Labeling Color 2D Digital Images in Theoretical Near Logarithmic Time

F. Díaz-del-Río[1], P. Real[1], and D. Onchis[2,3]

[1] H.T.S. Informatics' Engineering, University of Seville, Seville, Spain
fdiaz@atc.us.es, real@us.es

[2] Faculty of Mathematics, University of Vienna, Vienna, Austria
darian.onchis@univie.ac.at

[3] Faculty of Mathematics and Computer Science, West University of Timisoara, Timisoara, Romania

Abstract. A design of a parallel algorithm for labeling color flat zones (precisely, 4-connected components) of a gray-level or color 2D digital image is given. The technique is based in the construction of a particular Homological Spanning Forest (HSF) structure for encoding topological information of any image. HSF is a pair of rooted trees connecting the image elements at inter-pixel level without redundancy. In order to achieve a correct color zone labeling, our proposal here is to correctly building a sub-HSF structure for each image connected component, modifying an initial HSF of the whole image. For validating the correctness of our algorithm, an implementation in OCTAVE/MATLAB is written and its results are checked. Several kinds of images are tested to compute the number of iterations in which the theoretical computing time differs from the logarithm of the width plus the height of an image. Finally, real images are to be computed faster than random images using our approach.

Keywords: Digital image · Gray-level · Color · Adjacency · Flat zone · Contour · Connected component · Hole · Parallel algorithm · Homological spanning forest

1 Introduction

In general, n-xel's values of biomedical digital images have a relevant physical meaning. In this context, to find a semantically correct segmentation of a gray-level or color nD digital image based on merging original regions (or flat zones) of constant color represents an important processing problem in image understanding. For undertaking this task, we can start with an initial decomposition of the domain of the image into a set $\{R_i\}$ given by the color flat zones of I (connected components where color is constant). This is called here *color-constant region pre-segmentation* and the process in order to construct it, *connected component labeling* (or *CCL*, for short) of the original image. Within the digital context, this pre-segmentation strongly depends on the adjacency relationship we choose for connecting n-xels. We can interpret this problem as well as the generation of subsequent high level segmentations in terms of topological reductions (in number of cells) of abstract cell complexes (or ACC, for short) [18] representing the segmentations. Depending of the dimensionality (0 or n) of the cells of the different labeled connected components (CCs) of the image, we get a Region-Incidency-ACC or a Contour-Incidency-ACC. Classically, this issue has been treated in the literature with ACCs of dimension one (that is, graphs), like the Region-Adjacency-Graph and its dual (see, for instance, [4,8,17] for a 2D treatment). Problems yet unsolved of different nature and difficulty appears in the process of building a model for digital images capable to be efficiently used in topological recognition tasks. Recently, an important advance in this sense has been the development of the HSF (Homological Spanning Forest) framework for topological parallel computing of 2D digital images [7,20]. Roughly speaking, an HSF of a digital image I is a set of trees living at interpixel level within an abstract cell complex version of I and appropriately connecting all cells without redundancy. This notion is in principle independent of the pixel's values of the image. If there is an interest to analyze a concrete digital object D within it, it is possible to construct an admissible sub-HSF structure for D, "adapting" an original HSF of the whole image to the interpixel frontier or membrane between the object and the background. In this way, such HSF structure can be a useful tool for topologically classifying digital closed curves inside the object.

In this paper, working with the connectivity criterion of 4-adjacency for all the pixels, we develop a parallel CCL algorithm for a gray-level or color 2D digital image based on the previous computation of a particular HSF structure of this image called *region-contour HSF*. This HSF is "adapted" to the image's contour (at interpixel level, the set of digital curves, formed by crack vertices and edges, between neighbor color flat zones). Again, this adaptation process is done in parallel via a combinatorial optimization technique called *crack transport* applied to an initial HSF of the image (called Morse Spaning Forest). Figure 1 gives an idea of how this technique works.

Distinguishing some cells in the region-contour HSF, it is possible to generate a labeling procedure of 4-CCs, including their corresponding area and perimeter measurements.

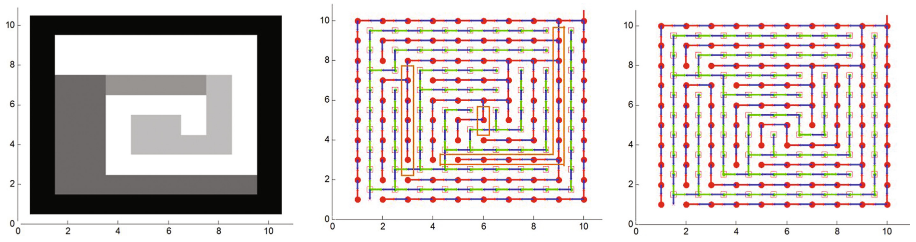

Fig. 1. (Left) A gray level 2D digital image. (Middle) HSF structure of the gray-level image consisting in two trees that is not adapted to the image's contour; (Right) Resulting Region-contour HSF after three crack transports

A sledgehammer to crack nuts? The advantages of this new CCL approach based on HSF structure are:

– It can be considered as one of the more parallel procedures for labeling CCs up to know. Its theoretical time complexity is near the logarithm of the width plus height of the image and the experimental results obtained certify that the method correctly labels even large images. In this sense, we think that an implementation over a many-core processor could significantly improve the speed of the algorithm.
– It is susceptible to be extended to any dimension, due to the fact that the topological HSF structures can be defined in nD space.
– Its versatility regarding the type (binary, gray-level, color, ...) of the analyzed image. Most of the CCL algorithms existing in the literature exclusively works for binary images.
– It is susceptible to be promoted to obtain a useful and efficient HSF-based model representation for recognition tasks or other high level computer vision applications. A possible idea for properly extending the above CCL algorithm is to distinguish in a region-contour HSF extra critical cells of dimension 1 and 2 that allow to find relations of the kind "to be surrounded by" between sets of neighbor CCs.

There is a plethora of CCL algorithms available in the literature. Regarding the technique, there are mainly two classes of algorithms: *raster-scan algorithms*, such as one-pass ([1,15]) and two-pass ([11,12,24,26,30]), and *label propagation algorithms* ([3,5,14,27]). Regarding the processing, we have sequential ([25,28]) and parallel ([2,10,13,16,21]) algorithms. Regarding the type of image, there are also two classes: binary CCL algorithms (most of the previous references enter into this category) and gray-level and color ([6,19,22,23,29]) CCL algorithms. A historical overview of this fundamental low-level image processing operation is given in [9]. Most of the previous references are valid in 2D digital context and the criterion of pixel connectivity relies on 4-adjacency or 8-adjacency.

2 Generation of HSF Trees

CCL is one of the fundamental operations in real time applications. The labeling operation transforms an image into a symbolic matrix in which all elements (pixels) belonging to a CC are assigned to a unique label. One of the possible applications of a region-contour HSF is obtaining a CCL. The information of CCs and their contours is registered in some special cells, which are called critical cells. For example, in Fig. 2, we show a region-contour HSF of a gray-level 2D digital image and the same HSF in which we have distinguished on it some critical cells of dimension 0 (square), 1 (circle) and 2 (hollow square), attending to some criteria. Precisely, the 0-critical cells we have choose are representative pixels of the different 4-CCs. The 1-critical cells named by A1 and B1 specify 1-holes of the regions A and B, respectively. For instance, the hole B1 surrounds the 8-connected set formed by the 4-connected regions C, D, E and F. The 1-critical

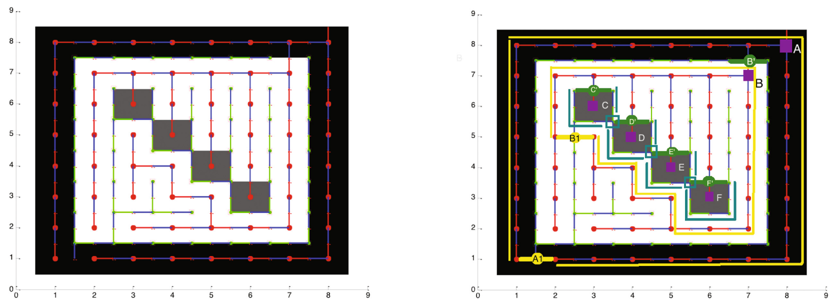

Fig. 2. (left) Region-contour HSF structure of a gray-level 2D digital image; (right) Distinguishing some critical cells on the HSF.

cells named by C', D', E', and F' determine 1-holes of the contour of the image. The three 2-critical cells (hollow squares) are the crossing points of the image's contour. Let us limit ourselves to say that good choice of extra critical cells of dimension 1 and 2 on a region-contour HSF would allow to discover efficient topological recognition solvers.

From now on, we focus on strategies allowing the implementation of an efficient and parallel computation of a region-contour HSF. In this respect, we keep the mathematical background to a minimum and we avoid the use of a too much technical and not informative pseudo-code style. The main three stages for constructing an HSF structure of a digital image are in order:

1. **Input data**. Input data are 2-dimensional positive integer-valued matrices of size $m \times n$ associated to a color or gray-level 2D digital image I. The outermost border of I is filled with an inexistent color (e.g. -1).
2. **Generation of initial HSF trees**. The topological interpretation of the image is condensed at inter-pixel level into two trees of an HSF of I. One of them is composed by 0 and 1-cells (called 0–1 tree) whereas the rest of 1-cells and 2-cells lives on the 1–2 tree. In principle, an HSF is a notion that is independent of the pixel intensities in the image. In our algorithm, the initial constructed HSF is called Morse Spanning Forest (MrSF) and we limit ourselves to say that it can be built in parallel [7], with an architecture of one processing unit element (PE) for each image's pixel.
   This process is explained here using a 8×8-pixel image in Fig. 3. The 0-cells are drawn with small solid red circles, 1-cells with crosses and 2-cells with squares. Edge-vectors of the HSF trees (called links in [7]) having a 0, 1 or 2-cell as its heads are respectively shown with red, blue and green lines. The final result is that the 0–1 tree has a root at the most Northeast corner of the image, while the 1–2 tree is rooted at the most southwest 1-cell. Some special cells (called critical cells) are registered in this stage: a) The 0-cells c (which play a similar role to the sinks in [7]) which are the heads of a link (c, c') whose tail is a 1-cell c' surrounded by two 0-cells of different colors. In Fig. 3 (Left and Middle), sinks are marked with dotted circles; b) Those 1-cells c

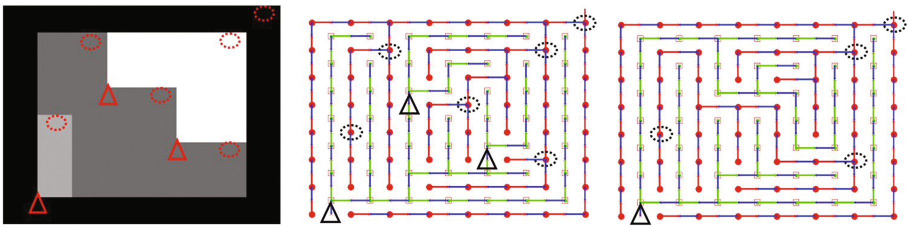

Fig. 3. (Left) A 2D digital image containing a stair-like shape. (Middle) the MrSF of the image. 0-cells are drawn with small solid red circles, 1-cells with little crosses and 2-cells with squares Links coming from 0, 1 and 2-cell are shown with red, blue and green lines, resp. (Right) a resulting HSF after optimization based on crack transport. (Color figure online)

(which play a similar role to the sources in [7]) that, being surrounded by two 0-cells of same color, are the head of a link (c, c') whose tail is a 2-cell c' of a contour of the CC. In Fig. 3 (Left and Middle), sources are marked with triangles. In fact, associated to each sink and source c, there is a pair of neighbor links: (c, c') and (c', c''), such that the dimension of c'' is the same than that of c. This pair of links is called a *crack of the MrSF*.

3. **Crack transport**. It is a combinatorial optimization process (in terms of "transports" of cracks) in order to get two new trees (an HSF as in Fig. 3, right) from the original MrSF. In the resulting region-contour HSF, there is only one sink for each CC and one source for each hole. Finally in Fig. 3 (right), there are only four sinks (one for each region) and one source (the dummy contour of the image surrounds the rest of regions). This means that the number of critical cells must be reduced to its minimum. The correct and parallel implementation of these transports is detailed in the next section.

3 A Parallel Algorithm for Building the HSF of Color Images

In this paper, we propose a variation of our previous works [7,20] so that two important achievements are attained: (1) The parallel algorithm presented here is extended to color images and; (2) No crack transport is done in a sequential manner. The first goal is accomplished thanks to the proper definition of sinks and sources for color images. This yields to the construction of an MrSF which is valid for the following parts of the processing. The nature and degree of the parallel computing in the MrSF construction remains the same than in [7]. In consequence, working with a processing element per pixel, the time complexity is preserved to be the logarithm of the width plus height of the image. We refer the reader to these works for specific details of the algorithm, or directly to our implementation (http://es.mathworks.com/matlabcentral/fileexchange/62644--labeling-color-2d-digital-images-in-theoretical-near-logarithmic-time-).

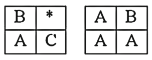

Fig. 4. Pixel patterns for detecting sinks (left) and sources (right). (*) means any color. Letters A, B, C are specific colors, being A different from B and C.

The second aspect comes from certain properties of the MrSF trees. In this case, the number of parallel crack's transport to get to the final HSF is considerably reduced in relation to the previous work. In fact, the biggest number of iterations for images up to 4 Mpixels has been found to be 5 for only some random images (and usually inferior for real images). To sum up, the time order complexity results to be very near to that of the MrSF building (that is, logarithmic). In the rest of this section several issues to construct the resulting HSF are fully detailed.

MrSF is built in a similar manner to [7], that is, each 0-cell link must travel only to its North or to its East. The next priority rules are followed in order to compute the link direction of each 0-cell. First, connection between contour pixels of the same color is preferred. If both (North and East) neighbors are in the CC contour, the North direction is (arbitrarily) chosen. Second (if previous rule is not satisfied), North direction is chosen if both neighbors have the same color. Finally, if both neighbors have a different color, the 0-cell is marked as a sink. Note that the case in which only one (North and East) neighbor has the same color that of the current 0-cell is included in the first rule.

Besides, when the two North and East neighbors of a pixel have the same color but the North-East one is of different color, one of the 1-cells that are between these three identical color pixels is marked as critical. To sum up, in order to build an MrSF for color images using a 4-adjacency criterion, the definitions of sink and source are that given by Fig. 4. Previous link direction rules and the search of the critical cells can be done fully in parallel, which supposes a time order complexity of O(1).

Once the rules for the 0-cell links are defined, the rules for the rest of links is done in the same manner than in [7]. The next step is the building of a provisional labeling over the MrSF trees. As our aim is finding topological magnitudes of an image, it is necessary the extraction of global information. This force unavoidably to insert a global searching across the whole image. In this step, parallelization remains the same than in [7], which yields to a time order complexity equal to $O(log(m+n))$. This is the most time consuming part of the parallel labeling.

After the previous MrSF provisional labeling, we proceed to transform the MrSF into an HSF, which must contain the minimum possible number of critical 0-cells and 1-cells. This conversion can be understood from several points of view:

1. If a CC held several 0–1 trees (each one had a sink as a root), those trees must be fused into one. This fusion supposes a crack transport that also changed some links in the 1–2 tree. An example of a region with several 0–1 trees is the biggest CC in Figure 3 (left). This region contains three sinks (marked

with dotted circles), thus it holds three 0–1 trees. After the fusion process, the region will contain one tree (Fig. 3, right)

2. If a CC contained several sinks and several sources, they must be paired so that finally only one sink (and one source per hole) would remain. This cancellation process involves crack transports in both the 0–1 and the 1–2 trees. Using the same example, the biggest region in Fig. 3 (left) has three sinks and two sources, which implies that two sink/source pairs must be canceled.
3. Previous viewpoints are useful to understand our goal but they do not give any insight about how to proceed with the crack transports or cancellations in a parallel manner. In order to achieve parallelism, we must find a procedure to detect the maximum number of sink-source pairs that can be simultaneously canceled. In this regard, the two MrSF tree structures provide uniqueness conditions that help us finding a suitable parallel cancellation method.

In fact, the method proposed here develop the third point of view. In Fig. 5 (Left) some possible indexations of three sinks (named 1, 2, 3) along the 1–2 tree are shown. Indexation of sources A, B and C (along the 0–1 tree) are also drawn. The indexation along a tree is unique. However, two nodes can arrive to the same node as depicted in this figure. When a sink points to a source (through the 1–2 tree), and this source points to the same sink (through the 0–1 tree), this implies that this sink/source pair can be canceled. Moreover, because indexations are unique, all the pair cancellations can be done in parallel pairs. For example, in Fig. 5 (Left) the continuous arrows determine that the pairs 1/A and 3/C can be canceled in parallel.

Furthermore, in the case of 2D images, two possible indexations of each sink and each source must be taken into consideration. Each sink splits the 1–2 tree into two parts: one of them would travel to the South and the other to the West. Likewise, each source divides the 0–1 tree into two parts: one of them would travel to the East, and the other to the North. This yields to two different indexations for each critical cell and, thus, the possibility of considering two possible cancellations for a same sink (with two different sources). This case is

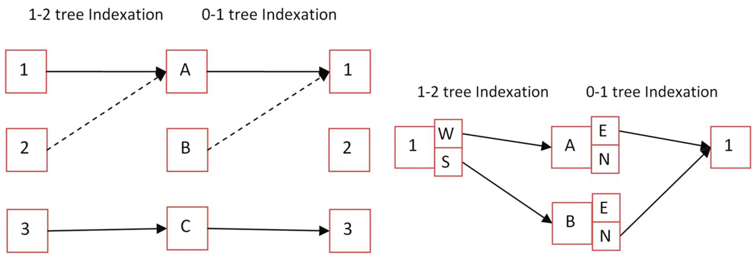

Fig. 5. (Left) An example of tree indexations from two sinks and two sources. (Right) an example of two possible pair cancellations for the same sink through two different sources.

depicted in Fig. 5 (right), and can frequently occur for stair-like shapes as it happens in the center sink of the biggest region of Fig. 3.

In conclusion, in order to proceed with parallel cancellations, in a first iteration sink/source pairs must be searched, for instance, using the South indexation of every sink and the North indexation of every source (which can be called South-then-North cancellation). After that, in a second iteration, different pairs can be found using the West indexation of every sink and the East indexation of every source (which can be called West-then-East cancellation). Note that the selected order in our method is arbitrary: a similar procedure can be done in the opposite order, or, even, using first the indexations from sources and then from sinks. Evidently the selected order would produce different results in terms of numbers of iterations. These numbers depend on the shapes of the regions that are unknown when trying to label them.

To understand the computational difficulties involved in the crack transport, a more complex example to transform the MrSF into HSF through two iterations is given by the following Fig. 6. Figure 6 (left) shows an image that contains a spiral, an L-shape and a reflected L-shape. The superfluous sink and source (4 and A) of the L-shape region can be promptly canceled using a first iteration (South-then-North cancellation). In consequence, this implies that cracks of the trees are transported and those sinks that pointed to A now points to B, and likewise, those sources that pointed to 4 must be now redirected to 5. The resulting trees after the first South-then-North cancellation is displayed at Fig. 6 (middle). Nevertheless, it must be noted that the superfluous sink and source of the spiral cannot be canceled in the first iteration. Thus, the spiral must wait until a second iteration (West-then-East cancellation), resulting in the final Fig. 6 (right).

Previous redirections in the 0–1 and 1–2 trees suppose a searching of the corresponding root that must jump across several hops. If the number of hops is high, it may imply an increment of the total time processing. However, the number of cells involved in these redirections is not very high, which means that the total number of operations is very much lower than that of initial MrSF building. One question remains: When should these iterations be stopped? This

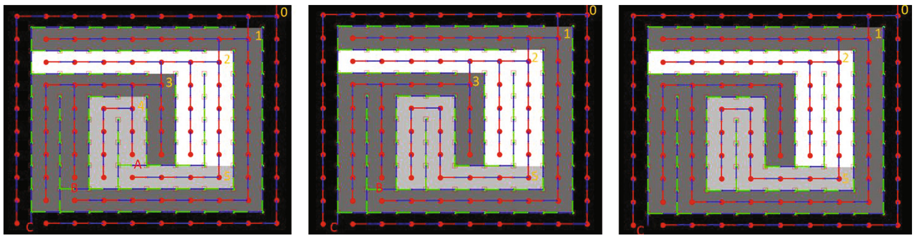

Fig. 6. from left to right: The MrSF of an image containing a spiral, an L-shape and a reflected L-shape; Its MrSF after a first iteration (South-then-North cancellation); Its HSF, obtained after a second iteration (West-then-East cancellation). Letters indicate the sources and numbers the sinks.

question can be answered if the number of false sources are counted. A true source, which denotes the presence of a hole in a region, is that one whose East and North indexations along the 0–1 tree points to the same sink (e.g. the source C in Fig. 6, right). Conversely, a false source is that one whose East and North indexations along the 0–1 tree points to two different sinks. After each iteration, the number of false sources can be computed: if this number is exactly zero, iterations must be stopped. In conclusion, the number of iterations for the parallel crack transport process cannot be a priori known and depends on the shapes of the regions of an image. As explained in the next section, our tests show that this number is very low.

4 Experimental Results

In order to corroborate the correctness of our algorithm, an implementation in OCTAVE/MATLAB is written. Results are checked against those values returned by functions like bweuler() and bwlabel(). All the figures along this work are generated with these codes. Several kinds of images are tested to compute the number of iterations and the number of remaining sinks after each iteration. In general, real images are computed faster than random images. The number of iterations is lower for the first ones due to the fact that they usually present fewer spiral-like shapes. In Table 1 results for random images of different gray levels and sizes are presented. These images are generated by multiplying the number of gray levels by the function rand(). From left to right, the rate in which the number of sinks and false sources is reduced is shown (until no false source remains). In general the less the number of levels, the more difficult to process the image is (it needs more parallel iterations). Nevertheless, we have not found any image requiring more than 5 iterations (for sizes until 2048x2048 pixels). Table 2 shows the corresponding result from some very viewed medical images (taken from http://goldminer.arrs.org/top-40.php).

Another interesting parameter is the number of hops along the redirections when searching for the root of each sink and source in the cancellation process. This gives an idea of the total time spent on this stage. The mean results after four tests using big random images are shown in Table 3. This implies that for the image of 2-gray levels and 4 MiPixels only an amount of (1263 + 311 + 35)=1609 of access operations will be necessary; this number is reduced to 58 when processing the 8-gray level image of the same size. Evidently this quantities are very much lower than the total number of operations for obtaining the initial MrSF building, which means that this stage does not have a considerable impact on the processing times. It can be seen that for random images the number of hops decreases considerably along with the number of gray levels. Due to this, real images present usually a much smaller number of hops: the maximum number is 386 in the third iteration for the larger medical tested image.

Table 1. Results for random images of different gray levels.

# gray levels	height	width	#iter.	# initial sinks	# sinks after iter. 1	# sinks after iter. 2	# sinks after iter. 3	# sinks after iter. 4	# sinks after iter. 5	# initial false sources	# false sources after iter. 1	# false sources after iter. 2	# false sources after iter. 3	# false sources after iter. 4	# false sources after iter. 5
2	128	128	3	3976	2532	2169	2153			1905	406	16	0		
2	256	256	4	16175	10222	8594	8520	8518		7889	1776	87	2	0	
2	512	512	4	65434	41871	35162	34876	34868		31615	7347	329	8	0	
2	1024	1024	5	261152	165643	138842	137611	137564	137563	127924	29566	1428	56	2	0
2	2048	2048	5	1049780	667224	559273	554159	553986	553985	513098	119079	5857	193	2	0
8	128	128	2	12225	12035	12034				191	1	0			
8	256	256	2	49548	48703	48692				856	11	0			
8	512	512	2	199045	195433	195384				3661	49	0			
8	1024	1024	2	799253	785023	784823				14430	200	0			
8	2048	2048	2	3207902	3151400	3150598				57304	802	0			
32	128	128	2	14897	14883	14883				14	0	0			
32	256	256	2	60503	60452	60452				51	0	0			
32	512	512	2	244186	243963	243963				223	0	0			
32	1024	1024	2	980744	979775	979775				969	0	0			
32	2048	2048	2	3928115	3924217	3924214				3901	3	0			

Table 2. Results for several medical images of different sizes.

Image Name	# gray levels	height	width	# iter.	# initial Sinks	# sinks after iter. 1	# sinks after iter. 2	# sinks after iter. 3	# sinks after iter. 4	# initial false sources	# false sources after iter. 1	# false sources after iter. 2	# false sources after iter. 3	# false sources after iter. 4
EURORAD2015	256	600	498	4	202861	196423	195659	195645	195644	7323	808	19	1	0
AJR2004	256	295	166	3	13417	11286	10457	10432		4006	1378	39	0	
AJNR2013	256	1196	1800	4	581353	471635	460343	460275	460274	122025	11479	71	2	0
AJR2009	256	295	260	2	57208	55608	55492			1729	117	0		
Radiology2007	256	526	1274	2	264588	262253	262127			2503	131	0		
PartFibreToxicol2014	256	1000	1200	4	391612	353944	348131	347876	347850	45881	6664	422	48	0

Table 3. number of hops along the redirections when searching for link transports for random images of different sizes and gray levels.

# gray levels	**height**	**width**	**# iter.**	**# hops during iter. 1**	**# hops during iter. 2**	**# hops during iter. 3**	**# hops during iter. 4**
2	512	512	4	0	32	166	375
2	1024	1024	4	0	34	414	674
2	2048	2048	4	0	35	311	1263
8	512	512	2	0	18		
8	1024	1024	2	0	27		
8	2048	2048	2	0	58		

5 Conclusions

In this paper, we design a parallel 4-adjacency CCL algorithm based on region-contour HSF of a 2D digital image. In a near future, we intend to progress in several directions: (a) to develop a fully functional implementation of the HSF framework in a language (like C++ or python) which allows us to efficiently exploit the parallelism over multicore architectures; (b) to extend the CCL algorithm based on HSF structure to 3D and 4D.

At long term, we are interested in developing a topologically consistent and robust nD digital image analysis and recognition, trying to establish meaningful and efficient topological representation models of images and objects based on HSF structures. For future high level computer vision applications, CCs would be the fundamental bricks whose adjacency relationships would be established through an image HSF structure.

Acknowledgments. This work has been supported by the Spanish research projects (AEI/FEDER,UE) TEC2016-77785-P and MTM2016-81030-P. The last co-author gratefully acknowledges the support of the Austrian Science Fund FWF-P27516.

References

1. Abubaker, A., Qahwaji, R., Ipson, S., Saleh, M.: One scan connected component labeling technique. In: IEEE International Conference on Signal Processing and Communications, pp. 1283–1286. IEEE (2007)
2. Alnuweiri, H.M., Prasanna, V.K.: Parallel architectures and algorithms for image component labeling. IEEE T. Pattern Anal. **10**, 1014–1034 (1992)
3. Ballard, D.H., Brown, C.M.: Computer Vision. Prentice-Hall, Upper Saddle River (1982)
4. Braquelaire, J.P., Brun, L.: Image segmentation with topological maps and inter-pixel representation. J. Vis. Commun. Image R. **9**(1), 62–79 (1998)
5. Chang, F., Chen, C.J., Lu, C.J.: A linear-time component-labeling algorithm using contour tracing technique. Comput. Vis. Image Und. **93**(2), 206–220 (2004)
6. Crespo, J., Schafer, R.W.: The flat zone approach and color images. In: Serra, J., Soille, P. (eds.) Mathematical Morphology and Its Applications to Image Processing Computational Imaging and Vision, vol. 2, pp. 85–92. Springer, Dordrecht (1994)
7. Diaz-del-Rio, F., Real, P., Onchis, D.M.: A parallel homological spanning forest framework for 2D topological image analysis. Pattern Recogn. Lett. **83**, 49–58 (2016)
8. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Internat. J. Comput. Vis. **59**(2), 167–181 (2004)
9. Grana, C., Borghesani, D., Cucchiara, R.: Optimized Block-based connected-component labeling with decision trees. IEEE T. Image Process. **19**(6), 1596–1609 (2010)
10. Han, Y., Wagner, R.A.: An efficient and fast parallel-connected component algorithm. J. ACM **37**(3), 626–642 (1990)
11. He, L., Chao, Y., Suzuki, K.: A run-based two-scan labeling algorithm. IEEE T. Image Process. **17**(5), 749–756 (2008)
12. He, L., Chao, Y., Yang, Y., Li, S., Zhao, X., Suzuki, K.: A novel two-scan connected-component labeling algorithm. In: Yang, G.-C., Ao, S.-L., Gelman, L. (eds.) IAENG Transactions on Engineering Technologies. Lecture Notes in Electrical Engineering, vol. 229, pp. 445–459. Springer, Dordrecht (2013)
13. Hesselink, H., Meijster, A., Bron, C.: Concurrent determination of connected components. Sci. Comp. Programm. **41**, 173–194 (2001)
14. Hu, Q., Qian, G., Nowinski, W.L.: Fast connected-component labeling in three-dimensional binary images based on iterative recursion. Comput. Vis. Image Und. **99**(3), 414–434 (2005)

15. Johnston, C.T., Bailey, D.G.: FPGA implementation of a single pass connected components algorithm. In: 4th IEEE International Symposium on Electronic Design, Test and Applications, pp. 228–231. IEEE(2008)
16. Kalentev, O., Rai, A., Kemnitz, S., Schneider, R.: Connected component labeling on a 2D grid using CUDA. J. Parallel Distrib. Comput. **71**(4), 615–620 (2011)
17. Kropatsch, W.G.: Building irregular pyramids by dual-graph contraction. IEEE Proc. Vis. Image Signal Process. **142**(6), 366–374 (1995)
18. Kovalevsky, V.: Geometry of Locally Finite Spaces. Publishing House Dr. Baerbel Kovalevski, Berlin (2008)
19. Meyer, F.: From connected operators to leveling. In: Mathematical Morphology and its Applications to Image and Signal Processing. Computational Imaging and Vision, vol. 12, pp. 191–198. Kluwer Academic Publishers (1998)
20. Molina-Abril, H., Real, P.: Homological spanning forest framework for 2D image analysis. Annals Math. Artificial Intell. **4**(64), 385–409 (2012)
21. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tessellations. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4), 307–316 (1991)
22. Mandler, E., Oberlander, M.F.: One-pass encoding of connected components in multi-valued images. In: Proceedings of the IEEE International Conference on Pattern Recognition, vol. 2, pp. 65–69 (1990)
23. Niknam, M., Thulasiraman, P., Camorlinga, S.A.: A parallel algorithm for connected component labeling of gray-scale images on homogeneous multicore architectures. J. Phys: Conf. Ser. **256**(012010), 1–7 (2010)
24. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. J. ACM **13**(4), 471–494 (1966)
25. Samet, H.: Connected-component labeling using quadtrees. J. ACM **28**(3), 487–501 (1981)
26. Schwenk, K., Huber, F.: Connected-component labeling algorithm for very complex and high-resolution images on an FPGA platform. In: SPIE Remote Sensing, ISOP (2015)
27. Shima, Y., Murakami, T., Koga, M., Yashiro, H., Fujisawa, H.: A high-speed algorithm for propagation-type labeling based on block sorting of runs in binary images. In: Proceedings the 10th International Conference Pattern Recognition, pp. 655–658, June 1990
28. Suzuki, K., Horiba, I., Sugie, N.: Linear-time connected-component labeling based on sequential local operations. Comput. Vis. Image Und. **89**(1), 1–23 (2003)
29. Sang, H., Zhang, J., Zhang, T.: Efficient multi-value connected component labeling algorithm and its ASIC design. In: Proceedings of the SPIE Medical Imaging Conference (2007). 67892I
30. Wu, K., Otoo, E., Suzuki, K.: Optimizing two-pass connected-component labeling algorithms. Pattern Anal. Appl. **12**(2), 117–135 (2009)