Parallel Multi-scale Feature Extraction and Region Growing: Application in Retinal Blood Vessel Detection

Palomera-Pérez, M.A., Martinez-Perez, M.E., Benítez-Pérez, H., and Ortega-Arjona, J.L.

***Abstract*—This paper presents a parallel implementation based on ITK for a multi-scale feature extraction and region growing algorithm, applied to retinal blood vessels segmentation. This implementation is capable of achieving an accuracy comparable to its serial counterpart (about 92%), but 8 to 10 times faster. In this paper, the accuracy of this parallel implementation is evaluated by comparison with expert manual segmentation (obtained from public databases). On the other hand, its performance is compared with previous published serial implementations. Both these characteristics make this parallel implementation feasible for the analysis of a larger amount of high-resolution retinal images, achieving a faster, high-quality segmentation of retinal blood vessels.**

***Index Terms*—Distributed algorithms, Data processing, Image analysis, Image processing, Image segmentation, Parallel programming.**

I. Introduction

RECENT studies show that quantitative measurements of the retinal microvasculature can be used for the diagnosis of several diseases [1], [2]. Commonly, blood vessel assessment is used for predicting cardiovascular diseases, as an indicator of target organ damage in hypertension, and in the diagnosis of diabetes. However, manual or semi-automated techniques for the quantification of the retinal microvasculature are time-consuming and labor intensive. Thus, the development of rapid means for retinal microvascular assessment are valuable. This situation opens the use of computer algorithms for segmenting and measuring retinal vasculature as useful techniques.

Nowadays, computer-aided image analysis is becoming increasingly important to efficiently and safely handle large amounts of high-resolution retinal images. Developments in acquisition technology enable us to capture increasing amounts of high-resolution retinal images, with unprecedented detail. For instance, the use of digital cameras for retinal screening in diabetes is generating an ever increasing large database for retinal analysis in high risk individuals. In clinical routine, such large amounts of data raise challenges for retinal image analysis and processing. Hence, it is necessary to develop computer algorithms capable of processing current large databases of these retinal images. Currently, information processing is undergoing rapid advances driven by the use of high-performance computing, resulting in the development of practical medical applications [3], [4].

In particular, a novel multi-scale feature extraction and region growing for segmentation of retinal blood vessels, developed in Matlab, has been proposed achieving considerably accurate segmentation results [5]. Nevertheless, this effort is relatively slow and unable to process high-resolution images due to memory limitations. An alternative computationally efficient version has been implemented using the Insight Segmentation and Registration Toolkit (ITK) [6], aiming to solve speed and memory limitations [7]. However, although this implementation allows to process higher-resolution retinal images, it requires a considerable time for processing.

The main objective of this paper is to propose a parallel implementation for retinal blood vessel segmentation, capable of achieving an accuracy similar to the ITK serial version [7], while providing a faster processing of higher-resolution images and larger data sets.

The challenge of deploying a parallel segmentation algorithm is to keep the amount of communication low. In this work, a novel approach is presented where the image is subdivided into sub-images. Each sub-image to be processed should have overlapping regions in order to have a low rate of communications. Moreover, it is shown that using this new methodology improves the segmentation process time without compromising the algorithm accuracy.

This paper is organized as follows: In Section II we briefly describe the original blood vessel segmentation algorithm [7]. Section III outlines the basics of the parallel implementation. Section IV presents an evaluation of the parallel implementation performance compared with the original ITK serial version (in terms of execution time and speed-up), as well as the accuracy of its results compared with public databases. Finally, Section V gives the conclusions and suggestions for future work.

II. Blood vessels segmentation algorithm

The algorithm for blood vessel segmentation is based on basic principles of multi-scale differential geometry, using

Manuscript received on November, 2007. This work was supported in part by the Macroproyecto: Tecnologías para la Universidad de la Información y la Computación.

Palomera-Pérez and Benítez-Pérez are with the Department of Computer Systems Engineering and Automatization, IIMAS, UNAM. Mexico. (phone: +(52) 5556223639 e-mail: ese.mike@gmail.com and hector@uxdea4.iimas.unam.mx).

Martinez-Perez is with the Department of Computer Science, IIMAS, UNAM. Apdo. Postal 20-726. Delg. Alvaro Obregon, Mexico, D.F., C.P. 01000. Mexico. (e-mail: elena@leibniz.iimas.unam.mx).

Ortega-Arjona is with Department of Mathematics ,Faculty of Science, UNAM Apdo. Postal 20-726. Admon. 20 Delg. Alvaro Obregon, Mexico, D.F., C.P. 01000. Mexico. (e-mail: jloa@fciencias.unam.mx).

first and second derivative information, in combination with a region growing strategy [5]. The latter algorithm has been tested using several databases of complete manually labelled images [8], [9]. These databases have been used by other authors for similar validation purposes [8], [9], [10].

Multi-scale techniques provide a way for isolating information about objects, by considering geometrical features at different scales. These features are obtained by convolving the original image $I(x, y)$ with a Gaussian kernel $G(x, y; s)$ with variance s^2:

$$I_s(x, y; s) = I(x, y) \otimes G(x, y; s) \tag{1}$$

where G is:

$$G(x, y; s) = \frac{1}{2\pi s^2} e^{-\frac{x^2+y^2}{2s^2}} \tag{2}$$

and s is a length scale. In order to extract geometrical features, an approach based on differentiation is used. Derivatives of an image are numerically approximated by a linear convolution of the image with scale-normalized derivatives of the Gaussian kernel.

$$\partial^n I_s(x, y; s) = I(x, y) \otimes s^n \partial^n G(x, y; s) \tag{3}$$

where n indicates the order of the derivative.

A. Feature Extraction

In this algorithm, the features obtained using the multi-scale technique are *(1)* the magnitude of the gradient and *(2)* the maximum eigenvalue of the Hessian matrix.

1) Gradient Magnitude: is defined as:

$$|\nabla I_s| = \sqrt{(\partial_x I_s)^2 + (\partial_y I_s)^2} \tag{4}$$

It represents the slope of the image intensity for a particular value of the scale parameter s.

2) Maximum Eigenvalue: Vessels appear as ridge-like structures in the image, so their pixels are identified where the intensity has a local maximum in the direction of the gradient, the largest change (largest concavity) [11]. Hence, the second derivative information is obtained from the Hessian of the intensity image $I(x, y)$:

$$H = \begin{pmatrix} \partial_{xx} I_s & \partial_{xy} I_s \\ \partial_{yx} I_s & \partial_{yy} I_s \end{pmatrix} \tag{5}$$

The Hessian matrix is symmetrical since $\partial_{xy} I_s = \partial_{yx} I_s$ with real eigenvalues and orthogonal eigenvectors, which are rotation invariant. A pixel belongs to a vessel region if its maximum eigenvalue is $\gg 1$. Thus, it is weighted as a *vessel* pixel. Hence, the overall multi-scale integration is based on extracting the information across the scales by finding the local maximum over scales (pixel by pixel) for both measurements of feature strength [12].

B. Region Growing

The region growing algorithm is based on an iterative relaxation technique. All region growing parameters are obtained for each image, analyzing its extracted features through an histogram of the complete image. Two class of pixels are determined from the histogram of the maximum eigenvalue: *vessel* and *background*. And only one class is calculated from the histogram of the gradient magnitude: *low gradient*.

For region growing, seeds are planted considering the mean values of each eigenvalue class. The classification of pixels as *vessel* or *background* is based on the maximum eigenvalue, where the criteria for determining seeds are defined. Using spatial information from the classification of the 8-neighboring pixels, *vessel* and *background* classes are iteratively grown.

The algorithm starts a first stage in regions with *low gradient* magnitude, allowing a relatively broad and fast classification. Meanwhile, it also suppresses the classification in the edge regions, where the gradients are large. In a second stage, the classification constraint is relaxed, and both the *vessel* and the *background* classes grow based on the the value of the maximum eigenvalue. During this procedure boundaries between regions are defined. For further and complete details of the segmentation algorithm, refer to [5].

III. Parallel implementation

The purpose of parallelizing the segmentation algorithm described above is to process larger data sets of images, whose resolution varies from low to high, in an acceptable time. The main problem for processing such images (particularly high-resolution ones) is the available local memory. Even though the trivial solution may be increasing the amount of memory per processor, the essential problem is not truly solved. Parallelism is applied so the solution is developed by partitioning the images, so each sub-image can be partially processed within the available memory per processor. This kind of parallelism is known as *domain partitioning*, and it is considered for both stages of the segmentation algorithm, *feature extraction* and *region growing*.

A. Feature Extraction in Parallel

The multi-scale process of feature extraction is a local process for both, gradient magnitude and maximum eigenvalue. Thus, for a current pixel, the feature value does not depend on its neighboring pixels. Considering this, the parallel implementation consists of partitioning the image into equal size sub-images, so each sub-image is processed by each processor. After parallel feature extraction, the resulting image is then compound by all resulting sub-images. However, this procedure yields some false vessel edges for boundary pixels of each sub-image. In order to avoid this problem, the original image is divided so that neighboring sub-images contain certain overlapping between each other.

Figure 1 shows an example of partitioning an image into five sub-images. Figure 1(a) shows the simple division into R_i $(i = 5)$ sub-images. Figure 1(b) depicts the overlapping needed to process each sub-image. Each pixel region to be processed is then the composition of an overlapping T region (defined by a number of extra pixels from the neighboring sub-images) along with the pixels belonging to the R_i sub-image. Hence, the pixel region for a particular sub-image is $R_i + 2T$. Nevertheless, top-most and bottom-most sub-images have a size of $R_i + T$. The particular value of T used in this parallel implementation will be described in Section IV. We note that T and R_i are not modify during the process.

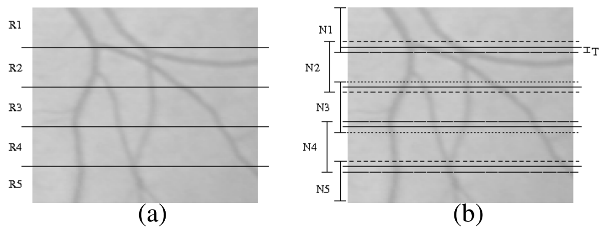

Fig. 1. Partitioning: (a) into sub-images (R_i), and (b) considering pixel regions ($R_i + 2T$).

Once the feature extraction of all the sub-images has finished, their results are collected and assembled into a single overall resulting image. At this point, it is important to eliminate the overlapping for the final image. Notice that this parallel processing can be extended to any number of sub-images without further programming effort.

B. Region Growing in Parallel

Since the region growing algorithm depends on the iteration stage and on the processing of neighbor pixels, the paralleling of this algorithm is not straightforward.

In the current region growing algorithm (as described in Section II-B), a global statistic defines where pixel seeds are planted. The position of these pixel seeds depends on the gradient magnitude and the maximum eigenvalue. Starting from each seed, each pixel class grows by iteratively classifying each pixel as *vessel* or *background*. The growing rules for a given pixel considers the pixels classified in the previous iteration along with the current class of its 8-neighboring pixels. Thus, dividing the image into sub-images is not enough to address the growing problem.

This is a challenging problem. Suppose, for example, that an image is divided into two sub-images, as shown in Figure 2. If the image is horizontally divided (as shown in Figure 2(a)), vessels horizontally growing have no problem. Nevertheless, vessels vertically growing may go across the sub-image borders, and therefore, the algorithm cannot find a direction to grow giving truncated vessels. This also occurs when the image is vertically divided, as it is shown in Figure 2(b).

A possible solution to the problem of truncated vessels is to allow communication between sub-images that share an overlapping, at each iteration. However, this approach may produce an excessive number of communications, so it is not recommended.

Another effective solution is to divide the image into sub-images in both directions, horizontal and vertical, as shown in Figure 2. The resulting image then is obtained by combining both sets of sub-images (horizontal and vertical), as shown in Figure 2(c).

In summary, the parallel region growing algorithm takes an initial empty image, plants the seeds in it, and divides it into two sets of sub-images: horizontal and vertical. Then, each sub-image is processed, performing the growing algorithm on it. Since *vessel* and *background* classes grow in any direction,

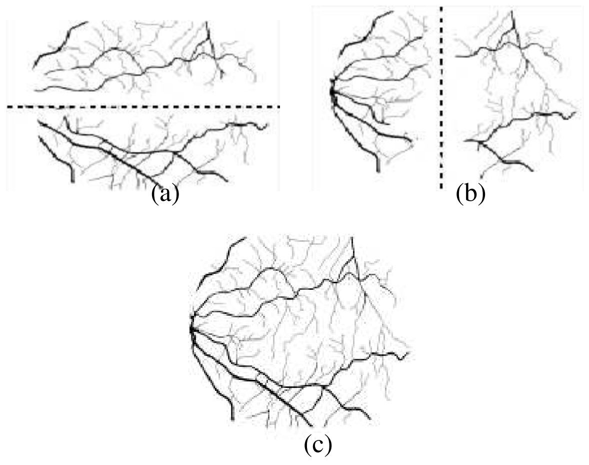

Fig. 2. Process with two nodes. (a) Horizontal, (b) vertical and (c) combined results.

it is likely that some vessels would appear in the vertical sub-image but would not appear in the horizontal sub-image and vice versa. So, the use of an OR operator applied to all pixels of the image corrects this situation. The image is put back together at the end of the growth, considering *vessel* with value 1 and 0 for *background*.

Although each sub-image is represented twice (one vertical, one horizontal) instead of only once, this partitioning scheme does not imply communications, which normally may represent a bottleneck for any parallel application.

Figure 3 shows an example of this kind of partitioning. The image is divided into five horizontal and five vertical sub-images. This partitioning scheme makes easy to collect the final results, allowing to load-balance the process.

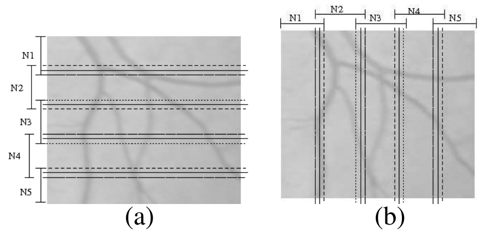

Fig. 3. Partition scheme: (a) horizontal and (b) vertical sub-images.

Once again, this partitioning scheme can be generalized for any number of sub-images.

IV. Evaluation

The parallel implementation for the evaluation is based on Message Passing Interface (MPI), a message passing library [13]. Actually, the Mpich version 1.2.7 is used here and image processing algorithms are developed using ITK version 3.0.0. [6]

The parallel implementation was developed and tested on a Beowulf cluster, composed of a master node and 13 slaves nodes. The main features of this cluster are shown in Table I.

Applying the proposed partitioning scheme, each sub-image (along with its overlapping with neighboring sub-images) is mapped onto each node, for both algorithms, *feature extraction* (Figure 4(a)) and *region growing* (Figure 4(b)).

TABLE I
CLUSTER FEATURES

Features	*Master node*	*Internal nodes*
CPU(s)	2 Xeon at 2.6 GHz	13 Pentium IV at 2.6 GHz
Memory	1.5 GB	13 × 500 MB
Linux Kernel	2.6.8	2.6.12

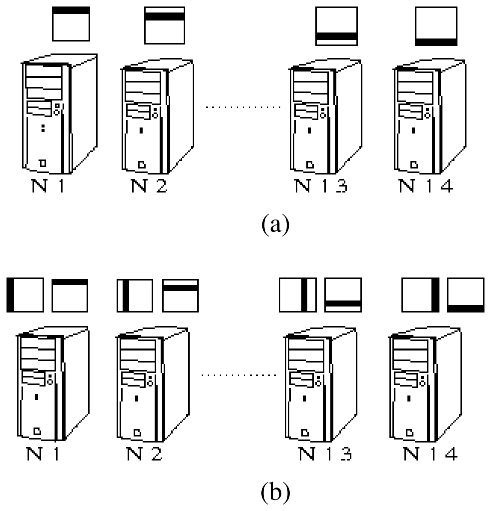

Fig. 4. Partition schemes: (a) Parallel feature extraction (horizontal partition) and (b) Parallel region growing (mixed horizontal-vertical partition)

To test the algorithm, three experiments were performed:

A Comparison of parallel performance with the performance reported for the ITK serial version [5], [7]. In this experiment a local database composed of images of different sizes was used.

B Computation of the accuracy for both, ITK serial and parallel segmentation.

C Calculation of the speed-up of the parallel implementation (regarding the ITK serial implementation) using a high-resolution image (2890 × 2308 pixels), and varying the number of nodes (from 1 to 14).

A. Comparison with ITK serial version

The previous serial version was implemented with the ITK library using the C++ language [7]. ITK includes various high-level and low-level image processing algorithms. It also uses similar concepts to the Standard Template Library (STL), so it works with arbitrary image formats [6].

The ITK version used here was 3.0.0. This version implements the operation of convolution (used in the feature extraction stage) with the derivatives of recursive Gaussian filters, using approximated 2D Gaussian operators [14]. This is computationally more efficient. Besides, during the region growing stage, ITK makes use of an "image iterator" to scan an image region. This implies a more efficient use of memory.

As the ITK serial version, the parallel implementation makes use of ITK, considering the parallelizing domain (as described in Section III). Here, fourth order recursive Gaussian filters of ITK are used with a padding of 4 pixels on each direction (horizontal and vertical). So, each sub-image overlapping (T) is of 4 pixels. The ITK serial version has been tested on the master node of the cluster.

Table II shows the execution time performance of the ITK serial version. To this end we used a local database of 7 different images, with different sizes and scale intervals. The scale intervals correspond to the lower and larger s values of the scale object in the image. These values are set manually, and depend on both, the size of the image in pixels and the magnifying settings of the camera used to capture the retinal image. Execution times are reported in seconds for *feature extraction* (FE) and *region growing* (RG) stages, respectively. All images were successfully segmented.

TABLE II
EXECUTION TIMES OF THE ITK SERIAL VERSION.
FE-FEATURE EXTRACTION, RG-REGION GROWING.

Image	Size (pixels)	Scales	FE (s)	RG (s)
Img1	134 × 116	[1 - 5]	0.284	0.015
Img2	253 × 290	[1 - 4]	1.028	0.115
Img3	703 × 599	[2 - 6]	7.125	1.973
Img4	1319 × 1518	[2 - 8]	49.626	20.823
Img5	2308 × 2890	[5 - 25]	614.934	57.018
Img6	3500 × 3000	[5 - 21]	794.217	252.304
Img7	3500 × 3000	[5 - 35]	1443.860	233.864

Table III shows the performance of the ITK parallel version (executing on 14 nodes), as execution time. For feature extraction (FE), the parallel implementation is 10 times faster than the ITK serial version, whereas for RG, it is hardly twice as fast as the ITK serial version. Globally, the complete segmentation for high-resolution images is processed almost 9 times faster than the ITK serial version.

TABLE III
EXECUTION TIME OF THE ITK PARALLEL VERSION.
FE-FEATURE EXTRACTION, RG-REGION GROWING.

Image	Size (pixels)	Scales	FE (s)	RG (s)
Img1	134 × 116	[1 - 5]	0.175	0.027
Img2	253 × 290	[1 - 4]	0.478	0.355
Img3	703 × 599	[2 - 6]	1.454	1.506
Img4	1319 × 1518	[2 - 8]	6.996	10.872
Img5	2308 × 2890	[5 - 25]	48.192	25.799
Img6	3500 × 3000	[5 - 21]	62.897	74.593
Img7	3500 × 3000	[5 - 35]	105.761	106.794

B. Algorithm Accuracy Computation

In order to validate the segmentation accuracy, two public databases of digitalized retinal images were used.

The first database, STARE [8], contains 20 retinal fundus images photographed with a TopCon TRV-50 fundus camera with a 35° field of view. Each positive is digitalized to produce a 605×700 pixel image, with 24 bits per pixel (standard RGB). Ten images are from subjects with no pathology (normals) and the other 10 with pathology (abnormals). Each of these 20 images were hand-segmented by two different observers named, AH and VK, respectively.

The second public database, DRIVE, has been made available by [9]. This database consists of 40 images captured in digital form from a Canon CR5 non-mydriatic 3CCD camera at $45°$ field of view. The images are 768×584 pixels, 8 bits per color channel. The images have been divided into a *test* and *train* sets with 20 images each. Images in the *test* set are hand-segmented twice resulting in sets A and B. Only the *test* set is used.

Because the green band of color images on RGB format gives the highest contrast between vessel and background, only the green band is used for both databases. Images from STARE database are processed through a scale interval $s \in [2, 12]$, whereas those from DRIVE database uses $s \in [2, 8]$.

The validation of segmentation outcome (SO) is based on contingency tables built using a hand-segmented image as the "ground truth" (GT). Along with STARE database, AH images are used as "ground truth", whereas for DRIVE database, the set A is used. Contingency tables are built as follows: any pixel marked as *vessel* in both GT and SO is a true positive (TP). Any pixel which is marked as *non-vessel* in both GT and SO is a true negative (TN). Any pixel marked as *vessel* in SO, but *non-vessel* in GT is a false positive (FP). The true positive rate (TPR) is established by dividing the number of TP by the total number of *vessel pixels* in GT. The false positive rate (FPR) is computed by dividing the number of FP by the total number of *non-vessel pixels* in GT. A measure of accuracy (Ac) is defined by the sum of TP and TN divided by the total number of pixels in the image. For a perfect segmentation, TPR should be 1 and FPR should be 0.

Comparisons of the present segmentation method (with the Matlab implementation) against the methods used by other authors using these databases have been reported in [5]. These comparisons show that the accuracy achieved using the STARE database is similar as the one reported in [8], having a slightly lower performance than that of [10]. In the case of DRIVE database, the Matlab serial method performs better on detecting correct vessels than the one in [9]. It detects more blobs from the background, particularly in those images with strong pathologies. Anyhow, the accuracy value is very similar for both databases.

Evaluation of the segmentation method between both serial implementations (Matlab and ITK) have been already reported in [7]. It is shown that, although the value of FPR is slightly larger for Matlab, the ITK segmentation had a similar accuracy (Ac) still comparable with other author's techniques, and it is adequate for blood vessel geometry measurements.

Table IV shows the mean and standard deviation values for TPR, FPR, and Ac for both implementations, ITK serial and parallel, using both STARE and DRIVE databases. The parameters (TPR, FPR, and Ac) are calculated only in the field of view (FOV) of each image rather than the complete rectangular image, since the dark background outside the FOV is easily extracted. The same practice is reported by [9], unlike that of [8].

Since there are slight differences between TPRs of the ITK serial and parallel versions for both databases, it seems necessary to determine if these are significant. For this verification, a statistical analysis is used, based on the well known Wilkoxon signed rank test [15]. Based on this, and considering the STARE database, the p values (for $p \leq 0.01$) of the two sided Wilcoxon signed rank test show that there are not significant differences for any of the three parameters between groups (with all $p >> 0.6$). This enables the comparison in this experiment.

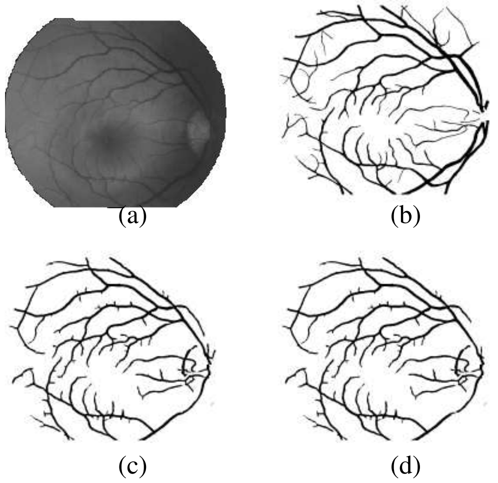

Fig. 5. (a) Green band image from STARE database, (b) manual segmentation by AH, (c) automatic segmentation by ITK serial version, and (d) automatic segmentation by ITK parallel version.

TABLE IV
COMPARISON OF SO WITH HAND-SEGMENTED SET. MEAN AND STANDARD DEVIATION (SD) OF TPR, FPR, AND AC, FOR BOTH DATABASES

STARE Database

Implementation	TPR mean(SD)	FPR mean(SD)	Ac mean(SD)	n
ITK Serial	0.779(0.066)	0.0591(0.040)	0.924(0.035)	20
ITK Parallel	0.769(0.070)	0.0551(0.035)	0.926(0.031)	20

DRIVE Database

Implementation	TPR mean(SD)	FPR mean(SD)	Ac mean(SD)	n
ITK Serial	0.660(0.056)	0.038(0.019)	0.922(0.016)	20
ITK Parallel	0.644(0.059)	0.033(0.011)	0.925(0.011)	20

The bottom part of Table IV shows the comparisons with DRIVE database. In this case, p values, again, do not show any significant difference ($p >> 0.3$).

Since ITK serial and parallel versions are implemented using the same tools, this evaluation shows that the parallel implementation performs the same segmentation outcome with an Ac value of 92%, with an improved performance compared with that of the ITK serial version.

Figure 5 shows an example of one image taken from STARE database. Figure 5(a) is the original green band, (b) the hand-segmented by AH, (c) the ITK serial, and (d) ITK parallel segmentation results.

C. Speed-Up of the Parallel Implementation

In order to obtain the speed-up, the parallel implementation is executed by gradually increasing the number of nodes used for the parallel execution, from 1 to 14, for processing a high-resolution image of size 2890×2308 pixels. The image is processed through the scales $s \in [5, 25]$. The full segmentation (feature extraction and region growing) takes less than 2 minutes, using the ITK parallel version, with 14 nodes.

Table V shows communication time (CT), processing time (PT), and total time (TT) against the number of nodes (N), for feature extraction and region growing. Notice that total TT is composed of PT and CT. CT includes both, the time required for distributing the image and the time required for collecting the results. These times are obtained using two timers. The first one at the beginning of the communication point, and the second at its respective end. The measurement of TT is similarly obtained, using timers at the beginning and at the end of the whole program. Finally, PT is obtained as the difference between TT and CT.

TABLE V
TIMES (CT, PT, AND TT) AGAINST NUMBER OF NODES.

	Feature Extraction			Region Growing		
N	*CT* (s)	*PT* (s)	*TT* (s)	*CT* (s)	*PT* (s)	*TT* (s)
1	0.0000	646.4280	646.4280	0.0000	40.2711	40.2711
2	9.8415	256.7310	266.5720	2.2427	34.3705	36.6132
3	10.4730	162.2180	172.6910	4.0459	27.1860	31.2319
4	10.9809	120.1640	131.1450	6.2991	23.2851	29.5842
5	11.1359	96.8638	108.0000	4.6305	21.7022	26.3327
6	10.7143	81.4611	92.1755	5.7617	20.0627	25.8245
7	9.9628	70.5231	80.4858	5.9725	18.6531	24.6255
8	10.8059	62.4687	73.2746	5.5059	18.8504	24.3562
9	9.3711	56.2291	65.6002	5.2227	18.3495	23.5722
10	9.3335	51.0964	60.4299	5.2931	17.2408	22.5339
11	9.2112	46.8360	56.0472	5.5995	17.2048	22.8044
12	9.4583	43.3362	52.7944	5.5864	16.8260	22.4124
13	9.3029	40.5326	49.8354	5.1995	17.1958	22.3953
14	8.9073	37.9851	46.8924	5.1635	17.1538	22.3173

The effectiveness of parallel implementation is evaluated using a relation known as *speed-up* [16]. The times from Table V are used for calculating the speed-up of the parallel version for both, *feature extraction* and *region growing*. The speed-up is defined as:

$$\text{speed-up} = \frac{T_s}{T_p} \tag{6}$$

where T_s is the processing time of the ITK serial version, and T_p is the time taken by the parallel version. Notice that T_p depends on the number of nodes.

Figure 6 shows the related speed-up of the complete segmentation procedure (*feature extraction* and *region growing*) against the number of nodes. The time includes the whole process. As parallelism is based on partitioning the image, the increment of speed-up related to the number of nodes tends to be linear. Notice that, due to the region growing algorithm is not as efficient in parallel, there is a slight decrement on speed-up from using seven nodes and onwards.

Figure 7 shows the speed-up regarding only the *feature extraction* stage, which is almost linear. This means that data partitioning is efficient for feature extraction [16]. Furthermore, communication time tends to be constant, and as the process time continuously decreases (by increasing the number of nodes), this means that it is possible to increase the number of nodes without efficiency costs.

Figure 8 shows the speed-up for parallel region growing. Nevertheless, notice that the parallel *region growing* does

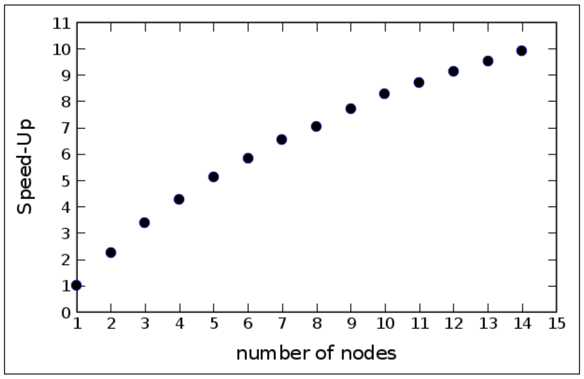

Fig. 6. Speed-Up of segmentation *vs* number of nodes.

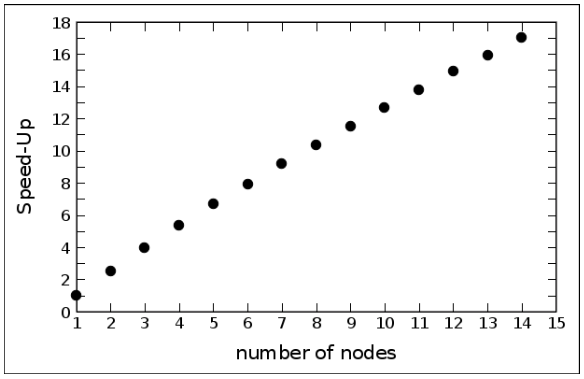

Fig. 7. Speed-Up of feature extraction *vs* number of nodes.

not result as effective as the parallel *feature extraction*. The parallel region growing implementation has a speed-up at most of 2, even using 14 nodes. This speed-up tends to be constant from 10 nodes onwards. This behavior is due to the parallelizing strategy used for this algorithm.

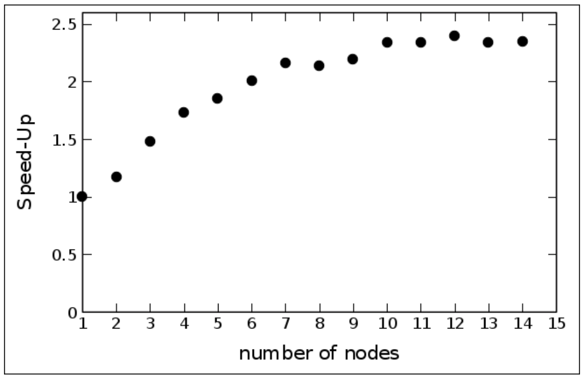

Fig. 8. Speed-Up of region growing *vs* number of nodes.

Even though the speed-up for *region growing* is less than the speed-up for *feature extraction*, considering the whole image processing application of retinal blood vessel detection, the overall segmentation for high-resolution images is processed in an acceptable time compared with any other approach in literature, as far as we are aware.

There are several reasons why the speed-up of the RG algorithm behaves in this fashion. First, the process depends on the number of seeds within a sub-image. The larger the number of seeds, the larger the number of iterations needed. Second, the vessel distribution is not homogeneous. It is likely that with the actual partition scheme there are sub-images with larger number of vessels than others. And third, since both classes may grow in any direction (vertical, horizontal or both) two sub-images are needed instead of one.

V. Conclusions

This paper introduced an *adhoc* parallel implementation for the segmentation of high-resolution images. This implementation is based on a data partitioning, which allowed a faster processing of those images.

Two data partitioning schemes were proposed and analyzed: a horizontal partitioning for *feature extraction*, and a mixed (horizontal and vertical) partitioning for *region growing*. From the results above, it is noticeable that the first partition scheme adequately works with local and linear characteristics of the *feature extraction* algorithm, whereas the latter produces inferior results due to the ordering and neighboring dependence characteristics of the *region growing* algorithm.

Originally, the Matlab implementation did not have enough memory to handle high-resolution images. On the other hand, the ITK serial version was able to segment higher-resolution images ($\approx 3500 \times 3000$ pixels size).

It has been shown that the segmentation outcome between both ITK versions (serial *vs* parallel) are statistically the same, using two public databases [8], [9]. Furthermore, the ITK parallel version has an improved performance, about 9 times faster than the ITK serial version. Both versions present a reasonable enough accuracy (92%) for blood vessel geometry measurements. The implementation presented here provides a unique setting for the transfer of these methods from the laboratory into the clinical environment,

However, the results obtained for the *region growing* algorithm are not completely satisfactory, due to the intrinsic serial part of this algorithm. Despite this, the current segmentation outcome, in terms of accuracy and computation speed, is already superior for the analysis of large amount of high-resolution retinal images than the approaches already available in the literature. Further work should be related with the development of a new parallel region growing algorithm.

References

[1] N. Witt, T. Y. Wong, A. D. Hughes, N. Chaturvedi, B. E. Klein, and R. Evans, “Abnormalities of retinal microvascular structure and risk of mortality from ischemic heart disease and stroke,” *Hypertension*, vol. 47, no. 5, pp. 975–981, 2006.

[2] T. Y. Wong, R. Klein, B. Klein, J. Tielsch, L. Hubbard, and F. J. Nieto, “Retinal microvascular abnormalities and their relationship with hypertension, cardiovascular disease, and mortality.” *Surv. Ophthalmol.*, vol. 46, no. 1, pp. 59–80, 2001.

[3] J. C. Crane, F. W. Crawford, and S. J. Nelson, “Grid enabled magnetic resonance scanners for near real-time medical image processing.” *J. Parallel. Distrub. Comput.*, vol. 66, no. 12, pp. 1524–1533, 2006.

[4] F. Zhang, A. Bilas, A. Dhanantwari, K. N. Plataniotis, R. Abiprojo, and S. Stregiopoulos, “Parallelization and performance of 3D ultrasound imaging beamforming algorithms on modern clusters.” in *ICS ’02: Proceedings of the 16th International Conference on Supercomputing*, New York, USA: ACM, 2002, pp. 294–304.

[5] M. E. Martinez-Perez, A. D. Hughes, S. A. Thom, A. A. Bharath, and K. H. Parker, “Segmentation of blood vessels from red-free and fluorescein retinal images,” *Medical Image Analysis*, vol. 11, no. 1, pp. 47–61, 2007.

[6] “Itk insight segmentation and registration toolkit,” [Online]. Available: http://www.itk.org.

[7] M. E. Martinez-Perez, A. D. Hughes, S. A. Thom, and K. H. Parker, “Improvement of a retinal blood vessel segmentation method using the insight segmentation and registration toolkit (ITK),” in *29th IEEE EMBS Annual International Conference*, Lyon, France, August 23-26 2007.

[8] A. Hoover, V. Kouznetsova, and M. Goldbaum, “Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response,” *IEEE Trans. Med. Imag.*, vol. 19, pp. 203–210, 2000.

[9] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, “Ridge-based vessel segmentation in color images of the retina,” *IEEE Transactions on Medical Imaging.*, vol. 23, pp. 501–509, 2004.

[10] X. Jiang and D. Mojon, “Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images,” *IEEE Transactions on Pattern Recognition Analysis and Machine Intelligence.*, vol. 25, pp. 131–137, 2003.

[11] D. Eberly, *Ridges in Image and Data Analysis*, ser. Computational Imaging and Vision. Netherlands: Kluwer Academic Publishers, 1996.

[12] T. Lindeberg, “On scale selection for differential operators,” in *Proc. 8th Scandinavian Conference on Image Analysis*, K. Heia, K. A. Hogdra, and B. Braathen, Eds., Tromso, Norway, 1993, pp. 857–866.

[13] R. Hempel, “The MPI standard for message passing,” in *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking Volume II: Networking.*, ser. Lecture Notes in Computer Science, vol. 797. Springer-Verlag, 1994, pp. 247–252.

[14] R. Deriche, “Recursively implementing the gaussian and its derivatives,” Unite de Recherche INRIA-Sophia Antipolis, Tech. Rep. 1893, 1993.

[15] Wilcoxon, “Individual comparisons by ranking methods.” vol. 1, pp. 80–83, 1945.

[16] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *AFIPS Conference Proceedings*, vol. 30, 1967, pp. 483–485.

Palomera-Pérez, M.A. received in 2002 a B.Sc. degree in Telecommunication Engineering and in 2005 a MSc degree in Computer Science, at the Universidad Nacional Autónoma de México (UNAM). He is currently pursuing a PhD. His research interest includes parallel and distributed systems.

Martinez-Perez, M.E. received in 1992 a B.Sc. degree in Computer Engineering and in 1996 a MSc degree in Computer Science at the Universidad Nacional Autónoma de México (UNAM), and a PhD degree by the Imperial College London, UK in 2001. Since 2002, she holds a Research position in the Department of Computer Science from the Institute of Research in Applied Mathematics and Systems, UNAM, Mexico. Her research interests include digital image processing, pattern recognition, and computer vision.

Benítez-Pérez, H. is a full time Researcher in the IIMAS UNAM (México). He received in 1994 a B.Sc. degree in Electronic Engineering by the Faculty of Engineering, Universidad Nacional Autónoma de México (UNAM), and a PhD degree by the University of Sheffield, UK, in 1999. His research interest includes real-time control and fault-diagnosis.

Ortega-Arjona, J.L. is a full time lecturer at the Department of Mathematics Faculty of Sciences, Universidad Nacional Autónoma de México (UNAM). He received a B.Sc. degree in Electronic Engineering as well as his M.Sc. degree in Computer Science at UNAM and earned his Ph.D. degree at the University College London (UCL),UK. His research interest includes parallel processing, object-oriented programming software patterns, and software architecture and design.