Parallel Image Processing Using a Pure Topological Framework

Fernando Diaz-del-Rio[1], Helena Molina-Abril[2], Pedro Real[2], Pablo Sánchez-Cuevas[1], Antonio Ríos-Navarro[1]

Abstract— **Image processing is a fundamental operation in many real time applications, where lots of parallelism can be extracted. Segmenting the image into different connected components is the most known operations, but there are many others like extracting the region adjacency graph (RAG) of these regions, or searching for features points, being invariant to rotations, scales, brilliant changes, etc. Most of these algorithms part from the basis of Tracing-type approaches or scan/raster methods. This fact necessarily implies a data dependence between the processing of one pixel and the previous one, which prevents using a pure parallel approach. In terms of time complexity, this means that linear order *O(N) (N* being the number of pixels) cannot be cut down. In this paper, we describe a novel approach based on the building of a pure Topological framework, which allows to implement fully parallel algorithms. Concerning topological analysis, a first stage is computed in parallel for every pixel, thus conveying the local neighboring conditions. Then, they are extended in a second parallel stage to the necessary global relations (e.g. to join all the pixels of a connected component). This combinatorial optimization process can be seen as the compression of the whole image to just one pixel. Using this final representation, every region can be related with the rest, which yields to pure topological construction of other image operations. Besides, complex data structures can be avoided: all the processing can be done using matrixes (with the same indexation as the original image) and element-wise operations. The time complexity order of our topological approach for a *m*×*n* pixel image is near *O(log(m+n))*, under the assumption that a processing element exists for each pixel. Results for a multicore processor show very good scalability until the memory bandwidth bottleneck is reached, both for bigger images and for much optimized implementations. The inherent parallelism of our approach points to the direction that even better results will be obtained in other less classical computing architectures.**

Keywords — **Topology, Component-Labeling, Adjacency Tree, Image Processing, Parallelism.**

I. INTRODUCTION

The starting point in our work is that topology is the ideal scenario for promoting parallelism, although it drives to less classical approaches. The nature of the topological properties goes necessarily from local-to-global relations. The power of our method resides in that topological magnitudes are, by definition, robust under deformations, translations and rotations.

Up to now, the only topological invariant that has been calculated using a fully parallel computation is the Euler number [3]. Other authors have recently proposed other parallel algorithms that compute some aspects of the homological properties of binary images [13]. In [4], a digital framework for parallel topological computation of 2D binary digital images based on a sub-pixel scenario was developed, modeling the image as a special abstract cell complex [11], in order to facilitate the generalization of this work to images of higher dimensions. Still, topological approaches in that sense are rare in the literature.

Within a purely discrete level, combinatorial versions of CW-complexes, called abstract cell complexes (ACC, for short [11]), can be used for a correct algorithmic development. They are formed of basic elements (representing the cells using topological coordinates) of different dimensions together with a bounding function describing the combinatorial relationship "to be in the boundary of". Different definitions of ACCs can be found in the literature (see [29] for a thorough survey).

To sum up, we construct our scaffolding on the basis of the two following basic topological properties: "being adjacent to" and "being surrounded by". Moreover, we take advantage of the powerful duality properties that the topological invariants of connected components and holes have in the context of 2D binary digital images based on square pixel. In other words, we exploit the duality that the holes of 4-adjacent CCs (connected components) must be 8-adjacent CCs and vice versa (see Fig. 1). Finally, our algorithms use only trees as their basis. Each CC is then described by only one tree, which is connected to another CC tree by only one edge. Our framework allows us to extend the parallelism to every single pixel, in such a way that all of them do the same operations without any real dependence among them.

When writing the code, we must carefully estimate the number of operations, the memory consumption, and, the most important aspect, the ratio of memory accesses per pixel, if a fast execution is required. In fact, this last parameter is in many occasions a measure of the final algorithm performance [23].

In this paper we summarize how a pure topological framework can extend because the degree of parallelism to every single pixel. These novel image processing methods can sensibly decrease computation times if enough PEs (Processing Elements) were available.

II. RELATED WORKS

In relation to the representation of digital objects or, alternatively, binary digital images, various topological models have been exhaustively used. Adjacency trees (also called topological, inclusion or homotopy trees [2, 16, 17], and here AdjT, for short) offer a classical region-based representation in terms of rooted tree of

[1] Department of Computer Architecture and Technology. University of Seville. Spain.
[2] Department of Applied Mathematics. University of Seville. Spain.
Corresponding author: fdiaz@us.es

certain topological and spatial properties of the connected components in a binary image. Within an AdjT, each node represents a distinct foreground (FG) or background (BG) component, and an edge between two nodes means that one of them is surrounded by the other. The root in an AdjT always represents the unique BG component “surrounding" the image (if it does not exist, it can be artificially created) and two 2D binary digital images are topologically equivalent if and only if their AdjTs are equivalent. An example of an AdjT of the binary image in Fig. 1 (Left) is shown in Fig. 1 (Right).

Aside from image understanding [18] and mathematical morphology applications [7, 10, 15], AdjTs have encountered exploitation niches in geoinformatics, dermatoscopics image, biometrics, etc. (see [3, 5, 6] for instance). Therefore, finding fast algorithms for segmenting and computing the AdjT of a 2D digital binary image is crucial for solving important problems related to topological interrogations in the current technological context. It is evident that the compression of those nodes of a CCL tree (CCLT) satisfying the neighboring condition “having the same color", directly yields to the AdjT.

Connected component labeling (CCL) of binary images is one of the fundamental operations in real time applications, like fiducial recognition [6] or classifying objects as connected components (CCs). The labeling operation transforms a binary image into a symbolic matrix in which every element (pixel) belonging to a connected component is assigned to a unique label. Currently, there are mainly four classes of CCL algorithms: Multi-scan algorithms, Two-scan algorithms, Tracing-type algorithms and Hybrid algorithms mixing the previous ones. All of them (including the fastest one) use raster or tracing-type approaches, scanning the whole binary image or its contours in a sequential manner. For instance, they can label the first pixel; and then, the second one is labelled as a function of the first pixel label. This local processing runs progressively until the last pixel is reached. This fact necessarily implies real data dependencies between the labeling of one pixel and the previous one, which restricts from using a pure parallel approach. In terms of time complexity, this means that linear order *O(N)* (being *N* the number of pixels) cannot decrease independently of the number of available processing units.

Implementations for computing topological magnitudes can be achieved using classical approaches. These algorithms would contain two main stages: *1)* the scanning phase where provisional labels are sequentially assigned to pixels depending on their neighbors, *2)* and some kind of union-find technique [33] to detect and process label equivalence information of the previous assignment. Still, there is some space for parallelism when codifying scan or tracing-based CCL algorithms.

For example, dividing the image into strips is a classical data partition technique for obtaining parallelism. The second stage must then use a more sophisticated union-find technique for the provisional labels to get to the CCL. Using this classical *divide-and-conquer* approach, many works have addressed different implementations [8, 10, 15] including tuning parallel algorithms for specific computers [1]. The issue is that this division necessarily implies more data dependences between the strips in which the original image was divided (it makes harder the union-find stage). Thus, a pure parallel approach is not allowed.

Other interesting topological representations of digital images are appearing in the last years, thus leading to successful applications, for instance, in the field of image registration and matching. Most of them are hierarchical representations, which can be categorized into two classes: inclusion trees and partition trees. Leaves in inclusion trees are often image extrema, and inner nodes are formed by region growing from the leaves until the root which covers the whole image. In general, any cut of an inclusion tree does not form a complete partition of the underlying image. Typical examples are Max- and Min-tree and Tree of Shapes, which combines both of them [27]. Partitioning trees, on the other side, are initialized from an image partition. Then they rely on iterative merges of small regions at finer scale into larger regions at coarser levels. One of the most commonly used are Binary Partition Trees (BPT), α-trees and ω-trees [28]. More concretely, the α-tree, was first introduced to avoid relying on an ordering relation among image pixels (as in Max- and Min-trees). It is based on representing quasi-constant color regions of the original image.

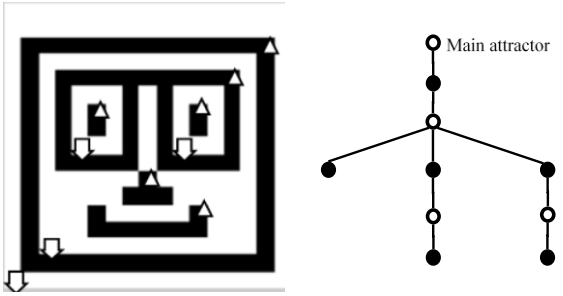


Fig. 1. Left: possible black attractors (little triangles) and white attractors (downwards arrows) for a face-like image. Holes of 4-adjacent CCs are 8-adjacent CCs and vice versa for 2D binary digital images. Right: AdjT (Adjacency tree) of the image. The main attractor is the representative of the white component that surround the whole image.

III. Topological Approach

Topological analysis of digital images studies their degree of connectivity, defining specific adjacency relations between pixels as “local neighborhood measures”. Thus, connectivity and adjacency are the key concepts in topological methods. Correctness of a framework prevents from paradoxes when an image representation is pursued. In relation to topological frameworks (Homological Spanning Forest, HSF in our case), we take advantage of the powerful duality and isotopic properties that the topological invariants of connected components and holes have in the context of 2D binary digital images based on square pixels. In fact, our method starts from an AdjT at pixel’s level and computes an AdjT at CC’s level. Let us develop this notion with a simple example.

When an object is discretized into a 2D image, it is obvious that all pixels can be linked as a tree, simply by connecting adjacent pixels using some trivial criterion for all of them (Fig. 2, (1)). For instance, the edge goes

to the South, if not possible to the West (we call it a South-West or simply SW-criterion). Note that this tree can be built independently of pixel colors. To sum up, we can state that the whole 2D image can be represented by one root pixel, which we call "attractor" (because it will attract the rest of pixels when building the tree in our framework).

Let us introduce two objects in the image, having different colors to distinguish them. According to the previous consideration, each object must be represented by only one tree, which means that one of the objects must contain the attractor of the whole image, and the other object can be represented by an attractor that falls into the previous object (Fig. 2, (2)). Conditions for one pixel to be considered being an attractor depends on adjacency criteria. Most common criteria in labelling algorithms are 8-adjacency for black pixels and 4-adjacency for white ones. In this case, tree edges connecting two pixels of different colors are candidates to attractors (Fig. 3).

Still, guessing the correct directions so that each object can be embodied by a tree cannot be achieved by local criterions. Instead, we need a global knowledge of the objects to find the correct direction for every pixel; this is patent for a spiral-like object. For instance, if we used a NE criterion for dark and SW for white pixels, cycles can appear. This is the case of Fig. 2, (3), where a simple black 'L' shape produces an undesirable situation: both black and white objects have two attractors. Thus a cycle comes out.

Thus, a meticulous strategy that allows a fast detection of incorrect directions and their parallel corrections must be found. For two dimensional digital images this can be achieved in two stages: 1) using a NE (North-East) criterion for dark pixels and a SW one for the white pixels; 2) Once a cycle is detected, transport two edges so that we get to the correct HSF (having one tree per object) (Fig. 2, (4)). Dashed arrows are the new transported edges.

To sum up, edges that connect different colors are candidates (attractors) of frontiers between CCs. False attractors (in case a cycle is detected) can be transported to get to the correct HSF. In the end, any tree covering the image plus the region frontier candidates is an instance of a connectivity tree that holds the complete information of the image. Further details about how to implement a fully parallel implementation of this process are detailed in next sections.

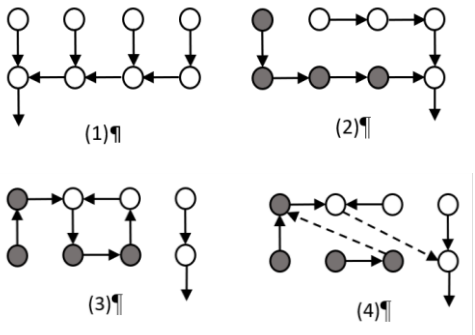


Fig. 2. (1) Any 2D image can be connected as a single tree. (2) Guessing the correct directions so that each object is a tree (3) Using NE criterion for dark and SW for white pixels can produce cycles. (4) A transport pair to get the correct HSF (one tree per object).

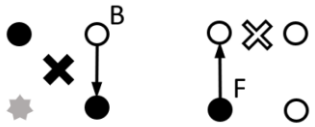


Fig. 3. The two unique possible patterns for attractors. Left: white attractor that is connected to an 8-adj black set of pixels. Right: black attractor connected to a 4-adj white set of pixels. Grey star represents a pixel of any color

IV. PARALLEL PROCESSING KEYS

For the sake of clarity, from now on, this paper concentrates on the parallel procedure to label a B/W image of mxn pixels (the problem known as CCL, Connected Component Labelling). Current CCL solutions are fully sequential on their first stage. That is, the provisional label of a pixel is written as a function of some set of the previous one (Fig. 4). In fact the strength of fastest CCL algorithm (according to YACCLAB [22]) resides on the use of a big window of neighboring pixels and a very ingenious way to reduce the hundreds of combinations of this window into a few dozens of cases (a Decision Tree or Table strategy) [24].

After the sequential stage, a 'Union-Find' phase combines and relabel those labels that are detected to belong to the same CC. This will be the case of labels 1 and 3 in Fig. 4; their label equivalence would be discovered when approaching the most South-East black corner.

1	1	2	2	2	2	3
1	1	1	2	2	2	3
4	4	1	?			

Fig. 4. A B/W image showing a sequential labelling (using a South-East direction). When a new CC is found, it is assigned a new incrementing label. Next pixel (e.g. that marked with '?') must be labelled as a function of the previously labelled pixels.

Our previous topological framework allows to build in a fully parallel manner all the labelling. Instead of assigning a non-meaningful label to each pixel, we can set them with the jump distance to their attractor. Thus, true attractors are to be set as 0, whereas false attractor (after a transport) will be given a jump distance to the corresponding attractor (see Fig. 5). There are two main phases to proceed with the jump distance computation.

+1	0	-1	+4
-4	+1	0	0

+1	+6	-1	+4
-4	+1	-2	0

Fig. 5. Left: Jump distances of Fig. 2, (3). A linear address distance is followed, being the jump of +/-4 a change of row (because image has 4 columns). Right: New jump distances assigned to false attractors after the transport is done.

A. *First stage: from local to global jump distance.*

In the first stage, every pixel (in parallel) computes its jump distance to its attractor. This can be done by using exponentially growing jump distances in a logarithm number of iterations. Fig. 6 shows an example for 9 adjacent pixels, which can be completed after three iterations. Arrows expresses the memory accesses that every pixel must do. For each reading, each pixel must

add its previous distance with the new read value. As far as we know, the first work that proposed a similar scheme was [25] with the purpose of producing highly efficient Monte Carlo simulations for two and three-dimensional critical Ising models.

Similar procedures can be extended for any dimension. Further details for applying this phase to 2D binary images can be found in our previous paper [26], which it is shown that this phase can be executed in *log2(m+n-1)* iterations at most, due to its exponential nature. Supposing that we have *p* PEs, their complexity is *O((mn/p)log(m+n))*, thus being this phase usually the most time consuming.

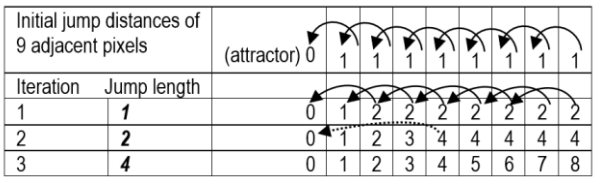


Fig. 6. Three iterations of parallel jump distance computation for 9 adjacent pixels. Most left pixel is the attractor. Arrows expresses the memory accesses that every pixel must do (only one pointed arrow is depicted for the second iteration for clarity purposes).

B. *Second stage: transports.*

The second stage consists of the parallel transports of pairs of false black and white attractors. This supposes a transport of edges until a unique tree existed, and no cycle remains. Any transport implies the updating of two jump matrix elements (or equivalently, redirecting two edges for each pair of false attractors). If black and white pixels conformed tree structures and followed different directions when doing previous jump distance computation, all transports can be done in parallel if the next *concurrency condition* is detected for each possible transport. This perfect concurrency of executing many transports is guaranteed since for each transport there are two travels through trees up to their corresponding roots (attractors). The condition that the beginning pixel must be the same as the destination after the two tree travels ensures the unicity of the pair to be cancelled (Fig. 7). Hence transport phase has no need for any critical section or atomic clause. This cannot be ensured in classical Union-Find techniques.

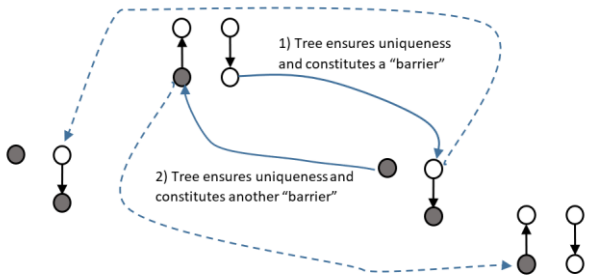


Fig. 7. Condition for guarantying the perfect concurrency of executing many transports in parallel. First, from the above black attractor a tree ensures finding a unique white attractor, which additionally constitutes a "barrier" of white pixels. Going back (step 2) from the black adjacent pixel to the white attractor also ensures unicity. Finally, dashed arrows represent the new jump distances to be computed after the transport is done.

This stage must execute several pairs of cycle searching. Although this phase seems to be tricky, if there were more PE than attractors, its timing complexity is reduced to a few iterations. In [26] it is found that the number of iterations reached a maximum of six pairs even for the most problematic images (big random images -16 Mpixels- having a 50% of black pixels). Conversely, it was only one for the real images tested (having a size until 2 Mpixels). The worst-case scenario of this phase is left for future research.

At the end of these two stages, we get to a new representation of the 2D image, in which any matrix element contains a jump distance to its true attractor, that is, to the root of the tree that represents the whole CC (see Fig. 10). Obtaining the AdjT is quite straightforward; simply by looking for each attractor the jump distance of its adjacent opposite color pixel (which goes to a new attractor).

Jump distance information is the basis for many other topological representations; some of them are shortly discussed in section VI.

V. Experimental Results

Two complete implementations were done in C++/OpenMP. The first was a direct translation of a previous MATLAB/OCTAVE implementation presented in [26]. The second is a more optimized version, whose results have been submitted to the journal 'Pattern Recognition Letters'. The server where tests were carried out was an Intel Xeon E5 2650 v2 with: 2.6 GHz, 8 cores, 8x32 KB data caches, Level 2 cache size 8x256 KB, Level 3 cache size 20 MB, maximum RAM bandwidth: 59.7 GB/s. Experiments were run 25 times and mean times were collected.

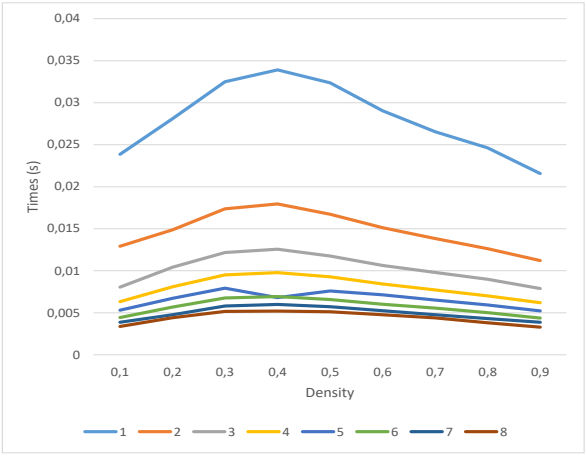


Fig. 8. Times for 1 to 8 threads as a function of density (random images).

For the first implementation, when optimization is poor, the scalability is very high. Thus, speedup (time for various threads divided by time for 1 thread) is near the number of threads (Table I), which points out that achieved scalability is excellent for all image sizes and densities. Fig. 8 depicts times for a set of 512x512 pixel images with different densities, showing that processing times are very near to that of current fastest algorithm [22]. Taking into account the good scalability, we expect that our implementation ran even faster in a massive

multicore processor. Although scalability is a little inferior for real images (Fig. 9) than for random images, times are much smaller. In fact, the processing time for the only random image ("633.png", 4196 Kpixel) in this figure is even bigger than that of a real image with a double size. This is due to the higher amount of CC that random images usually have (in relation to the real ones).

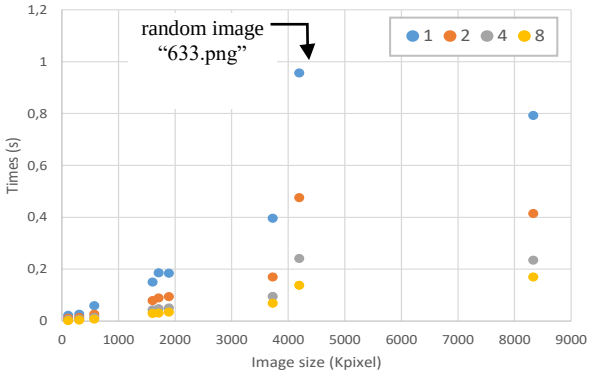


Fig. 9. Times for 1, 2, 4, 8 threads for real images.

TABLE I

SPEEDUP FOR RANDOM IMAGES OF DIFFERENT SIZES (DENSITY = 0.9).

#threads	256x256	512x512	1024x1024	2048x2048
2	1,88	1,92	1,94	1,95
3	2,65	2,73	2,89	2,82
4	3,39	3,48	3,79	3,61
5	3,92	4,13	4,57	4,33
6	4,45	4,95	5,30	4,98
7	5,56	5,59	5,94	5,55
8	5,93	6,58	6,48	6,19

TABLE II

MEAN TIMES FOR RANDOM IMAGES OF DIFFERENT SIZES AND DENSITIES. BBDT IS FROM [22][23] AND HSF IS OUR METHOD (#THREADS IN BRACKETS).

Size	BBDT	HSF (1)	HSF (2)	HSF (3)	HSF (4)
1024	0,009	0,023	0,100	0,095	0,127
4096	0,029	0,064	0,144	0,130	0,158
16384	0,102	0,221	0,296	0,243	0,268
65536	0,374	0,839	0,896	0,624	0,666
262144	1,444	3,305	3,040	2,229	2,129
1048576	5,706	13,239	10,888	7,660	7,229
4194304	23,662	65,644	42,131	32,372	28,553
16777216	117,962	338,320	210,366	154,011	129,632

Size	BBDT	HSF (5)	HSF (6)	HSF (7)	HSF (8)
1024	0,009	0,128	0,106	0,110	0,114
4096	0,029	0,149	0,129	0,135	0,127
16384	0,102	0,235	0,199	0,203	0,189
65536	0,374	0,546	0,452	0,410	0,397
262144	1,444	1,885	1,391	1,252	1,180
1048576	5,706	6,384	4,811	4,305	3,836
4194304	23,662	25,878	20,275	18,742	18,170
16777216	117,962	110,767	90,458	80,713	78,685

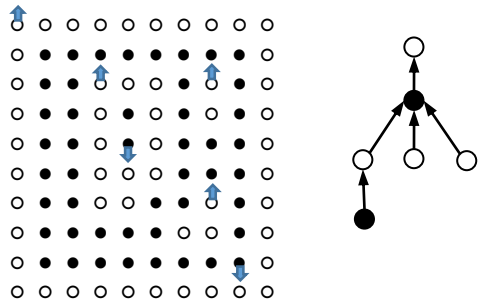
Besides, Table II shows the results from the second (optimized) version of our method compared with the BBDT method, which is the currently fastest CCL algorithm according to [22]. The optimization of our code introduces more than 7x speedup with respect to the timing of Fig. 8, but decreases the multithread speed-up to only 4x for 8 threads. Of course, for little images the extra overhead time (introduced by OpenMP when creating the threads) hinders speedup. This supposes that speedups are also decreased for medium images.

For this second optimized implementation, we can beat the fastest sequential CCL algorithm when executing on a convenient number of cores (in general, 5 or 6 threads in our experiments with medium/big random images, see Table II). However, scalability begins to be less high because data accesses come to be a bottleneck.

Finally, an additional advantage of our approach is that it presents lower deviation for a same size and different densities than the BBDT method. This is manifest when processing images of very different textures.

VI. FUTURE WORK: OTHER TOPOLOGICAL REPRESENTATIONS

Jump distances define another image representation that allows to obtain topological measures straightforwardly (Fig. 10).



0	-1	-2	-3	-4	-5	-6	-7	-8	-9
-10	77	76	75	74	73	72	71	70	-19
-20	67	66	0	-1	-2	62	0	60	-29
-30	57	56	-10	10	-12	52	-10	50	-39
-40	47	46	-20	0	-22	42	41	40	-49
-50	37	36	-30	-31	-32	32	31	30	-59
-60	27	26	-40	-41	1	22	9	20	-69
-70	17	16	15	14	13	0	-1	10	-79
-80	7	6	5	4	3	2	1	0	-89
-90	-91	-92	-93	-94	-95	-96	-97	-98	-99

Fig. 10. Up Left: A 10x10 B/W image Attractors are marked with upwards (for the white) downwards (for the black) arrows. Up Right: Its AdjT. Bottom: The Jump distance matrix. Black pixels follow a SE criterion (positive values in general) and white ones a NW (negatives values in general). Attractors are assigned a value 0 (highlighted with shadow).

Going further, digital images of any dimension and with multiple object inside (that is, color images) requires more powerful topological description. In this case, we can benefit from other duality topological properties, like object/border. This concept needs to declare a convenient abstract cell complexes (ACC, [11]) for dealing with color images. Exploration of this approach demonstrates that two dimensional color images can be treated with 4 cells per elemental PE [4], using cells of dimensions 0, 1 and 2. An example of a color image and an elemental PE is found in Fig. 11. Each PE covers a pixel in a digital image and can hold the information related to flat color zones and their borders (called cracks in [4]), that is, a region-contour

HSF [31]. That is, a complete image can be composed as many trees as correlative dimensions (*0-1* tree and *1-2* tree for the case of Fig. 11). Region-contour information can be seen in Fig. 11 (Right) as a set of 0, 1, 2 cells for flat zones (regions) and a selected set of 1, 2 cells drawing the region interfaces (contours as black segments).

For doing so, we must proceed in a similar manner to that explained previous sections, that is, cells (having their topological coordinates) of different dimensions must be joined together with a bounding function, and a combinatorial optimization process must compress the whole image into just one pixel. The relationship "to be in the boundary of" for the different regions must be efficiently computed and stored to preserve the topological information of the image. Each of these relations can be seen as a division of one tree into several sub-trees (Fig. 11, Right).

Besides, using this previous topological information, a potential idea consists of introducing color order relations among sub-trees to extract more sophisticated features. In this sense, during the last decade, features based of pure topological relations (Max- and Min-tree, Tree of Shapes, Binary Partition Trees, α-trees and ω-trees, etc.). Recently the so-called Tree-Based Morse Regions (TBMR, [32]) determines local invariant "interest" points, with the same complexity as classical MSER (Maximally Stable Extremal Regions), and a repeatability on par with state-of-the-art methods. In addition, it obtains a significantly higher number of features, being both accurate and robust enough to be applied to image registration and 3D reconstruction.

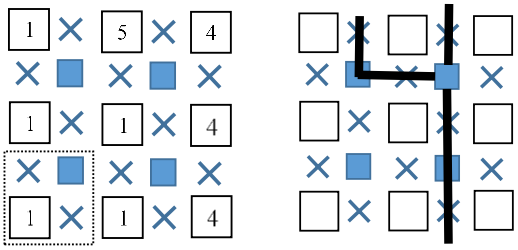


Fig. 11. Left: a fragment (9 pixels) of a color image. Numbers represent color values of the original image pixels. For the ACC representation, numbers are 0-cells, crosses are 1-cells and solid squares are 2-cells. At the most bottom left corner, the dotted square is an elemental PE composed of 4 cells. Right: a possible contour tree (divided into subtrees) containing the border information of the image represented by 1 and 2 cells.

For two dimensional objects, only two homology groups must be considered: those representing connected components and holes. However, this topological framework can be extended to high dimensional images by defining the proper elemental PE that allows a complete topological representation and, afterwards, by building the different *k-(k+1)* trees (being *k* a dimension) in the most effective way [30]. Then, objects immersed on the *nD*-image would be represented by homology groups of dimensions *0, 1, …, n-1*.

CONCLUSIONS

Most of the image processing algorithms part from tracing-type or scan/raster methods. This fact necessarily introduces data dependences between the processing of one pixel and the previous one, which prevents pure parallel implementations. We describe a very different approach based on a pure topological framework, which allows to implement fully parallel algorithms. This yields to an image representation that avoids complex data structures. In fact, all the processing can be done using matrixes (with the same indexation as the original image) and element-wise operations. Theoretical time complexity orders of our topological approach for an image of $m \times n$ pixels is near $O(\log(m+n))$. Being a consistent topological framework, this method can be extended to color *n*-dimensional images.

ACKNOWLEDGEMENT

This work has been supported by the Spanish research projects MTM2016-81030-P (AEI/FEDER,UE)) and TEC2012-37868-C04-02 of Ministerio de Economía y Competitividad and the VPPI of the University of Seville.

REFERENCES

[1] P. Bhattacharya. Connected component labeling for binary images on a reconfigurable mesh architecture. Journal of Systems Architecture, 42(4):309-313, 1996.
[2] O.P. Buneman. A Grammar for the Topological Analysis of Plane Figures. Edinburgh Univ. Press, pp. 383-393, 1969.
[3] F. Chiavetta, V. Di Ges. Parallel computation of the Euler number via connectivity graph. Pattern Recognition Letters 14, 849-859, 1993.
[4] F. Diaz-del-Rio, P. Real, D., Onchis: A parallel Homological Spanning Forest framework for 2D topological image analysis. Pattern Recognition Letters 83, 49-58, 2016.
[5] A. Cohn, B. Bennett, J. Gooday, N. Gotts: Qualitative spacial representation and reasoning with the region connection calculus. GeoInformatica 1(3), 275-316, 1997.
[6] E. Costanza, J. Robinson. A region adjacency tree approach to the detection and design of fiducials. Video, Vision and Graphics, pp. 63-99, 2003.
[7] R. Cucchiara, C. Grana, A. Prati, S. Seidenari, G. Pellacani. Building the topological tree by recursive FCM color clustering. 16th IEEE ICPR, 1, pp. 759-762, 2002.
[8] S. Gupta, D. Palsetia, M.M.A.Patwary, A. Agrawal, A.N. Choudhary. A new parallel algorithm for two-pass connected component labeling, in: IEEE IPDP Symposium, pp. 1355-1362, 2014.
[9] H. J. Heijmans. Connected morphological operators for binary images, Comput. Vis. Imag. Understand., 73 (1), pp. 99-120, 1999.
[10] O. Kalentev, A. Rai, S. Kemnitz, R. Schneider. Connected component labeling on a 2d grid using CUDA. J. Parallel Distrib. Comput. 71, 615-620, 2011.
[11] V. Kovalevsky.: Algorithms in Digital Geometry Based on Cellular Topology. 10th IWCIA, Springer Berlin Heidelberg, vol. 3322, pp. 366-393, 2004.
[12] R. Keshet.: Shape-tree semilattice. J. Math. Imag. Vis., 22 (2-3), pp. 309-331, 2005.
[13] A., Murty, V., Natarajan, S., Vadhiyar.: Efficient homology computations on multicore and manycore systems, in: 20th Annual International Conference on High Performance Computing, pp. 333-342, 2013.
[14] NVIDIA, Cuda C best practices guide version. http://developer. nvidia.com/. Oxley, J.G., Matroid theory. volume 3. Oxford University Press, 2017.
[15] M. Patwary, M. Ali, P. Refsnes, and F. Manne. Multi-core spanning forest algorithms using the disjoint-set data structure. In 26th IEEE IPDP Symposium, pp. 827-835, 2012.
[16] REDHOM, Redhom. http://redhom.ii.uj.edu.pl/, Institute of Computer Science, Jagiellonian University, 2017.
[17] V. Ranwez, P. Soille, Order independent homotopic thinning for binary and grey tone anchored skeletons, Pattern Recognition Letters 23 (6), 687-702, 2002.
[18] A. Rosenfeld. Adjacency in digital pictures. Inf. Control 26, 24-33, 1974.
[19] J. Serra. Image Analysis and Mathematical Morphology. Academic Press, 1982.

[20] J. Stell and M. Worboys.: Relations between adjacency trees. Theo. Comp. Sci., 412 (34), pp. 4452-4468, 2011.
[21] S. Williams, A. Waterman, D.A. Patterson. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52, pp. 65-76, 2009.
[22] YACCLAB - Yet Another Connected Components Labeling Benchmark. https://github.com/prittt/YACCLAB, 2017.
[23] Grana, C., Bolelli, F., Baraldi, L., Vezzani, R., 2016. YACCLAB - Yet Another Connected Components Labeling Benchmark, in: 23rd International Conference on Pattern Recognition, ICPR.
[24] Grana, C., Borghesani, D., Cucchiara, R., 2010. Optimized block-based connected components labeling with decision trees. IEEE Transactions on Image Processing 19, 1596–1609
[25] Swendsen, R.H., Wang, J., 1987. Non-universal critical dynamics in Monte Carlo simulations. Phys. Rev. Lett. 58, 86–88.
[26] Díaz-del-Río, F., H., Molina-Abril, P. Real, 2019. Computing the component-labeling and the adjacency tree of a binary digital image in near logarithmic-time. Computational Topology in Image Context, Lecture Notes in Computer Science, Springer 11382, 82–95.
[27] Soille, P. , Constrained Connectivity for Hierarchical Image Partitioning and Simplification, IEEE Transactions on Pattern Analysis and Machine Intelligence, V 30, N. 7, pp 1132–1145, 2008.
[28] Petra Bosilj, Ewa Kijak, Sébastien Lefèvre. Partition and Inclusion Hierarchies of Images: A Comprehensive Survey. Journal of Imaging, MDPI, 2018, 4 (2), pp.1-31.
[29] R. Klette , Cell complexes through time, in: Proceedings of SPIE 4117, vol. 4117, 20, pp. 134–145 .
[30] Real, P., Diaz-del-Rio, F., Onchis, D.: Toward Parallel Computation of Dense Homotopy Skeletons for nD Digital Objects. In International Workshop on Combinatorial Image Analysis, pp. 142-155. Springer, 2017.
[31] Real, P., Diaz-del-Rio F., Onchis, M. Labeling color 2D digital images in theoretical near logarithmic time. 17th international Conference on Computer Analysis of Images and Patterns. Istad, Suecia. 2017
[32] Y. Xu, P. Monasse, T. Géraud, L.Najman. Tree-Based Morse Regions: A Topological Approach to Local Feature Detection October 2014IEEE Transactions on Image Processing 23(12).
[33] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition 70 (2017) 25–43