A Parallel and Serial Domain Decomposition for Noise Removal Using a Nonlinear Diffusion Equation

Maurílio Boaventura[1], Evanivaldo Castro Silva Júnior[2], Célia Aparecida Zorzo Barcelos[3] and Inês Aparecida Gasparotto Boaventura[1]

[1] São Paulo State University, São José do Rio Preto-SP, Brazil
[2] Faculty of Technology, S.J. Rio Preto - SP and UNIFEV - Votuporanga - SP, Brazil
[3] Federal University of Uberlândia, Uberlândia-MG, Brazil

Abstract. The main purpose of this paper is to show a domain decomposition with parallel and serial approaches for a nonlinear diffusion model used in the processing of digital images suggested by Barcelos, Boaventura and Silva Jr. [3], in addition to a multi-domain analysis for the evolution of this equation, which has as its main characteristic a balanced diffusion with a consequent preservation of boundaries. Some numerical results will be shown in order to illustrate the performance obtained in the model.

1 Introduction

The use of partial differential equations (PDE) for digital processing of images has an intrinsically dynamic character regarding the development of mathematical models. Several papers have been developed from the anisotropic equation suggested by Malik Perona (see, [1], [2], [3], [5], [6], [7], [8], [9] and [11]).

The Malik-Perona model has been continually researched considering that diffusion equations may be successfully used in the noise elimination and segmentation processes, even though it may cause the loss of essential information on the image, such as boundaries. In order to avoid such an undesirable effect, several modifications were introduced into the original model. To reduce the degenerative effects of the diffusion to acceptable levels Nörstrom, [10], suggested the introduction of a new term, into the Malik-Perona model, which maintains the noisy initial matrix close to the smoothed matrix during the whole process of noise elimination. This procedure eliminates the need to interrupt the evolutional process at a certain time, keeping the original characteristics of the image such as edge, but causing, as a consequence, insufficient noise elimination.

Aiming to solve this problem Barcelos, Boaventura and Silva Jr. [3] proposed a balanced diffusion model with boundary preservation. The introduction of the term suggested by Nördstrom has been maintained with a introduction of the a moderation selector function, keeping the image's original characteristics effectively in the boundary or textured regions, so keeping the main structure

of the image and intensifying the smoothing of the image in the homogeneous regions, eliminating the noise presented in the image.

In this work the domain of the image is split into several subdomains making the processing carried out by the proposed model in [3] naturally parallelizable. The main computational and theoretical aspects related to the parallel implementation of the algorithm are presented. To reach good results using the parallel procedure two points were proposed, the first related to the parameter k present in the border detector, the second regarding the use of appropriate boundary conditions applied to each subdomain avoiding spurious results when the subdomain are grouped to define the processed image at the end of the PDE time evolution process. In general the Neumann's boundary condition is used for the time evolution process for noise removal, however the use of such a condition allows that undesired perturbations in the boundary regions, be propagated into the image domain, causing damage in the image obtained by the smoothing process. In [12], the authors have suggested the use of other border conditions for treating images through PDE, which have shown better practical results, decreasing the propagation errors. As in [12], here also the Neumann-Median Filter condition is adopted, which was obtained by use of the Neumann's condition and Median Filters. These facts have contributed to obtaining a highly parallelizable algorithm.

Numerical results show the good performance reached by the parallel implementation.

This paper is organized as follows: in section (2) the mathematical model and its discretization are presented; section (3) presents the proposed domain decomposition; section (4) shows some experimental results; section (5) presents some results showing the performance of the proposed domain decomposition and, finally, section (6) presents some general conclusions.

2 Model and Discretization

We can consider an image as a limited function $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ which associates each coordinate $(x, y) \in \mathbb{R}^2$ to $u(x, y) \in \mathbb{R}$, as an intensity of gray level.

The model of nonlinear diffusion presented by Barcelos, Boaventura and Silva Jr.[3] is given by the equation:

$$u_t = g|\nabla u| div(\frac{\nabla u}{|\nabla u|}) - (1-g)(u-I), (x,y) \in \Omega,\ t > 0, \quad (1)$$

$$u(x, y, 0) = I(x, y), \qquad (x, y) \in \Omega$$

$$\frac{\partial u}{\partial n}|_{\partial\Omega \times R_+} = 0, \qquad (x, y) \in \partial\Omega, t > 0,$$

where $g = g(|G_\sigma * \nabla u|) = \frac{1}{1+k|G_\sigma * \nabla u|}$, $I(x, y)$ is an image to be processed, $u(x, y, t)$ is its smoothed version in the scale "t", G_σ is a convolution kernel

(here, a Gaussian function), $G_\sigma * \nabla u$ is the local estimate of ∇u used for noise elimination, and k is a parameter. $u - I$ is the term suggested by Nördstron and $(1 - g)$ is the moderation selector introduced in this model.

The balanced diffusion of the image allows the homogeneous regions ($g \sim 1$) to be smoothed even more in relation to the boundary regions ($g \sim 0$). This is obtained through the moderation selector $(1 - g)$ that by being in function of $g(|\nabla G_\sigma * u|)$ allows the identification of these different regions of image since $|\nabla G_\sigma * u|$ is an approximation to $|\nabla u|$.

An important fact in the proposition of this model is the introduction of the parameter σ, used in the Gaussian kernel convolution, as the standard deviation of the noisy image, σ_{noise}. Other authors have arbitrarily chosen this parameter. The choice $\sigma = \sigma_{noise}$, justified by the authors, represents a measure of error dispersion in the image and, thus, it functions appropriately as a size of the Gaussian filter.

Another important point presented in [3] is the introduction of the smoothing optimal time concept, which gives us an estimate of the necessary time to stop the evolutional process in the time scale. This estimate is given in the formula:

$$T_o = \frac{\sigma^2}{a},$$

where a is a parameter. The smoothing optimal time concept has been improved in [4], where the parameter a has been changed, appropriately, by σ_{noise}.

The discretizations used in the implementation of the model [3] are very simple taking into account the explicit character used. Let us consider a matrix of intensity values $u(x, y)$, at the points $x = x_i = i\Delta x$ and $y = y_i = i\Delta y$. Let us denote $u(x_i, y_i, t_n)$ as $u^n_{i,j}$ where $t_n = n\Delta t$ and Δt is the step time.

The derivative of u in relation to the time, u_t, calculated in (x_i, y_j, t_n) is approximated by Euler's method, i.e., $u_t \sim \frac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t}$, and the diffusion term

$$|\nabla u|(div(\frac{\nabla u}{|\nabla u|})) = \frac{u_x^2 u_{yy} - 2u_x u_y u_{xy} + u_y^2 u_{xx}}{u_x^2 + u_y^2}$$

is approximated by using central differences.

By using Neumann's boundary conditions, also approximated by using central differences, we calculate u^{n+1}_{ij} , $n = 1, 2, \cdots, N$, by

$$u^{n+1}_{ij} = u^n_{ij} + \Delta t \mathcal{L}(u^n_{ij})$$

with $u^0_{ij} = I(x_i, y_i)$ and

$$\mathcal{L}(u) = g|\nabla u| div(\frac{\nabla u}{|\nabla u|}) - (1 - g)(u - I).$$

3 The Domain Decomposition

The domain decomposition is based on data partitioning, which consists of partitioning the matrix input data A into several matrices A_i that will be distributed

among the processors. This means that the image spatial domain is decomposed into s sub-domains of suitable size and form. The domain will be split, in this work, in a regular form, but different forms could be used. For instance, considering a image A (figure 1) split into four subdomains A_i $i = 1, 2, 3, 4$, we reach the result show in figure 2:

Figure 1: Image in a single domain

Figure 2: The image A partitioned image into four sub-domains

Following this idea, the diffusion equation given by equation (1), will be applied to each sub-domains $\omega_i \subset \Omega$. That is:

$$u_t = g|\nabla u| div(\frac{\nabla u}{|\nabla u|}) - (1-g)(u-I), (x,y) \in \omega_i,\ t > 0 \quad (2)$$

$$u(x,y,0) = I(x,y), \qquad (x,y) \in \omega_i,$$

where $g = g(|G_{\sigma_i} * \nabla u|)$, $I(x, y)$ is an image to be processed, $u(x, y, t_i)$ is its smoothed version in the scale "t_i", G_{σ_i} is a convolution kernel (here, a Gaussian function), and $G_{\sigma_i} * \nabla u$ is the regional estimate of ∇u used for noise elimination, and $\bigcup \omega_i = \Omega$ and the $\bigcap \omega_i = \emptyset$

In the same way the σ_{noise} parameter, presented in the Gaussian function, was used to smooth the image $u(x, y)$ throughout the entire domain, the local parameter σ_i is used in each subdomain $\omega_i \subset \Omega$. This choice is justified considering the local characteristics of the image. Consequently, there will be a better parametrization, which will result in a different time evolution according to ω_i, considering that each subdomain can have their proper and independent characteristics from the other sub-domains. For instance, considering two sub-domains ω_i and ω_j, with standard noise deviation σ_i and σ_j, respectively, such that $\sigma_i < \sigma_j$, which means that the region ω_i is more homogeneous than the region ω_j, we will have in ω_j an evolutional process more accentuated than in ω_i, once that the stop time is given by $T_o = \frac{\sigma^2}{\sigma_{noise}}$ [3]. In summary, each subdomain can have a different scale space.

The parameter k present in the edge detector, defined by the g function, is frequently taken as an arbitrary parameter chosen by the user and it is intrinsically related to the amount of detail we want to preserve. The success of the model depends on the correct finding of the edges, which, in turn, depends on the appropriate choice for the constant k. When working with several different subdomain an appropriate choice parameter for each region is required for a better performance.

Since σ_i is distinct for each subdomain ω_i, we should expect to have distinct values for the constant k, once it is directly related to the quantity of detail we want to keep in the image, and also with the amount of noise we want to eliminate. In [4], the authors suggested an automatic method to estimate the value of k. From the analysis of a set S of different images with different levels of noise and different complexities, a k versus σ dispersion diagram was defined, which was used to find the best fitting curve for that data. The parameter σ represents the noisy image standard deviation and the used k values were obtained from the images belonging to S. This value was taking as that produced as the best result for each image in S. Using the least square approximation they found a fitting curve $k(\sigma) = a \times e^{b\sigma}$, with constants a and b real and positives.

Thus, the parameter k was automatically obtained using this approximation, which can take different values in each subdomain, since they can have different complexities and proper characteristics, eliminating in this way, the necessity of the the arbitrary definition, by the user, of the different values of k for different subdomains.

Another very important fact for suitable parallelization of the model is the use of suitable boundary conditions, which does not introduces interferences in the numerical noise elimination process, that is, conditions that do not introduce great perturbations in the process, given more stability to the numerical process. Neumann's boundary condition is usually used to obtain the PDE through time evolution, however, the use of such a condition may produce undesirable perturbations in the boundary regions, which can be propagated into the domain, resulting in an inefficient image smoothing process.

One can easily see how the use of Neumann's boundary conditions can interfere in the result. The image contrast was changed completely.

Figure 3 (a) shows the original image and the image processed by use of Neumann's boundary conditions and Figure 3 (b) shows the same image, where the first and last lines and columns, of the processed matrix, have been withdrawn after processing. These regions are those which suffer the most influence from the boundary condition.

(a) (b)

Figure 3: (a) Image processed with Neumann's boundary condition and (b) image processed with Neumann's boundary condition without the first and last lines and columns

This example illustrate how the boundary condition used may interfere with the final result. The proliferation of these effects may lead to rather unsatisfactory results, when dealing with several subdomains.

In [12], the authors suggest the use of others different boundary conditions, which can lead to better practical results, minimizing the error propagation into the image domain.

In this work, as suggest in [12], the condition called, by the authors, Neumann-Median Filter condition is used, which is defined by the use of Neumann's condition with Median Filters.

The Neumann-Median Filter condition is defined as:

$$\frac{\partial u}{\partial n}|_{\partial\omega_i \times R_+} = 0, \qquad (x, y) \in \partial\omega_i, t > 0$$

and

$$u|_{\partial\omega_i \times R_+} = \begin{cases} m, & \text{if} \quad |u - m|^2 > A\rho^2 \\ \\ u, \text{otherwise} \end{cases}$$

where $m = \frac{u(x,y-1)+u(x,y+1)+u(x-1,y)+u(x+1,y)}{4}$ is a pixel value mean in the neighboring region of a pixel (x, y) and $\rho^2 = \frac{u^2(x,y-1)+u^2(x,y+1)+u^2(x-1,y)+u^2(x+1,y)}{4} - m^2$ is its variance.

The use of the k parameter obtained automatically, associated with the use of a more appropriate boundary condition, has allowed each subdomain to be processed independently one from another, without causing great visual damage when joining the processed data in the subdomain.

The processes generated by this decomposition of the domains are all independent from one to another and so, the exchange of information among the processes is not needed. Only at the end of the execution of each process is there the need to join the results obtained from each individual process separately in order to obtain the entire

final image. This fact leads to a method with maximum use of parallelism, that is, a highly parallelizable algorithm, leading to an excellent performance. In addition, the total independence among the processes allows for the use of essentially sequential machines, even when dividing the original domain into several subdomains, without any increase the processing time when compared to the single domain processing. this domain decomposition, could be also used in a shared memory environment, in this situation we do not need any communication among de processors.

In this work we use a distributed memory environment. For the parallel implementation the basic idea of using parallel programming has been used by message passing, that is, a number of serial processes exchanging information (data) through messages. The public domain package MPI was used for message passing. The message passing process is needed only at the beginning of the parallel application, in order to initialize the processor, and in the final running, where it is necessary to grouping the data of each available processor. It is an opportunity to emphasize that there is no other message passing during the run code.

4 Experimental Results

Some results obtained with the application of the described models are presented in several images with different complexity levels.

The images used are represented by $N \times M$ matrices where each matrix element $u_{i,j}$ is a real value correspondent to the grayscale level of the image $u(x, y)$.

The original images (noiseless) are represented in using the grayscale levels. Although they present gray intensity levels in the range from 0 to 255, the addition of noise to the intensity values is far beyond the 0-255 range. For example, the noisy image shown in Fig.4. contains pixels with intensities as low as -870 and as high as 1150. The noise levels (SNR) for each of the considered pictures are shown on each of the picture descriptions respective legends. For visualization of the images the Matlab© code was used.

The first experiment (Figures 4-5) a synthetic image containing geometric objects was considered. Figure 4 presents the original and noisy images.

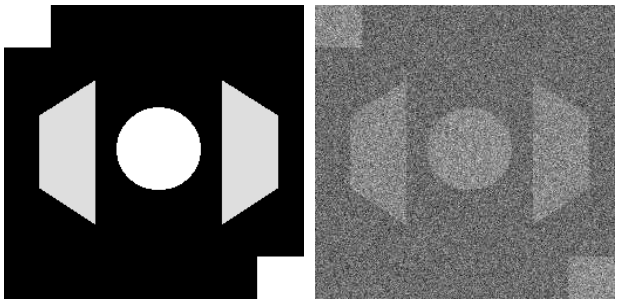


Figure 4: (a) and (b) - Original and Noisy Image (SNR = -6 db)

Figure 5 shows the processed image using four subdomains. Each of these shown in Figure 6.

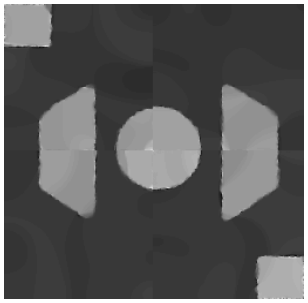


Figure 5: Processed image in four sub-domains

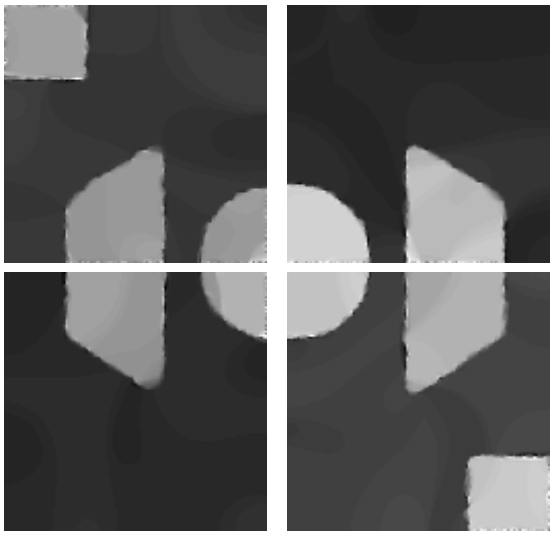


Figure 6: Image partitioned in four sub-domains

Figure 7 shows the same image processed with four subdomains and using Neumann's boundary conditions. In this case, one notes that there is a clear contrast problem.

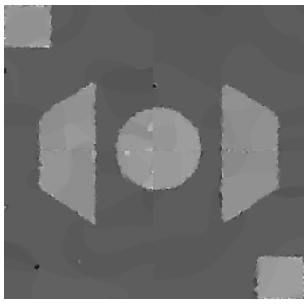


Figure 7: Processed image in four sub-domains using Neumann´s condition

Figure 8 (a) and (b) shows the original and noisy Lenna pictures, respectively. In Fig.9 (a) and (b), the processed version of the images it is presented with 1 and 4 subdomains, respectively.

(a) (b)

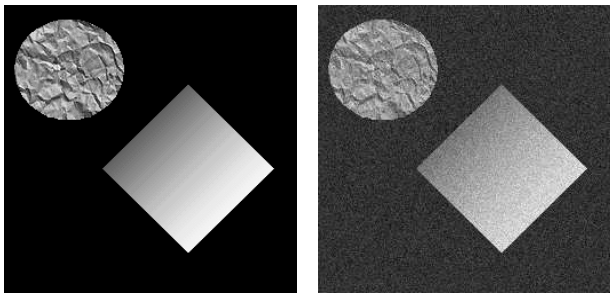
Figure 8: Original and Noisy Image (SNR = 9 dB)

(a) (b)

Figure 9: Processed image in (a) 1 sub-domain (b) 4 sub-domains

One can see in Figure 9 (a) and (b) that the processed image in four subdomains shows, subtly, better quality in the smoothing aspect without boundary losses, consequently offering better visualization.

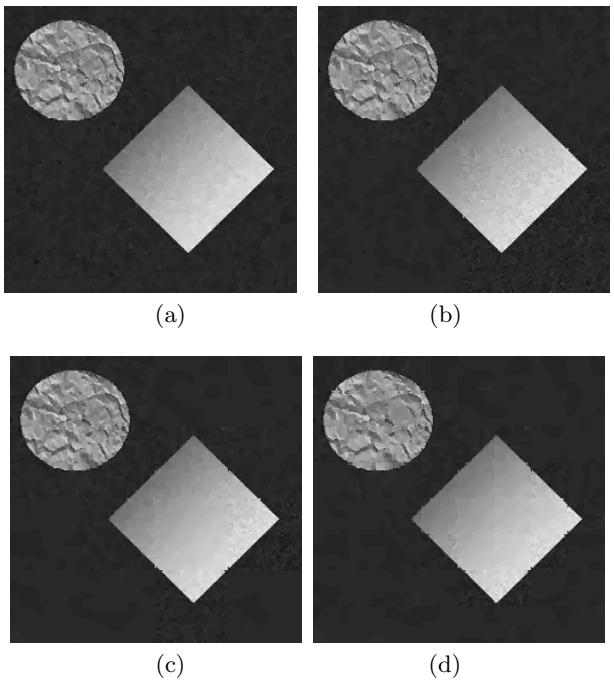
The next experiments consist of the decomposition of the image presented in (Figure 10 (b)). Figure 10 (a) the noiseless version.



(a) (b)

Figure 10: Original and Noisy Image (SNR = 12 dB)

In Figure 11 (a), (b), (c) and (d)the processed images with 1, 4, 16 and 64 subdomains, from left to right and top to bottom are presented.



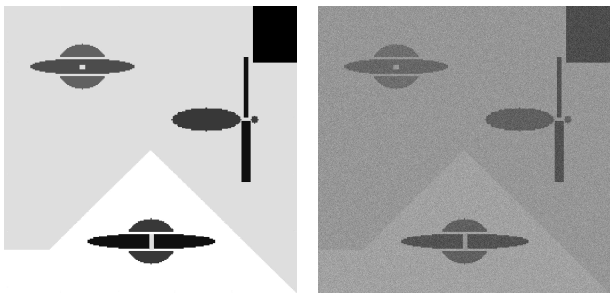
(a) (b)

(c) (d)

Figure 11: Processed image in (a) 1, (b) 4, (c) 16 and (d) 64 sub-domains

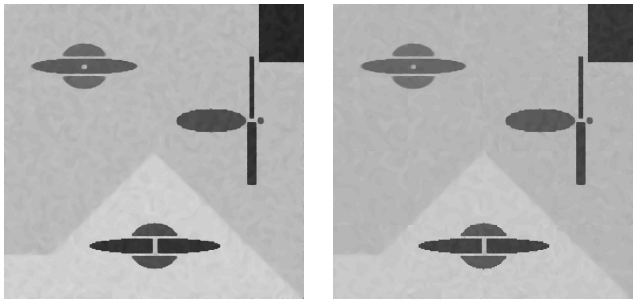
By analyzing these images (Figure 11 (c) and (d)), one notices that the great number of subdomains used for its processing have not affected the final result, preserving its visual quality.

The following figure refers to another synthetic image, where good results can be seen.



(a) (b)

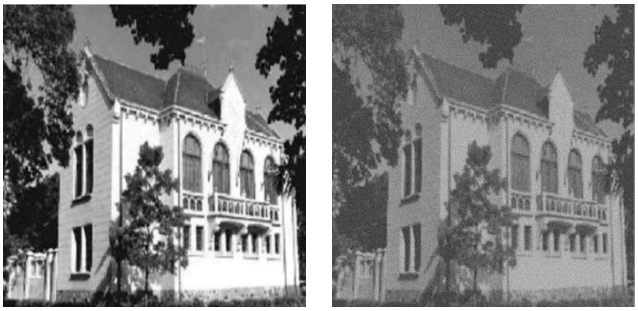
Figure 12: Original and Noisy Image (SNR = 0 dB)



(a) (b)

Figure 13: Processed image in (a) 1 sub-domain (b) 16 sub-domains

The following figure refers to a real life image, where good results can be seen.



(a) (b)

Figure 14: Original and Noisy Image (SNR = 6 dB)

(a) (b)

Figure 15: Processed image in (a) 1 sub-domain (b) 16 sub-domains

5 Performance Analysis

In order to analyze the performance the parallel code efficiency is measured by:

$$E = \frac{T_P}{p \times T_s},$$

where p denotes the number of processors, T_s denote the time spent by the sequential process, T_i, denotes the time spent by process i, and $T_P = \max(T_1, T_2, \cdots, T_p)$.

The first experiment refers to Figure 4, with $N = M = 256$ and $\triangle t = 0.1$ in equation (2). The following table, table 1, show the time (in seconds) spent for each processor (P_i), using 4 processors, the sequential code time, and other parameters are also presented.

Table 1: Time in seconds

processors	P_1	P_2	P_3	P_4	Sequential
Time(s)	183	192	180	193	756
σ	276	273	269	275	273
iterations	929	953	956	925	939

The second experiment, a synthetic image, refers to the image present in Figure 12 with $N = M = 1024$. In this case, $\triangle t = 0.25$ it was used. The following table, Table 2, show time in seconds obtained by each processor (Pi), for 4 processors, the sequential code time, and other parameters.

Table 2: Time in seconds

processors	P_1	P_2	P_3	P_4	Sequential
Time(s)	830	625	775	763	3300
σ	73	102	79	82	87
iterations	204	146	190	182	171

The third experiment, a real life image, refers to the image present in Figure 15 with $N = M = 2560$. In this case, $\triangle t = 0.1$ it was used. The Table 3 following show time in seconds obtained by each processor (Pi), for 4 processors, the sequential code time, and other parameters.

Table 3: Time in seconds

processors	P_1	P_2	P_3	P_4	Sequential
Time(s)	429	397	401	383	1634
σ	33.69	33.68	33.70	33.73	33.70
iterations	168	165	156	149	150

The efficiency measurements obtained in the first example were $E = 0.98$ and $E = 0.84$, obtained with 4 and 16 subdomains, respectively. In the second example, $E = 0.99$ and $E = 0.86$ with 4 and 16 subdomains, respectively. In the third example, for the problems with dimensions 2560×2560, the efficiency was $E = 0.95$ and $E = 0.78$,

with 4 and 16 subdomains, respectively. However, for the problem with dimensions 256×256, $E = 0.92$ and $E = 0.65$, with 4 and 16 subdomains, respectively. These results indicate that the efficiency is better for large problems.

The efficiency was worse in the third example, where the first process needed more iterations in the time scale space and this fact resulted in an unbalance among the processors. Nevertheless, these results can still be considered excellent.

For the Lenna picture, the computational results were similar to the ones from third example.

One can also note that, when one uses several subdomains in serial machines, the total processing time is lower than the processing time resulting from a single domain. For instance, for the third example, with dimensions 2560×2560, when running in a single domain, the processing time was 1634s; with 64 subdomains the processing times was 1204s and with 256 subdomains it was 1187s. This is justified for two reasons: first, splitting the domain in several subdomains there can exist some of the subdomains that use a shorter time of PDE evolution to reach an appropriate smoothing level; second, for large scale problems, if all the elements of the image matrix can not be stored in the cache memory, the processing time can be slow. However, with the use of domain decomposition, one reduces the dimensions of the problem, possibly storing all the required data in the cache memory, thus accelerating the computational process.

6 Conclusion

The use of the concept of optimal smoothing time and the automatic estimating of the k parameter, present in the boundary detector, was essential for the development of the proposed parallel model taking into accounting that each has a specific k value.

The parallel code has presented a considerable computational gain, reaching an excellent performance.

The total processing time, in serial architectures when using several subdomains, can be inferior to the processing time of the problem when we dealing with a single domain. Due fact that splitting the domain in several subdomains there can exist some of the subdomains that use a shorter time of PDE evolution to reach an appropriate smoothing level and due to the influence of cache memory in the processing time.

The use of the proposed parallel technique makes it possible that large images are quickly processed without any quality loss in the final result.

An important aspect of parallel computing is load balancing. Balancing the work load fairly among the available processors is crucial for the good performance of the parallel code. With the domain decomposition presented, the processors do not need data exchange with each other during the computation process. From time to time, one or another process might need to develop less in the time scale and this fact may cause a certain unbalance.

At this moment the code is being finely tuned, so that we hope to report an even better study of the performance in the near future.

This domain decomposition can be used by distributed and shared memory environment. This domain decomposition can also be used by serial process with great success for large images, especially when working with low memory capacity.

Acknowledgements

This research has been partly supported by CAPES.

References

1. Alvarez, L., Esclarim, J.:Image Quantization using Reaction- Diffusion Equations. SIAM Journal on Applied Mathematics. **57** (1997) 153–175.
2. Alvarez, L., Lions, P.L., Morel, J.M.:Image selective smoothing and edge detection by nonlinear diffusion. SIAM J.Numer. Anal. **29(3)** (1992) 845–866.
3. Barcelos, C.A.Z., Boaventura, M., Silva Jr., E.C.:A Well-balanced Flow Equation For Noise Removal and Edge Detection. IEEE Transactions on Image Processing. **12(7)** (2003) 751–763.
4. Barcelos, C.A.Z., Boaventura, M., Silva Jr., E.C.:Edge Detection and Noise Removal with Automatic Selection of Paramenters foa a PDE Based Model. Computational and Applied Mathematics. **24(1)** (2005) 131–150.
5. Barcelos, C.A.Z., Chen, Y.:Heat Flows and Related Minimization Problem in Image Restoration. Computers & Mathematics with applications. **39** (2000) 81–97.
6. Chen,y.,Vemuri, B.C., Wang, L.:Image denoising and Segmentation via nonlinear diffusion. Comput. Math. Appl. **39 (5/6)**, (2000) 131–149.
7. Catté, F., Coll, T., Lions, P.L., Morel, J.M.:Image selective smoothing and edge detection by nonlinear diffusion. SIAM Journal on numerical analysis. **29** (1992) 182–193.
8. Koenderink, J.J.: The structure of images. Biol. Cybernet. **50**, (1984) 363-370.
9. Malik J., Perona, P.: Scale-space and edge detection using anisotropic diffusion. IEEE TPAMI, **12(7)** (1990) 629–639.
10. Nordström, K.N.:Biased anisotropic diffusion: a unified regularization and diffusion approach to edge detection. Image and Vision Computing. **8** (1990) 318–327.
11. Rudin, L., Osher, S., Fatemi, E.:Nonlinear total Variation based noise removal algorithms. Physica D. **60** (1992) 259–268.
12. Silva Jr., E.C., Boaventura, M., Barcelos, C.A.Z.:Boundary Conditions Analysis in Digital Image Processing by PDE. Proceedings of the Workshop on Nonlinear Phenomena Modeling and Their Applications.(2005) 49–50.