HPC-ICTM: a Parallel Model for Geographic Categorization

Rafael K. Santos Silva and César A. F. De Rose
PPGCC, Catholic University of Rio Grande do Sul (PUCRS)
Av. Ipiranga 6681, 906190-9000 Porto Alegre, Brazil
rafaelkrolow@terra.com.br, derose@inf.pucrs.br

Marilton S. de Aguiar, Graçaliz P. Dimuro and Antônio C. R. Costa
School of Informatics, Catholic University of Pelotas (UCPel)
Rua Félix da Cunha 412, 96010-000 Pelotas, Brazil
{marilton, liz, rocha}@atlas.ucpel.tche.br

Abstract

This paper presents a parallel version of the Interval Categorizer Tessellation-based Model (ICTM) for the simultaneous categorization of geographic regions considering several characteristics (e.g., relief, vegetation, land use etc.) in clusters, called HPC-ICTM. Interval techniques are used for the modeling of uncertain data and the control of discretization errors. We analyze the performance of the HPC-ICTM and present results concerning its application to the relief/land-use categorization of the region surrounding the lagoon Lagoa Pequena (RS, Brazil), which is extremely important from an ecological point of view.

1. Introduction

The ICTM (*Interval Categorizer Tessellation Model*) is a multi-layered and multi-dimensional tessellation model for the simultaneous categorization of geographic regions considering several different characteristics (relief, vegetation, climate, land use etc.) of such regions, which uses interval techniques [6, 9] for the modeling of uncertain data and the control of discretization errors.

To perform a *simultaneous categorization*, the ICTM proceeds (in parallel) to individual categorizations considering one characteristic per layer, thus generating different subdivisions of the analyzed region. An appropriate projection procedure of the categorizations performed in each layer into a basis layer provides the final categorization that allows the combined analysis of all characteristics that are taken into consideration by the specialists in the considered application, allowing interesting analyzes about their mutual dependency.

An implementation of the ICTM for the relief categorization of geographic regions, called TOPO-ICTM (*Interval Categorizer Tessellation Model for Reliable Topographic Segmentation*), performs a bi-dimensional analysis of the declivity of the relief function in just one layer of the model [7]. The input data are extracted from satellite images, where the heights are given in certain points referenced by their latitude and longitude coordinates. The geographic region is represented by a regular tessellation that is determined by subdividing the total area into sufficiently small rectangular subareas, each one represented by one cell of the tessellation. This subdivision is done according to a cell size established by the geophysics or ecology analyst and it is directly associated to the refinement degree of the tessellation. Applications in Geophysics and Ecology were found, where an adequate subdivision of geographic areas into segments presenting similar topographic characteristics is often convenient (see, e.g: [2, 3]).

The aim of this paper is describe a particular implementation of the model for clusters, called HPC-ICTM, extending preliminary results presented in Aguiar et al [8]. We discuss the performance of the HPC-ICTM and present some results concerning the application of a 2-layered bi-dimensional model to the *relief/land use* categorization of the region surrounding the lagoon *Lagoa Pequena* (Rio Grande do Sul, Brazil), which is extremely important from an ecological point of view.

The paper is organized as follows. Section 2 presents the ICTM model and the categorization process. The parallel implementation of the model is described in Section 3. Some results on categorizations and the performance of the HPC-ICTM is discussed in Section 4. Section 5 is the conclusion.

IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06)
0-7695-2643-8/06 $20.00 © 2006 IEEE

2. The ICTM Model

This section shows the multi-layered interval categorizer tessellation-based model, formalized in terms of matrix operations. The single-layered ICTM was firstly presented in [7][1]. Here, we present the generalization of the number of the layers and the corresponding projection procedures.

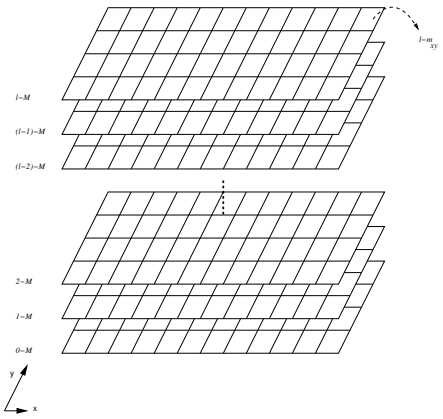

Figure 1. The ICTM multi-layered.

This type of projection allows interesting analysis of the mutual dependency of the analyzed characteristics. Each characteristic of the space is represented in a layer of the ICTM model. Thus, by the independency of the analysis, the subdivisions in each layer also occurs in a independently way. In the Figure 1, each bidimensional layer is represented by a label. All the steps of the categorization process is schematically presented at the Figure 2.

A tessellation is a matrix M with n_r rows and n_c columns. The entry at the x-th row and the y-th column is called the xy-cell of M[2].

Considering a $n_c \times n_r$ tessellation M and $l \in \mathbb{N}$, a multi-layered tessellation $\mathcal{L}$-$\mathcal{M}$ is the structure

$$\mathcal{L}\text{-}\mathcal{M} = (1\text{-}M, \ldots, l\text{-}M)$$

where the entry at the l-th layer, x-th row and y-th column is denoted by l-m_{xy}.

2.1. The interval matrices

In topographic analysis, usually there are too many data, most of which is geophysically irrelevant. We then take, for each subdivision, the average value of the heights at the points supplied by the satellite photos, which are the entries of the tessellation M, denoted by l-m_{xy}^{abs}.

We are interested in comparing the values corresponding to different cells, so we are not interested in absolute values, only in relative ones. To simplify the data of the matrix, we normalize them by dividing each l-m_{xy}^{abs} by the largest l-m_{max} of these values.

$$l\text{-}M^{rel} = \frac{l\text{-}M^{abs}}{l\text{-}m_{max}}.$$

The heights are measured pretty accurately, so the only errors in the values l-m_{xy} come from the discretization of the area in terms of the discrete set of tessellation cells. In other words, it is desirable to know the values of the relief function for all points in the space, but only the values determined by division of the region in $n_r n_c$ cells, are used in the effective calculations.

In the following, we apply Interval Mathematics techniques to control the errors associated to the cell values. To see examples of the advantages of using intervals in solving similar problems see, e.g., [2, 6].

The approximation error ϵ_x (at the coordinate x) is denoted by

$$\epsilon_x \leq \Delta_x = 0.5 \cdot \min\left(|l\text{-}m_{xy}^{rel} - l\text{-}m_{(x-1)y}^{rel}|, |l\text{-}m_{(x+1)y}^{rel} - l\text{-}m_{xy}^{rel}|\right).$$

Analogously, the approximation error ϵ_y is denoted by

$$\epsilon_y \leq \Delta_y = 0.5 \cdot \min\left(|l\text{-}m_{xy}^{rel} - l\text{-}m_{x(y-1)}^{rel}|, |l\text{-}m_{x(y+1)}^{rel} - l\text{-}m_{xy}^{rel}|\right).$$

If l-$m_{xy}^{x^{\pm}} = l$-$m_{xy}^{rel} \pm \Delta_i$ and l-$m_{xy}^{y^{\pm}} = l$-$m_{xy}^{rel} \pm \Delta_j$, the interval matrices l-$M^{x^{[\,]}}$ and l-$M^{y^{[\,]}}$, associated with the relative matrix l-M^{rel}, are defined by the $n_r \times n_c$ interval matrices

$$l\text{-}M^{x^{[\,]}} = \left[l\text{-}m_{xy}^{x^{[\,]}}\right] = \left[\left[l\text{-}m_{xy}^{x^-}, l\text{-}m_{xy}^{x^+}\right]\right] \text{ and}$$
$$l\text{-}M^{y^{[\,]}} = \left[l\text{-}m_{xy}^{y^{[\,]}}\right] = \left[\left[l\text{-}m_{xy}^{y^-}, l\text{-}m_{xy}^{y^+}\right]\right].$$

2.2. The declivity registers and the state matrix

We proceed to a declivity categorization inspired by [2]. To narrow the solution space to a minimum, we take a qualitative approach to the relief approximation functions, clustering them in equivalence classes according to the sign of their declivity (positive, negative, null), thus making the tessellation-based model build a single qualitative solution to that constraint satisfaction problem, namely, the class of approximation functions compatible with the constraints of the interval matrix. We proceed as follows:

Let l-$M^{x^{[\,]}}$ and l-$M^{y^{[\,]}}$ be interval matrices of layer l. For a given xy, if:

[1] The proofs are omitted since they are similar to those presented in [7].

[2] For the application considered in this paper, the entries of the tessellation matrices are all non-negative. However, negative values may also be considered (e.g., when the data coming from the relief are determined with respect to the sea level).

IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06)
0-7695-2643-8/06 $20.00 © 2006 IEEE

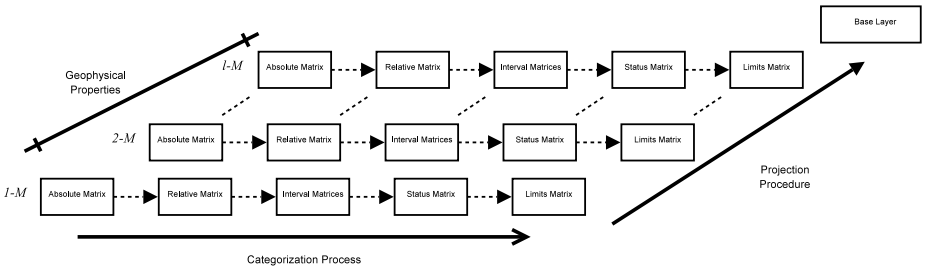

Figure 2. The categorization process performed by ICTM.

(i) $l\text{-}m_{xy}^{x^+} \geq l\text{-}m_{(x+1)y}^{x^-}$, then there exists a non-increasing relief approximation function between xy and $(x+1)y$ (direction west-east).

(ii) $l\text{-}m_{(x-1)y}^{x^-} \leq l\text{-}m_{xy}^{x^+}$, then there exist a non-decreasing relief approximation function between $(x-1)y$ and xy (direction west-east).

(iii) $l\text{-}m_{xy}^{y^+} \geq l\text{-}m_{x(y+1)}^{y^-}$, then there exists a non-increasing relief approximation function between xy and $x(y+1)$ (direction north-south).

(iv) $l\text{-}m_{x(y-1)}^{y^-} \leq m_{xy}^{y^+}$, then there exists a non-decreasing relief approximation function between $x(y-1)$ and xy (direction north-south).

For each cell, four directed declivity registers[3] – $reg.e$ (east), $reg.w$ (west), $reg.s$ (south) and $reg.n$ (north) – are defined, indicating the admissible declivity sign of the function that approximates the relief function in any of these directions, taking into account the values of the neighbor cells.

A declivity register of an xy-cell is a tuple

$$reg = (reg.e, reg.w, reg.s, reg.n),$$

where the values of the directed declivity registers are given by:

(a) For non border cells, $reg.e = 0$, if (i) holds; $reg.w = 0$, if (ii) holds; $reg.s = 0$, if (iii) holds; $reg.n = 0$, if (iv) holds; $reg.e, reg.w, reg.s, reg.n = 1$, otherwise.

(b) For east, west, south and north border cells: $reg.e = 0$, $reg.w = 0$, $reg.s = 0$ and $reg.n = 0$, respectively[4]. The other directed declivity registers of border cells are also determined according to item (a).

[3]This paper uses the dot notation of the object-oriented programming languages to represent the components of a data structure (e.g., $reg.e$ denotes the component e of the data structure reg).

[4]This is consistent with the relief function being a constant in the border cells.

The declivity register matrix of the layer l is defined as an $n_r \times n_c$ matrix $l\text{-}M^{reg} = \left[l\text{-}m_{xy}^{reg}\right]$, where the entry at the x-th row and the y-th column is the value of the declivity register of the corresponding cell.

Let $w_{reg.e} = 1$, $w_{reg.s} = 2$, $w_{reg.w} = 4$ and $w_{reg.n} = 8$ be weights to be associated to the directed declivity registers. The state matrix is defined as an $n_r \times n_c$ matrix given by $l\text{-}M^{state} = \left[l\text{-}m_{xy}^{state}\right]$, where the entry at the x-th row and the y-th column is the value of the corresponding cell state, calculated as the value of the binary encoding of the corresponding directed declivity registers, given as

$$\begin{aligned} l\text{-}m_{xy}^{state} &= w_{reg.e} \times l\text{-}m_{xy}^{reg.e} + w_{reg.s} \times l\text{-}m_{xy}^{reg.s} \\ &\quad + w_{reg.w} \times l\text{-}m_{xy}^{reg.w} + w_{reg.n} \times l\text{-}m_{xy}^{reg.n}. \end{aligned}$$

Thus, for given xy, the correspondent cell can assume one and only one state represented by the value $l\text{-}m_{xy}^{state} = 0..15$, previously defined.

2.3. The limit matrix and the constant-declivity sub-regions

A limit cell is defined as the one where the relief function changes its declivity, presenting critical points (maximum, minimum or inflection points). To identify such limit cells, we use a limit register associated to each cell. The border cells are assumed to be limit cells.

The limit matrix of the layer l is defined as the $n_r \times n_c$ matrix given by $l\text{-}M^{limit} = \left[l\text{-}m_{xy}^{limit}\right]$, where the entry at the x-th row and the y-th column is determined as $l\text{-}m_{xy}^{limit} = 0$, if the relief function no changes its declivity, and $l\text{-}m_{xy}^{limit} = 1$, otherwise.

Analyzing the limit matrix it is easy to detect the existence of known relief configurations. The presence of limit cells allows the subdivision of the whole area into declivity categories.

The constant declivity sub-region associated to the non limiting cell xy, denoted $l\text{-}\mathcal{SR}_{xy}$, is inductively defined as follows: *(i)* $xy \in l\text{-}\mathcal{SR}_{xy}$; *(ii)* If $x'y' \in l\text{-}\mathcal{SR}_{xy}$, then all its neighbor cells that are not limiting cells also belong to $l\text{-}\mathcal{SR}_{xy}$.

IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06)
0-7695-2643-8/06 $20.00 © 2006 IEEE

Observe that $l\text{-}\mathcal{SR}_{xy} = l\text{-}\mathcal{SR}_{x'y'}$ if and only if $x'y' \in l\text{-}\mathcal{SR}_{xy}$ (resp., $xy \in l\text{-}\mathcal{SR}_{x,y,}$). The above definition leads to a recursive algorithm similar to the ones commonly used to fill polygons in computer graphics.

3. Parallel Implementation

This section describes the parallel implementation of the ICTM for clusters, called *High Performance Computing Interval Categorizer Tessellation-based Model* (HPC-ICTM).

The first step to implement a parallel version of ICTM is to identify how the model can be broken in independent tasks. We explored four possibilities for problem decomposition: *Layers* - each parallel process calculates a determined layer of the model; *Functions* - each parallel process calculates a independent function of the model; *Domains* - each parallel process calculates part of the region that will be analyzed; *Cells* - each parallel process calculates individual cells of the model.

Since in clusters architectures the interconnection network can be considered bottleneck (the cost of local computation is much cheaper then communication with neighbors nodes) we are interested in a problem decomposition that results in big chunks of work with low communication overhead (coarse grain [13]).

Considering this granularity issue, decomposition in layers is the more simple and direct way to implement a parallel version of the ICTM. Being each process responsible for a determined layer of the model, these processes perform the ICTM calculations for each property in parallel.

The next step is to define how the processes will be coordinated to solve the problem. We implemented, using the MPI (*Message Passing Interface*) [12] library, a master-slave scheme [13] (Figure 3). The master process is responsible for loading the input files and parameters (the data and the radius), divide total work in nl tasks (nl is the number of layers that will be processed), send the radius value and the tasks for all slave processes to start the categorization process, and keep control of the tasks. The slave processes receive the information sent by the master process, execute their tasks and generate their own outputs. The directory with input and output files is in the same file system, being accessible by all the cluster nodes. After that they ask the master for more work. Until there is work to do the master keep sending tasks to the slaves.

Decomposition in layers follows the rule $np = nl + 1$, where np indicates the number of processes and nl indicates the number of layers or properties that need to be processed, considering each slave process running in a different node. Thus, each layer is analyzed by a different slave process and the remnant process is the master process.

Communication occurs between master and slaves (not among slaves) and only (i) when the master process sends the radius value and a layer for a slave to compute, and (ii) when a slave notifies the master that finished the computation of a layer.

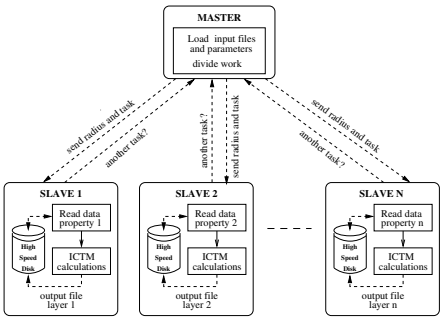

Figure 3. Master-slave scheme to solve the problem of decomposition in layers.

Each slave allocates only the amount of memory needed to calculate one layer/property of the model. However, if the main memory size of a cluster node is insufficient to execute one layer of the model, the application will access the hard disk (swap) or, in the worst case, it will abort. In these cases, the problem decomposition in layers is not recommended.

4. Performance Analysis

This section presents the *relief* and *land use* categorizations obtained for the region surrounded the lagoon *Lagoa Pequena* (Rio Grande do Sul, Brazil). These analyzes are to be used for the environment characterization of that region, aiming to give subsidies for its integrated preservation/management. Figure 4 shows the location of the lagoon and a *land use* categorization of the region surrounded it, which shall be combined with relief categorizations.

Figure 4. LANDSAT image – Land use Map of the region surrounded the Lagoa Pequena.

For the performance analysis, we consider three differ-

IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06)
0-7695-2643-8/06 $20.00 © 2006 IEEE

ent portions of the region surrounding the Lagoa Pequena, each with a type of tessellation matrix: *Portion A* with 241 rows and 241 columns; *Portion B* with 577 rows and 817 columns; and *Portion C* with 1309 rows and 1765 columns.

The results were obtained in two clusters of the CPAD[5], each one with the following configuration:

32 bits cluster: composed of 8 machines with two Pentium III 32 bits 1Ghz processors and 256MB of main memory (HP $e800$ Server), interconnected by a Myrinet network.

64 bits cluster: composed of 5 machines with two Itanium 2 64 bits 1.5Ghz processors and 2GB of main memory (HP Integrity rx2600 $IA64$), interconnected by a Fast-Ethernet network.

Both clusters have the following software configuration: MPICH implementation of the MPI Library; GNU Linux Operating System installed in each node (Debian distribution); GNU compiler collection (gcc 3.3.5). Further, we utilized also the Intel compiler (icc 8.0) in the 64 bits cluster nodes.

Table 1 presents the results of a sequential implementation of the model processed in a $e800$ and $IA64$ machine, the results of parallel implementation processed in the above clusters considering the rule $np = nl + 1$, and also the obtained *speedups* in each case. All measurements use the same radius value (*radius = 1*).

Notice that, increasing the input data set and/or the number of layers, the sequential execution time and the amount of memory necessary to execute the model also increases because the layers are processed sequentially in a only machine, being their outputs generated in the same text file on the current directory. For the test case of the portion C in the $e800$ machines with 3 layers, the amount of free memory in one node was not sufficient and the application needed to make accesses to the hard disk to calculate the model (swap). This of course increased the execution time considerably. With 5 and 7 layers, the memory needed by the application was much greater then the amount of free main memory in one node, and the operational system aborted the application. An advantage of the parallel implementation of the ICTM running in a cluster is that in some cases it can solve bigger problems since the calculation is not limited by the main memory size of the sequential machine.

Furthermore, even when the number of layers increased (3, 5 and 7), there is no significant difference in the parallel execution time. This behavior proves the effectiveness of this parallel version of the ICTM.

Figure 5 shows the ICTM behavior for the portion B of the region surrounding the Lagoa Pequena, considering 7 layers, $e800$ and $IA64$ machines, and different np values. We utilized the $IA64$ sequential results as the reference value to measure the speedups. This decision had a great impact on the speedups of the 32 bits cluster, being a gain of performance only observed with more then 4 processors. Not only the better performance of the 64 bits processors and the bigger main memory contributed to this gap but also the better optimization of Intel compiler used in the Itanium machines (icc).

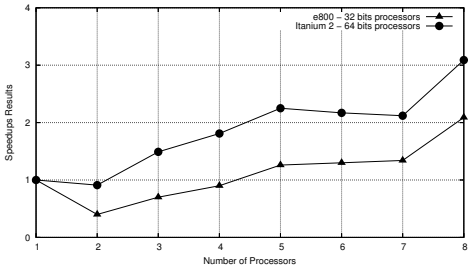

Figure 5. Parallel behavior of the ICTM for portion $B_{(577\times817)}$ with 7 layers.

The parallel implementation may be considered when the ICTM presents more than one layer. However, it becomes really necessary in the case that it has a great amount of layers or when the input data set of the layers is very big (analysis of big regions), since, in this case, a sequential implementation is practically not feasible.

5. Conclusions and Future Work

In the categorizations produced by the ICTM, the state of a cell in relation to its neighbors, concerning the declivity, is shown directly by arrows (see Figure 6), which has been considered a very intuitive representation, by the ecologists, since most geographic information systems present this kind of result by the usual color encoding of declivity, with no indication of direction.

The ICTM is regulated by two aspects, namely, the spatial resolution of the image (used data) and the neighborhood radius of the cell. Thus, regions with an agglomeration of limiting cells can be studied with more details by just increasing the resolution of altimetry data, or reducing the neighborhood radius. In plain areas (see Figure 6 (region A)), a large neighborhood radius indicated reasonable approximations for the declivity degree. However, regions with too much declivity variation (see Figure 6 (region B)) obtained good approximations only with small radius. The number of categories obtained is always inversely proportional to the neighborhood radius and to the area of a tessellation cell.

[5]Research Center of High Performance Computing of PUCRS/HP, Brazil, RS – http://www.cpad.pucrs.br.

Portion	nl	32 bits machines ($e800$)			64 bits machines ($IA64$)		
		Sequential Results	Parallel Results	Speedup Results	Sequential Results	Parallel Results	Speedup Results
$A_{(241\times241)}$	3	1.909s	2.085s	0.92	1.119s	1.711s	0.65
	5	3.069s	2.285s	1.34	1.827s	2.378s	0.77
	7	4.287s	2.840s	1.51	2.516s	2.630s	0.96
$B_{(577\times817)}$	3	15.885s	7.119s	2.23	7.188s	4.013s	1.79
	5	26.777s	7.437s	3.60	11.979s	4.557s	2.63
	7	37.045s	8.055s	5.00	16.818s	5.568s	3.02
$C_{(1309\times1765)}$	3	4714.042s	28.621s	164.71	33.652s	13.564s	2.48
	5	aborted	30.627s	no value	56.226s	14.014s	4.01
	7	aborted	31.005s	no value	79.568s	14.433s	5.51

Table 1. Results of sequential and parallel implementation in a 32 bits and 64 bits cluster ($np = nl + 1$ for the parallel versions).

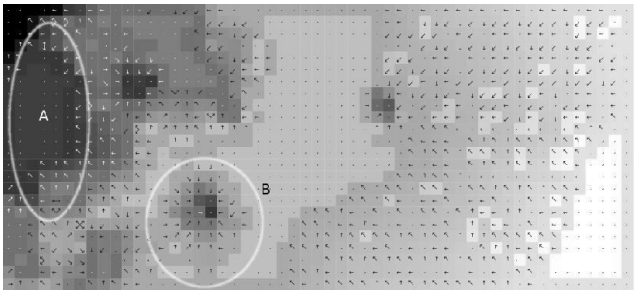

Figure 6. Relief categorization of a portion of LANDSAT image.

The analysis of some related works concerning image segmentation [1, 4, 5, 10] turns out that those methods are, in general, heuristic, and, therefore, the ICTM model presented here is more reliable (for other works, see, e.g.: [2, 11]).

The HPC-ICTM significantly reduced the execution time for our test cases, minimizing the amount of memory necessary to perform the model and allowing the analysis of bigger geographic regions.

This parallel implementation of the ICTM is being refined and the other presented alternatives for problem decomposition are being investigated. Furthermore, a parallel version of the ICTM for computational grids is being developed, that will allow higher speedups and the utilization of distributed geographic information.

References

[1] M. C. Cooper. The Tractability of Segmentation and Scene Analysis. *International Journal on Computer Vision*, 30(1):27–42, 1998.

[2] D. Coblentz, V. Kreinovich, B. Penn and S. Starks. Towards reliable sub-division of geological areas: interval approach. In L. Reznik and V. Kreinovich, editors, *Soft Computing in Measurements and Information Acquisition*, pages 223–233. Springer-Verlag, 2003.

[3] R. T. T. Forman. *Land Mosaics: the ecology of landscapes and regions*. Cambridge University Press, Cambridge, 1995.

[4] K. S. Fu and J. K. Mui. A Survey on Image Segmentation. *Pattern Recognition*, 13(1):3–16, 1981.

[5] J. L. Lisani, L. Moisan, P. Monasse and J. M. Morel. On The Theory of Planar Shape. *Multiscale Modeling and Simulation*, 1(1):1–24, 2003.

[6] R. B. Kearfort and V. Kreinovich, editors. *Applications of Interval Computations*. Kluwer, Dordrecht, 1996.

[7] M. S. Aguiar, G. P. Dimuro and A. C. R. Costa. TOPO-ICTM: an interval tessellation-based model for reliable topographic segmentation. *Numerical Algorithms*, 34(1):3–11, 2004.

[8] M. S. Aguiar, G. P. Dimuro, F. A. Costa, R. K. S. Silva, C. A. F. De Rose and A. C. R. Costa. `HPC-ICTM`: the Interval Categorizer Tessellation-Based Model for High Performance Computing. In *Workshop on State-of-the-art in Scientific Computing (PARA'04)*, Lecture Notes In Computer Science, pages 1–10. Springer-Verlag, 2005. (to appear).

[9] R. E. Moore. *Methods and Aplication of Interval Analysis*. SIAM, Philadelphia, 1979.

[10] S. E. Umbaugh. *Computer Vision and Image Processing*. Prentice Hall, New Jersey, 1998.

[11] K. Villaverde and V. Kreinovich. A Linear-Time Algorithm that Locates Local Extrenma of a Function of One Variable from Interval Measurements Results. *Interval Computations*, pages 4:176–194, 1993.

[12] W. Gropp, E. Lusk and A. Skjellum. *Using MPI: portable Parallel Programming with the Message-Passing Interface*. MIT Press 2nd ed., 1999.

[13] B. Wilkinson and M. Allen. *Parallel Programming: techniques and applications using networked workstations and parallel computers*. Prentice-Hall, Upper Saddle River, New Jersey, 1999.

IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06)
0-7695-2643-8/06 $20.00 © 2006 IEEE