smoothed Image by the new filter with window size of 9, core size of 5, σ_c=2, and σ_f=1.

F. Noise Cancellation

Our filter can also be used for noise cancellation. In contrast to smoothing, for this purpose the value of σ_f should be relatively high. In addition, for optimal results, we should increase the threshold of region growing. This difference in parameters is to make the region growing and smoothing more robust to noise. If the region growing conditions are set strict, we may exclude some noisy pixels from their original region. Besides, by using more pixels in our averaging, we will have a better diversity thus a better estimation of the pixels' original value. Some results of canceling the noise of Lena's image are brought in Figure 16. Each column corresponds to a different noise level.

Looking at Figure 16, we can observe that when the noise level is not very high, the new filter works better than the bilateral filter. But in images with a high noise level, the new filter cannot diminish the noise much more than bilateral and its edge preserving properties are only slightly better. This is because the region growing algorithm, though robust, does not work well when the image is very noisy. Therefore, it is recommended that highly noisy images are passed through this filter more than once or the noise level is diminished to some extent with another method prior to inputting the image to our new filter.

V. Conclusions

The proposed filter performs intra-region smoothing without mixing different regions. It also sharpens the boundaries. In addition, it does not add bias to the image and keeps the image mean intensity level intact, while reducing the image variance as a result of smoothing.

Acknowledgment

The authors should like to appreciate the helpful comments of anonymous reviewers that contributed to the strength and accuracy of this paper.

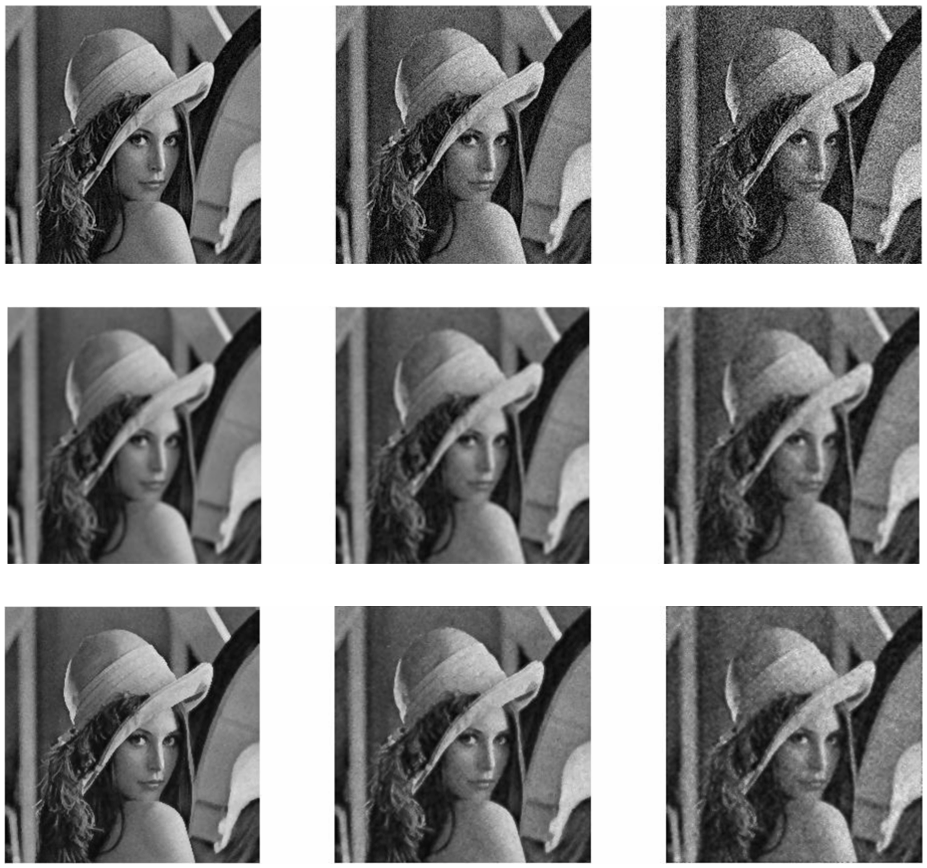


Fig. 16. First row shows the input images. Second row is the output of bilateral filter with window size of 5, σ_d=2, and σ_r=10. Third row is the output of the new filter with window size of 5 core size of 1, σ_d=2, and σ_f=10. The additive noise is Gaussian and in the first column its σ is 8, in second column 16 and in third column 32.

REFERENCES

[1] T. Boult, R. A. Melter, F. Skorina, and I. Stojmenovic. G-neighbors.*Proc. SPIE Conf. on Vision Geometry II*, 96–109, 1993.

[2] R. T. Chin and C. L. Yeh. Quantitative evaluation of some edgepreserving noise-smoothing techniques. *CVGIP*, 23:67–91, 1983.

[3] L. S. Davis and A. Rosenfeld. Noise cleaning by iterated local averaging. *IEEE Trans.*, SMC-8:705–710, 1978.

[4] R. E. Graham. Snow removal a noise-stripping process for picture signals. *IRE Trans.*, IT-8:129–144, 1961.

[5] N. Himayat and S.A. Kassam. Approximate performance analysis of edge preserving filters. *IEEE Trans.*, SP-41(9):2764–77, 1993.

[6] T. S. Huang, G. J. Yang, and G. Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Trans.*, ASSP-27(1):13–18, 1979.

[7] J. S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Trans.*, PAMI-2(2):165–168, 1980.

[8] M. Nagao and T. Matsuyama. Edge preserving smoothing. *CGIP*, 9:394–407, 1979.

[9] P. M. Narendra. A separable median filter for image noise smoothing. *IEEE Trans.*, PAMI-3(1):20–29, 1981.

[10] K. J. Overton and T. E. Weymouth. A noise reducing preprocessing algorithm. In *Proc. IEEE Computer Science Conf. on Pattern Recognition and Image Processing*, 498–507, 1979.

[11] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans.*, PAMI-12(7):629–639, 1990.

[12] G. Ramponi. A rational edge-preserving smoother. In *Proc. Int'l Conf. on Image Processing*, 1:151–154, 1995.

[13] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted median filters: a tutorial. *IEEE Trans.*, CAS-II-43(3):155–192, 1996.

[14] C. Tomasi, and R. Manduchi, Bilateral filtering of gray and colored images,. *Proc. IEEE Intl. Conference on Computer Vision*, pp. 836-846, 1998. P.

[15] Choudhury and J.Tumblin, The Trilateral Filter for High Contrast Images and Meshes, Eurographics Symposium on Rendering 2003, pp. 1-11.

[16] H. G .Senel, R. A. Peters II, and B. Dawant, Topological Median Filters, IEEE Trans. on image processing, vol. 10, No. 12, December 2001

[17] H. Soltanian-Zadeh, J. P. Windham, and A. E. Yagle, A Multidimensional Nonlinear Edge-Preserving Filter for Magnetic Resonance Image Restoration, IEEE Trans. on image processing, vol. 4, No. 2, February 1995

[18] P. Saint-Marc, J. S. Chen, and G. Medioni, Adaptive smoothing: A general root for early vision, in Proc. IEEE Comput. Soc. Conf. Computer .Vision Patt. Recogn., 1989, pp. 618424.

[19] F. Durand, and J. Dorsey, Fast bilateral filtering for the display of high-dynamic range images,. ACM Transactions on Graphics, *special issue on Proc. of ACM SIGGRAPH 2002*, San Antonio, Texas, vol. 21(3), pp. 249-256, 2002.

[20] M. Elad, .On the origin of bilateral filter and ways to improve it,. *IEEE Transaction Image Processing*, vol. 11(10), pp. 1141-1151, 2002.

[21] M. J. Black, G. Sapiro, D. Marimont, and D. Heeger, Robust anisotropic diffusion,. *IEEE Transactions on Image Processing*, vol. 7(3), pp. 421-432, 1998.

[22] Yin L., M. Gabbouj, and Y. Neuvo, Weighted median filters: a tutorial, IEEE Trans. Circ. Syst. II: Analog Dig. Signal Processing. vol. 43, no. 3, pp. 157-192, March 1996.

[23] Zadeh L., Fuzzy sets, Information and Control, vol. 8, pp. 338-353, 1965.