Enhancement of Medical Image using Spatial Optimized Filters and OpenMP Technology

Luis Cadena, Alexander Zotin, Franklin Cadena.

***Abstract* — For remove noise digital filters are used which in the last decades have taken great impetus for the treatment of images in different fields of science and in the case of the medical image processing better results are obtained in order to make better interpretations. Medical images can contain some noise therefore it makes sense to suppress noise on preprocessing stage. For the research we used digital filters such as mean filter, Gaussian filter and median filter. The speed performance comparison of optimized and classical implementations of these filters was conducted. It shows great speed improvement of optimized implementation.**

The possibilities of modern hardware allow using parallel technology for image processing. The parallel implementations of noise reduction filters algorithms taking into account the data parallelism were made utilizing OpenMP technology. The comparison of parallel implementation of filters algorithm with sequential and classic implementation show great increase in performance. Thus the parallel implementation gives a performance boost in 2-3 times for optimized version of algorithms.

***Index Terms* — medical image, image processing, denoising, Mean Filter, Median Filter, Gaussian 2D filter, parallel programing, OpenMP.**

I. INTRODUCTION

Image noise is random variation of brightness or color information in images, and is usually an aspect of electronic noise. It can be produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that adds spurious and extraneous information.

Medical images are often contaminated by impulsive, additive or multiplicative noise due to a number of non-idealities in the imaging process.

Filtering is a technique for modifying and enhancing an image. Various filters are used for image preprocessing. The primary purpose of these filters is a noise reduction, but filter can also be used to emphasize certain features of an image or remove other features. In image processing, 2D filtering techniques are usually considered an extension of 1D signal processing theory. For this work we used following filters: Classic Mean filter, Mean filter optimized, classical 2D Gauss filter and double 1D Gaussian filter, Classic Median filter, Median filter optimized.

Parallel processing is a process used to accelerate the execution time of a program by dividing it into multiple pieces that will be executed at the same time, each one on its own processors or core. The main reason for creating and using parallel computing is that parallelism is one of the best ways to bridge the bottleneck problem that signifies the speed of a single-core processor. The rationale for parallel processing is to speed up the resolution of a problem. Thus the acceleration that can be achieved depends on both the problem itself and the computer architecture. For the implementation of parallel algorithms widely spread OpenMP standard. The parallelization according to OpenMP standard is performed by inserting special directives into the text of the program, as well as call of support functions. OpenMP standard implements parallel processing using multithreading. Thus the "main" thread creates a set of subordinate threads, between which the task is distributed. The program is executed with sequential code – first running one process (thread), at the entrance to a parallel region are generated multiple threads, between which workload are distributed in the following code. If a forked thread completes its work before any other forked thread, it will block. Once all forked threads complete their work, the master thread then resumes execution. At the end of the parallel region, all threads except the "main", terminated, and starts another sequential region of a program. OpenMP standard also supports embedding of parallel regions. To improve the performance of image processing algorithms (Filters) is expedient to carry out parallelization for external loops (height of image).

Manuscript received December 08, 2017; this work was supported by Universidad de las Fuerzas Armadas ESPE, Av. Gral Ruminahui s/n, Sangolqui Ecuador.

L. Cadena, is with Electric and Electronic Department at Universidad de las Fuerzas Armadas ESPE, Av. Gral Ruminahui s/n, Sangolqui Ecuador. (phone: +593997221212; e-mails: ecuadorx@gmail.com).

A. Zotin is with Department of Informatics and Computer Techniques, Reshetnev Siberian State University of Science and Technology, 31 krasnoyarsky rabochy av., Krasnoyarsk 660037, Russian Federation (e-mail: zotinkrs@gmail.com)

F. Cadena is with College Juan Suarez Chacon, Quito, Ecuador (e-mail: fcfc041@gmail.com)

II. CLASSIC AND OPTIMIZED FILTERS

CLASSIC MEAN FILTER

The filter is a simple slidingwindow spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window [1-4, 6].

$$\mathbf{C}_{new}(y,x) = \frac{1}{K_{size}} \sum_{dy=-RH}^{RH} \sum_{dx=-RW}^{RW} \mathbf{C}_{old}(y+dy, x+dx),$$

where C_{new}, C_{old} – new and old values of the image pixels spectrum, respectively; RH, RW – constants defining the rank of the filter vertically and horizontally; K_{size} – kernel size is equals (2×RH+1)×(2×RW+1).

The mean filter can also be implemented as a convolution with coefficients $1/K_{zise}$.

The window, or kernel, is usually square but can be any shape. An example of mean filtering of a single 3×3 window

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of values is shown below.

unfiltered values

7	4	5
4	11	8
6	13	14

7 + 4 + 5 + 4 + 11 + 8 + 6 + 13 + 14 = 72 72 / 9 = 8

mean filtered

*	*	*
*	8	*
*	*	*

Center value previously 11 is replaced by the mean of all nine values, i.e. 8.

MEAN FILTER OPTIMIZED

The accumulation of the neighborhood of pixel P(y,x), shares a lot of pixels in common with the accumulation for pixel P(y,x+1). This means that there is no need to compute the whole kernel for all pixels except only the first pixel in each row. Successive pixel filter response values can be obtained with just an add and a subtract to the previous pixel filter response value [5, 7]. Thus, the filter computation can be considered the following way:

$$\mathbf{C}_{new}(y,x)=\begin{cases}\dfrac{1}{K_{Hs}\times K_{Ws}}\displaystyle\sum_{dy=-RH}^{RH}\sum_{dx=-RW}^{RW}\mathbf{C}_{old}(y+dy,x+dx), & if \quad x=0\\ \mathbf{C}_{new}(y,x-1)-\dfrac{1}{K_{Hs}}\displaystyle\sum_{dy=-RH}^{RH}\mathbf{C}_{old}(y+dy,x-RW-1)+ & \\ \quad+\dfrac{1}{K_{Hs}}\displaystyle\sum_{dy=-RH}^{RH}\mathbf{C}_{old}(y+dy,x+RW), & otherwise\end{cases}$$

where C_{new}, C_{old} are the new and old values of the image pixels spectrum, respectively; RH, RW are the constants defining the rank of the filter vertically and horizontally.

CLASSIC MEDIAN FILTER

The median filter is also a slidingwindow spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window [4].

$$\mathbf{C}_{new}(y,x)=\underset{(dy,dx)\in K_{xy}}{\text{med}}\{\mathbf{C}_{old}(dy,dx)\}$$

where C_{new}, C_{old} are the new and old values of the image pixels spectrum, respectively; K_{xy} is the kernel window.

Usually the kernel of median filter is usually square but it can take any shape. An example of median filtering of a single 3×3 window of values is shown below.

unfiltered values

7	4	5
4	11	8
6	13	14

in order: 4, 4, 5, 6, 7, 8, 11, 13, 14

median filtered

*	*	*
*	7	*
*	*	*

Center value previously 11 is replaced by the median of all nine values 4.

MEDIAN FILTER OPTIMIZED

In image processing the histogram of spectrum for median calculation can be far more efficient because it is simple to update the histogram from window to window, and finding the median of a histogram is not particularly onerous. Thus the histogram used for accumulating pixels in the kernel and only a part of it is modified when moving from one pixel to another [8, 12-13]. As illustrated in Figure 1, 2×RH+1 additions and 2×RH+1 subtractions need to be carried out for updating the histogram.

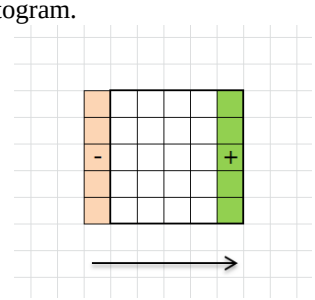


Figure 1. Median filter algorithm, 2RH+ 1 pixels must be added to and subtracted from the kernel's histogram when moving from one pixel to the next. In this figure, RH = 2.

CLASSIC GAUSSIAN 2D FILTER

The Gaussian filter uses a Gaussian function (which also expresses the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image. In two dimensions, it is the product of two such Gaussians, one in each dimension:

$$G(y,x)=\frac{1}{2\pi\cdot\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

Images during processing are usually represented as a collection of discrete pixels. Thus it is necessary to produce a discrete approximation to the Gaussian function before perform the convolution. Since the Gaussian distribution is theoretically non-zero everywhere, so the kernel would require an infinitely large size. We can say that depends on σ and kernel size some of non-zero coefficients, having very small values and do not have a significant effect on the result, can be outside of kernel. Thus it is possible to truncate the kernel. Usually kernel size equals 3σ, but in some cases Gaussian kernel can be truncated even more. After generation of Gaussian kernel with desirable size and σ, all coefficients must be corrected in the way that the sum of all coefficients equals 1. Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods.

IMPROVED GAUSSIAN FILTER 1D ×2

The convolution of Gaussian filter can be performed much faster since the equation for the 2D isotropic Gaussian is separable into y and x components [6, 9] Figure 2.

Gy =

0,004432
0,053991
0,241971
0,398942
0,241971
0,053991
0,004432

Gx =

0,004432	0,053991	0,241971	0,398942	0,241971	0,053991	0,004432

Gyx= Gy×Gx=

0,00002	0,00024	0,00107	0,00177	0,00107	0,00024	0,00002
0,00024	0,00292	0,01306	0,02154	0,01306	0,00292	0,00024
0,00107	0,01306	0,05855	0,09653	0,05855	0,01306	0,00107
0,00177	0,02154	0,09653	0,15915	0,09653	0,02154	0,00177
0,00107	0,01306	0,05855	0,09653	0,05855	0,01306	0,00107
0,00024	0,00292	0,01306	0,02154	0,01306	0,00292	0,00024
0,00002	0,00024	0,00107	0,00177	0,00107	0,00024	0,00002

Figure 2. Isotropic Gaussian is separable into Gy and Gx components.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Therefore to improve 2D Gaussian filter we propose to use two Gauss single-dimensional filters, which leads to a reduction in the number of processed elements (Table I). In some cases the approximation of Gaussian filter can be used instead of classic version [10, 11].

TABLE I.
REDUCTION OF PROCESSED ELEMENTS FOR GAUSS FILTER 1D ×2

Kernel Type	Kernel size	Kernel elements count	Reduction of processed elements
2D	5×5	25	in 2,5
1D ×2	5×1 & 1×5	5+5=10	
2D	7×7	49	in 3,5
1D ×2	7×1 & 1×7	7+7=14	
2D	9×9	81	in 4,5
1D ×2	9×1 & 1×9	9+9=18	
2D	11×11	121	in 5,5
1D ×2	11×1 & 1×11	11+11=22	

PARALLEL IMAGE PROCESSING

For the implementation of parallel algorithms widely spread OpenMP standard for the parallelization of programs in languages C, C ++ and Fortran [14-16]. The parallelization according to OpenMP standard is performed by inserting special directives into the text of the program, as well as call of support functions. OpenMP standard implements parallel processing using multithreading. Thus the "main" thread creates a set of subordinate threads, between which the task is distributed. The program is executed with sequential code – first running one process (thread), at the entrance to a parallel region are generated multiple threads, between which workload are distributed in the following code. If a forked thread completes its work before any other forked thread, it will block. Once all forked threads complete their work, the master thread then resumes execution. At the end of the parallel region, all threads except the "main", terminated, and starts another sequential region of a program. OpenMP standard also supports embedding of parallel regions.

To improve the performance of image processing algorithms (Filters) is expedient to carry out parallelization for external loops (height of image) – Loop level parallelism. To do this, in OpenMP can be used directive:

```
#pragma omp parallel for
```

It is offers a simple way to achieve loop-level parallelism, often existing in image processing algorithms (Data parallelism). The parallelization of loops is the most common use of OpenMP. Consider the following code, which represent a basis of any filter:

```
for (int y=0; y< Image_Height; y++)
  for (int x=0; x< Image_Width; x++)
   {
    // do some processing
   }
```

With OpenMP, this code can be parallelized according data parallelism in a simple way:

```
#pragma omp parallel for
for (int y=0; y< Image_Height; y++)
  for (int x=0; x< Image_Width; x++)
   {
    // do some processing
   }
```

Here, the directive instructs the compiler that the next for loop is to be parallelized. The compiler will then distribute the work among a set of forked threads. Thus we can divide image into lines which will be processed independently.

III. EXPERIMENTAL RESULTS

For our experimental study we used 50 medical images with different size. During experiments were conducted estimation of processing time of optimized filtering algorithms (Mean filter, Median filter, Gaussian Filter) and evaluation of noise suppression. For the experiment used PC based on Intel Core i5 3.1 GHz with 8 GB RAM.

At first we conducted a study that shows the advantage of optimized implementation of algorithms. Thus were generated map of acceleration with kernel changing in range 3×3 – 11×11 for each filters. This map shows acceleration coefficient which based on average processing time (200 calculations for each kernel size).

Kernel heigth \ Kernel width	3	5	7	9	11
3	1,41	1,97	2,53	3,12	3,68
5	1,56	2,16	2,93	3,63	4,32
7	1,57	2,35	3,11	3,84	4,56
9	1,60	2,39	3,17	3,93	4,68
11	1,61	2,42	3,24	4,06	4,89

(a)

Kernel heigth \ Kernel width	3	5	7	9	11
3	1,14	1,46	1,62	1,82	1,94
5	1,47	1,95	2,38	2,65	2,85
7	1,70	2,35	2,88	3,28	3,46
9	1,91	2,53	3,22	3,61	4,00
11	1,95	2,74	3,45	4,14	4,72

(b)

Kernel heigth \ Kernel width	3	5	7	9	11
3	4,40	7,60	11,21	14,71	19,22
5	5,28	9,24	13,91	18,16	22,67
7	5,88	10,62	15,29	20,09	25,93
9	6,09	10,98	15,98	21,99	27,59
11	6,66	11,37	17,48	23,39	29,11

(c)

Figure 3. Maps of acceleration of optimized filters implementation: (a) Mean filter; (b) Gaussian filter; (c) Median filter.

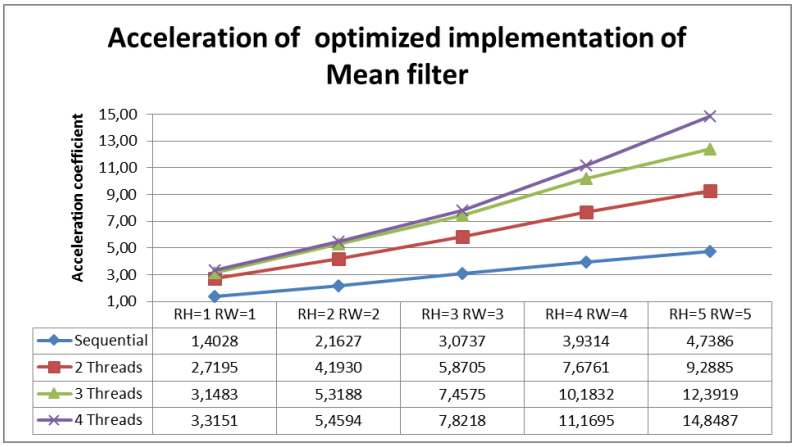
Also we conducted comparison of Mean, Median, Gaussian filters for optimized variant of algorithms implementation. Thus Figures 4-7 demonstrate an acceleration of optimized algorithms, which were parallelized using OpenMP and utilizing 2-4 threads, versus classical implementation for different kernel size.

The dependence of the processing time of filters and used threads for image size 1056×2148 and kernel size 9×9 is demonstrated on Figure 7.

The comparison of time taken for the image processing using the classic and optimized implementation of filters with the kernel size 5×5 pixels (RH=2 RW=2) is shown in Table II. For the experimental study images having different orientation and with size from 0.26 MP (512×512 pixels) to 2.89 MP (1396×2168 pixels) were taken.

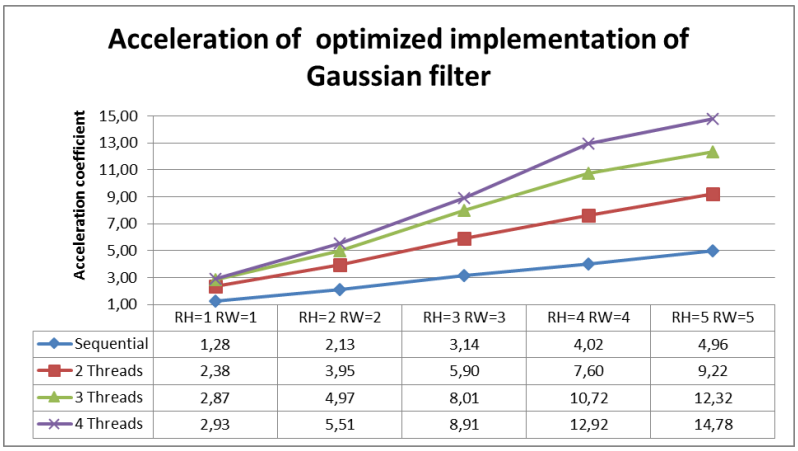
Additionally, we evaluated the acceleration of processing using OpenMP for different kernel sizes at two, three and four threads. During evaluation were used images with size 1056×2148 pixels. The experimental results reveal that the increase in the processing speed for different kernel sizes is

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



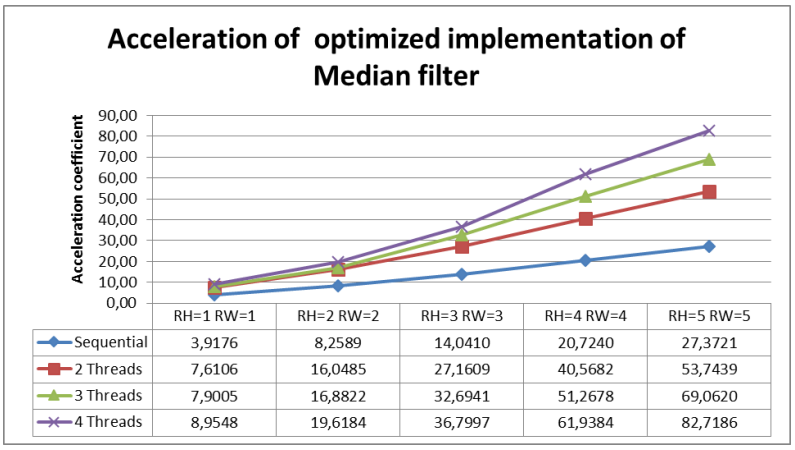
	RH=1 RW=1	RH=2 RW=2	RH=3 RW=3	RH=4 RW=4	RH=5 RW=5
Sequential	1,4028	2,1627	3,0737	3,9314	4,7386
2 Threads	2,7195	4,1930	5,8705	7,6761	9,2885
3 Threads	3,1483	5,3188	7,4575	10,1832	12,3919
4 Threads	3,3151	5,4594	7,8218	11,1695	14,8487

Figure 4. The computation sped up of optimized Mean filter utilizing OpenMP technology compare to classic sequential implementation



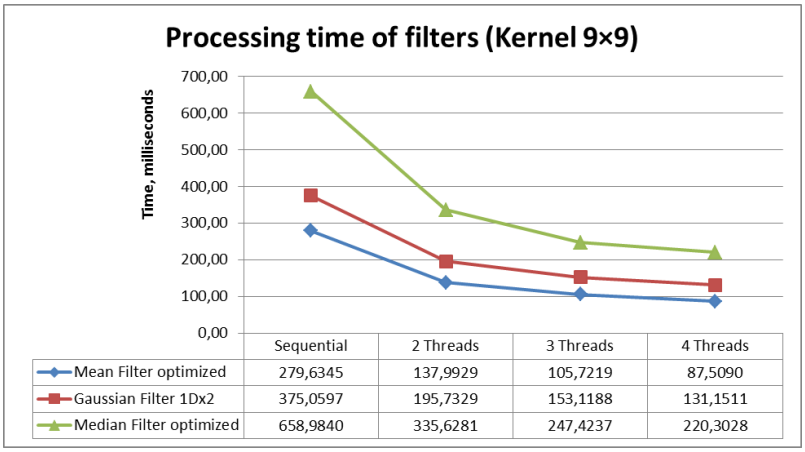
	RH=1 RW=1	RH=2 RW=2	RH=3 RW=3	RH=4 RW=4	RH=5 RW=5
Sequential	1,28	2,13	3,14	4,02	4,96
2 Threads	2,38	3,95	5,90	7,60	9,22
3 Threads	2,87	4,97	8,01	10,72	12,32
4 Threads	2,93	5,51	8,91	12,92	14,78

Figure 5. The computation sped up of optimized Gaussian filter utilizing OpenMP technology compare to classic sequential implementation



	RH=1 RW=1	RH=2 RW=2	RH=3 RW=3	RH=4 RW=4	RH=5 RW=5
Sequential	3,9176	8,2589	14,0410	20,7240	27,3721
2 Threads	7,6106	16,0485	27,1609	40,5682	53,7439
3 Threads	7,9005	16,8822	32,6941	51,2678	69,0620
4 Threads	8,9548	19,6184	36,7997	61,9384	82,7186

Figure 6. The computation sped up of optimized Median filter utilizing OpenMP technology compare to classic sequential implementation



	Sequential	2 Threads	3 Threads	4 Threads
Mean Filter optimized	279,6345	137,9929	105,7219	87,5090
Gaussian Filter 1Dx2	375,0597	195,7329	153,1188	131,1511
Median Filter optimized	658,9840	335,6281	247,4237	220,3028

Figure 7. The dependence of the filters processing time from used threads.

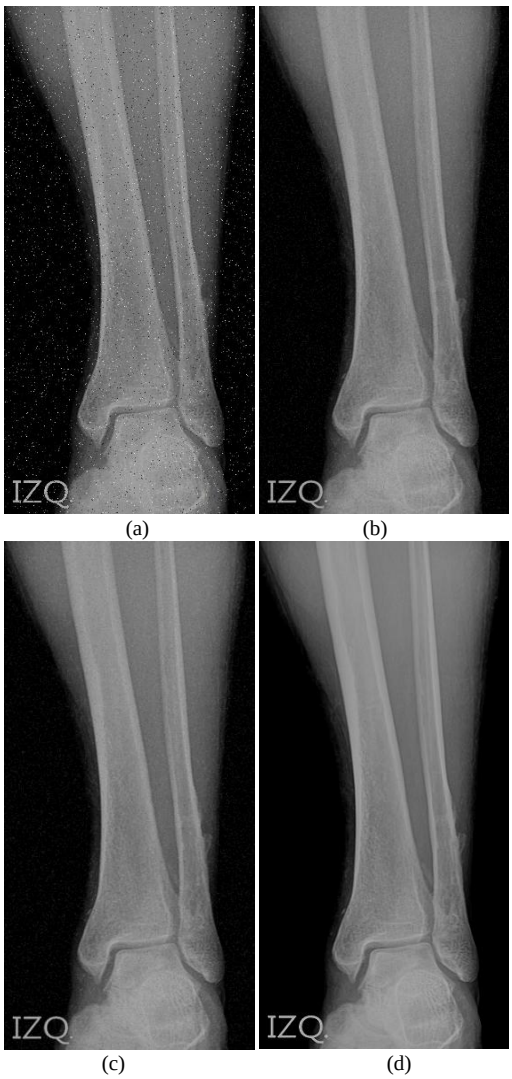
ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

almost the same (Table III).

To evaluate noise suppression of filters were generated test data set emulating noise which may occur in the equipment. The following noise characteristics were used. Thus we used noise which covers from 5% to 25% (noise level). The generated noise map consists 15% of the impulse noise component and 85% of the additive noise component. The value of the additive noise component considered as 15% of the dynamic range of the experimental data.

Figure 8 shows the results obtained when processing an image with 15% noise by Gaussian (sigma 2), mean and median filters with kernel size 5×5.

The evaluation of noise suppression for filters with different kernel size is shown in Table IV. It demonstrates average PSNR values for image with noise (size of image is 1056×2148) and images processed by filters. During experimental study were generated 300 images with noise.



(a) (b)

(c) (d)

Figure 8. Results of image processing: (a) original image with 15% noise 15; (b) processed by Gaussian filter kernel 5×5 sigma 2; (c) processed by mean filter kernel 5×5; (d) processed by median filter kernel 5×5.

TABLE II.
PROCESSING TIME OF FILTERS FOR DIFFERENT IMAGE SIZE (4 THREAD OPENMP), ms

Image size	Filter					
	Mean Filter		Gaussian filter		Median Filter	
	classic	optimized	classic	optimized	classic	optimized
512×512 (0,26 MP)	16,351	8,133	18,980	9,317	166,003	22,272
640×480 (0.31 MP)	21,915	9,281	28,780	11,657	189,979	27,877
768×768 (0.59 MP)	35,802	14,807	42,470	20,872	329,367	42,199
1024×768 (0.79 MP)	44,090	20,581	53,309	24,797	453,629	46,211
1280×720 (0.92 MP)	55,682	29,079	59,238	34,911	523,766	53,756
1440×1080 (1.56 MP)	80,516	41,719	109,943	45,306	850,261	98,906
1920×1080 (2.07 MP)	120,490	50,264	148,818	66,429	1084,792	129,491
1056×2148 (2.17 MP)	120,123	61,804	156,094	86,268	1176,891	134,528
1396×2168 (2.89 MP)	178,179	92,056	255,100	128,221	1632,550	203,446

TABLE III.
EVALUATION OF THE ACCELERATION STABILITY FOR OPTIMIZED FILTERS FOR DIFFERENT KERNEL SIZE

Filter	Threads	Kernel Size				
		3×3	5×5	7×7	9×9	11×11
Mean optimized	2	1,97	2,01	2,03	2,03	1,98
	3	2,70	2,76	2,73	2,65	2,64
	4	3,08	3,12	3,55	3,23	2,95
Gaussian optimized (1Dx2)	2	1,78	1,84	1,90	1,92	1,92
	3	2,38	2,55	2,56	2,45	2,45
	4	2,83	2,82	2,85	2,86	2,86
Median optimized	2	1,92	1,95	1,96	1,96	1,96
	3	2,64	2,70	2,75	2,66	2,72
	4	2,98	3,05	3,09	2,99	2,95

TABLE IV: PSNR VALUES

Filters	Kernel Size				
	3×3	5×5	7×7	9×9	11×11
Noise Level 5% / Noise image PSNR:51,4244039962383					
Mean	68,4842	72,4898	72,3296	70,927	69,2666
Gaussian	68,518	72,6628	73,3734	73,4386	73,8396
Median	88,7094	84,6942	81,5822	79,1196	76,7496
Noise Level 10% / Noise image PSNR:45,5018127600979					
Mean	63,2286	68,8242	70,0816	69,517	68,3272
Gaussian	63,1616	68,276	69,4224	69,591	69,9044
Median	88,2476	84,5432	81,6126	79,1476	76,7498
Noise Level 15% / Noise image PSNR:41,9414738268637					
Mean	59,9406	66,0944	68,0394	68,0304	67,2288
Gaussian	59,8356	65,2282	66,5362	66,7478	67,268
Median	87,9284	84,3688	81,4928	79,0522	76,7666
Noise Level 20% / Noise image PSNR:39,5352640898964					
Mean	57,4892	63,8614	66,1734	66,5362	66,0318
Gaussian	57,3658	62,8288	64,2	64,8324	65,4566
Median	87,5268	84,2286	81,439	79,0578	76,712
Noise Level 25% / Noise image PSNR:37,634963500737					
Mean	55,5328	61,9578	64,4476	65,0418	64,7528
Gaussian	55,3988	60,8406	62,2282	62,4686	62,4946
Median	87,1142	84,0664	81,188	78,8754	76,6108

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. CONCLUSIONS

Digital filters were used which in the last decades have taken great impetus for the treatment of images in different fields of science and in the case of the medical image processing better results are obtained in order to make better interpretations. Various filters are used for medical image preprocessing such as classic and optimized: mean filter, Gaussian 2D filter and median filter. The primary purpose of these filters is a noise reduction, but filter can also be used to emphasize certain features of an image or remove other features. Most of image processing filters can be divided into linear filters and nonlinear filters. Nonlinear filters include order statistic filters and adaptive filters. The choice of filter is often determined by the nature of the task and the type and behavior of the data.

Using of OpenMP we made parallel implementation of optimized algorithms, which gives performance boost up in almost two times for two threads and around 3 times for 3 and 4 threads. Experimental results show that the optimized version of filter algorithms can well do with the relationship between the effect of the noise reduction and the time complexity of the algorithms. The most increase of processing speed was gained for median filter. Thus for small kernel (3×3) the acceleration is about 8 times and for large kernel (11×11) 70 times. Optimized version of Mean filter and Gaussian filter give us acceleration about 3 times for small kernel (3×3) and around 13 times for large kernel (11×11).

ACKNOWLEDGMENT

Very thanks to Jorge Quinga medical technologist from Hospital IESS Sangolqui Ecuador for the medical X-ray images.

REFERENCES

[1] Davies E. Machine Vision: Theory, Algorithms and Practicalities, Academic Press, 2012.
[2] Szeliski R. Computer vision. Algorithms and applications. Springer-Verlag London Limited, 2011.
[3] Ramesh J, Rangachar K, Brian G Schunck. Machine Vision. McGraw-Hill, Inc., ISBN 0-07-032018-7, 1995.
[4] Gonzalez RC, Woods RE. Digital Image Processing 3rd edition, Prentice-Hall, 2008. ISBN-13: 978-0131687288, 2008.
[5] Zotin A., Simonov K., Kapsargin F., Cherepanova T., Kruglyakov A., Cadena L. Techniques for Medical Images Processing Using Shearlet Transform and Color Coding. In: Favorskaya M., Jain L. (eds) Computer Vision in Control Systems-4. Intelligent Systems Reference Library, vol 136. Springer, Cham. Chapter First Online: 27 October 2017 DOI https://doi.org/10.1007/978-3-319-67994-5_9
[6] Chandel et al. Image Filtering Algorithms and Techniques: A Review // International Journal of Advanced Research in Computer Science and Software Engineering 3(10), pp. 198-202, 2013.
[7] Gupta B, Singh Negi S Image Denoising with Linear and Non-Linear Filters: A REVIEW // International Journal of Computer Science Issues, Vol. 10, Issue 6, No 2, pp. 149-154, 2013.
[8] Lukin A. Tips & Tricks: Fast Image Filtering Algorithms. 17-th International Conference on Computer Graphics GraphiCon'2007: 186–189, 2007.
[9] Pascal G. A Survey of Gaussian Convolution Algorithms. Image Processing On Line 3: 286–310, 2013.
[10] Young IT, Van Vliet LJ. Recursive implementation of the Gaussian filter. Elsevier Signal Processing 44: 139–151, 1995.
[11] Zing A. Extended Binomial Filter for Fast Gaussian Blur. Vienna, Austria, 2010.
[12] Cline D, White KB, Egbert PK. Fast 8-bit median filtering based on separability. In Image Processing ICIP 2007 IEEE International Conference 5: V-281–V-284, 2007.
[13] Perreault S, Hebert P. Median filtering in constant time. IEEE Transactions on Image Processing 16(9): 2389–2394, 2007.
[14] Chandra R, Dagum L, Kohr D, Maydan D, McDonald J, Menon R Parallel programing in openmp. Academic Press. USA. 249p ISBN 1-55860-671-8, 2001.
[15] Kiessling A. An Introduction to parallel programming with OpenMP. A Pedagogical Seminar. The University of Edinburgh. UK, 2009.
[16] Shameem A, Jason R Multi-Core Programming. Digital Edition Intel Press. USA. 362p ISBN 0-9764832-4-6, 2006.