Pyramidal Connected Component Labeling by Irregular Graph Pyramid

1st Majid Banaeyan
Vienna University of Technology
Pattern Recognition and Image Processing Group 193/03
Vienna, Austria
majid@prip.tuwien.ac.at

2nd Walter G. Kropatsch
Vienna University of Technology
Pattern Recognition and Image Processing Group 193/03
Vienna, Austria
krw@prip.tuwien.ac.at

2021 5th International Conference on Pattern Recognition and Image Analysis (IPRIA) | 978-1-6654-2659-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/IPRIA53572.2021.9483533

***Abstract*—This paper presents a new logarithmic-time algorithm which simultaneously assigns labels to all connected components of a binary image in parallel. The irregular graph pyramid of an input binary image is constructed based on the optimized combinatorial structure. The novel small built pyramid has only three levels and is created at worst case with $O(log(N^2))$ complexity for a $N \times N$-sized (2D) binary image. To assign a label to each connected component, instead of the common linear-time raster scan techniques (with complexity $O(N^2)$), only two important movements, namely bottom-up and top-down traversing, are needed. First, in the bottom-up traversing, the redundant connections are removed and the contraction kernels are contracted. This results in a simpler graph at top of the pyramid each of its vertices has a unique label identifying corresponding connected component. Such reduced graph preserves all connecting relations including inclusions. Second, in the top-down traversing, each unique label propagates down into each individual corresponding pixel at the base level. The complexity of the labeling propagation procedure in worst cases is $O(log(image-size))$. The GPU implementation of the algorithm has high performance and the bottleneck is the bandwidth of the memory or equivalently the number of available independent processing elements. Finally, the experimental results show the proposed algorithm outperforms the other state-of-the-art methods.**

***Index Terms*—Image Processing, Combinatorial Pyramid, Parallel Processing, Invariant Topology, Connected Component Labeling**

I. Introduction

Connected Component Labeling (CCL) is a fundamental task in computer vision. Given a binary image it distinguishes between background or foreground components and assigns a unique label to each different region. It employs in many image processing fields such as image analysis, pattern recognition and image understanding. The role of the CCL is to assign a same label into each individual pixel of a region. There are plenty of algorithms which consider this task from different viewpoints [1]–[3]. However, such algorithms mostly can be divided into two main categories [4] which are based on label-propagation [5]–[7] or label-equivalence-resolving [8], [9]. The common property in both techniques is that they are all linear algorithms on the size of binary images. In other words, such algorithms may differ from one-scan or two-scan searching through the entire image, but all are in the order of image size, $O(N^2)$. In this paper, we propose an algorithm which reduces this complexity. It employs the advantage of the irregular hierarchical pyramidal structure [10]–[14] and reduced the complexity into a logarithmic order.

In addition, unlike the previous irregular pyramids [15], [16], here, the constructed pyramid has dramatically much smaller size and therefore much more efficient from the memory space viewpoint. It means no matter how much the input image-size is and no matter how complicated the content of the image is, the pyramid is built in at most three levels. Moreover, many of the redundant connections which have no effect on labeling task are identified and removed in parallel before the contraction operation. Finally, the algorithm represents the adjacency region graph (ARG) of the connected components and preserves the inclusion relationships such as [17], [18] on top of the pyramid.

The rest of this paper is organized as follows: Section 2 describes our proposed algorithm in details. Moreover, the mathematical proofs and parallelism of the proposed algorithm are explained in section 3. In section 4, the GPU implementation and the experimental results are described. Finally, the conclusion and future work are presented at section 5.

II. Pyramidal Connected Component Labeling Algorithm

In this section the Pyramidal Connected Component Labeling (PCCL) algorithm is presented in details. The proposed method consists of two main steps.The first one is to go up through the pyramid and contract all contraction kernels (CKs) to reach to the top level where remained vertices have the permanent unique labels. The second step is to move down from the top and propagate these permanent labels into each individual nodes at the base level. The height of the pyramid in the PCCL algorithm is at most three. It means one reaches to the final label for each connected component at most in the third level of the pyramid. The proposed algorithm removes the redundant edges in parallel and preserves the topology of the connected components. To create corresponding graph of the input binary image, the 4-neighborhood relationship between pixels is assumed. This results in the corresponding planar primal graph as illustrated in Figure 1.

Majid Banaeyan is supported by Pattern Recognition and Image Processing (PRIP) group via PRIP-Club.

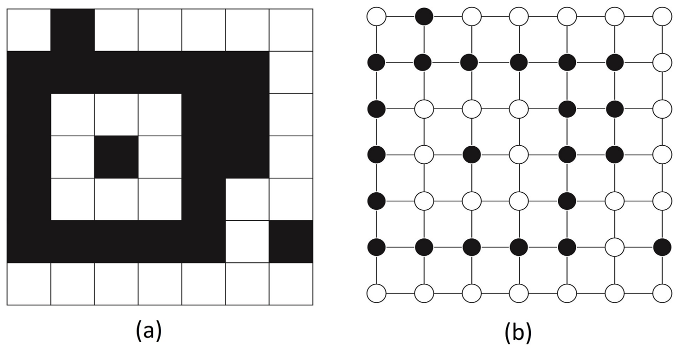

Figure 1. (a) The input binary image. (b) the corresponding primal graph at the base level of the pyramid.

Table I
PYRAMIDAL CONNECTED COMPONENT LABELING ALGORITHM.

```
PCCL Algorithm
Input: Image graph, G(d, $\alpha$(d), $\sigma$(d))
(Bottom-Up traversing)
0.1 Identifying the edge color
0.2 Selecting isolated vertices
1. while there are edges to contract do
1.1 Selecting the CKs
1.2 Defining the surviving vertices
1.3 Removing the Redundant Edges
1.4 Propagating the surviving vertices's labels
1.5 Contracting the selected CKs
End while
(Top-Down traversing)
2.Propagating the unique labels into each vertex (pixel) of the CCs
End
```

The steps of the algorithm are shown in the **Table I**. It should be noticed that in the combinatorial structure the fundamental elements are darts. Each dart (d), is defined as a half-edge. In addition, two functions namely the involution $\alpha(d)$ and the permutation $\sigma(d)$, completely encodes all the incident darts around a vertex. It means, an edge is encoded by the pair of ((d), α(d)) and the $\sigma(d)$ encodes the surrounding darts around each vertex by the counter-clock-wise orientation.

A. *Identifying the color of each edge*

Each node of the primal graph corresponding to an input binary image, has a unique index which is assigned by the top-down and left-right manner. In such primal graph, two colors to the edges are assigned. The Red-Edge (RE) and Black-Edge (BE). The former connects two adjacent vertices with different color and the later with the same color.

B. *Selecting isolated vertices*

After assigning color to all edges, now vertices that surrounded by only red edges identify as the isolated vertices. Note that the label (the index) of such isolated vertices is fix and will not change through the pyramid. In fact, these are the CCs that have only one pixel. In Fig. 2 the isolated vertices in the base level (with the indexes 1, 18, 48) are indicated with a blue circle around them.

C. *Selecting the contraction kernels (CKs)*

Every CK contracts two adjacent vertices belong to a connected component. Following rule, define how to select the contraction kernels:

Rule 1. Each black (white) vertex selects the maximum of bigger (smaller) vertices in its neighborhood.

In other words, if two vertices of a BE are black (white), the direction of the contraction kernel is East (North) or South (West) which the former one has the first priority. The CKs are shown in Fig. 2 with black arrows on the edges.

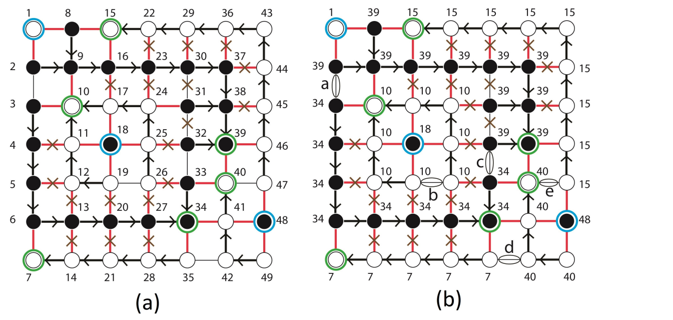

Figure 2. The base level of the pyramid, (a) The CKs are the black arrows. The isolated and surviving vertices are indicated with blue and green circle, respectively. The redundant edges indicated by brown cross.

D. *Defining the surviving vertices*

After selecting the CKs, beside the isolated vertices, the vertices are classified into tow categories; the non-surviving and surviving vertices. The surviving vertices are illustrated in Fig. 2 by a green circle around them. One of the main advantage of the Rule.1 is that there is an exclusive path of contraction Kernel(s) between each non-surviving vertex and its corresponding surviving one. The longest length of such contraction kernel path is 2N in a N by N binary image.

E. *Removing the Redundant Edges*

Redundant edges (connections) can be consider as edges inside the graph structure which by removing them the topology of the structure remains unchanged. Such connection usually are removed after contraction operation in graph pyramid. Based on the complexity of the image data, we may reach to an interwoven graph that need to be simplified due to move up through the pyramid. In general, removing such redundant edges performs in a sequential manner. Therefore, in this study, we are aiming at removing the redundant connections in parallel and before the contraction task .

Generally, there are two types of the redundant connections in the binary image. First, the redundant edges inside a connected component which are the black edges in our algorithm. Usually such redundant connections provide multiple (parallel) edges between two vertices. The second are the redundant red edges which are located between two adjacent connected components.

In this step of the algorithm, most of the redundant edges are

removed.
Note that the red edges play no rule in the labeling task and therefore if in this step some of them still remain, they do not interfere in the final result.

F. Propagating the survived vertices's labels

In order to reach into the topologically correct connected component labeling, the temporal labels need to be propagated through the corresponding exclusive paths. Therefore, all the vertices of contraction kernels before the contraction operation, receive their updated labels from their inverse directions of corresponding paths. This is illustrated in Fig. 2-b. This label propagation is the only sequential task in constructing the pyramid.

G. Contracting the selected CKs

At this point, all the CKs which already identified in step 2 are contracted at once. It is shown later that these contractions are performed simultaneously together. By contracting the CKs we reach to the second level of the pyramid. In the second level, all of the steps exactly similar to level one is performed and third (last) level of the pyramid is created. The algorithm terminates when there is no edge for contraction.
It should be noticed that those black edges which are remained on top of the pyramid represents the inclusion relationships between CCs. Through the contraction procedure, since they produces a non-empty loop they remains untouched. Figure 3 shows the second and third levels of the pyramid. In addition, the complete constructed pyramid with three levels illustrated in Figure 4.

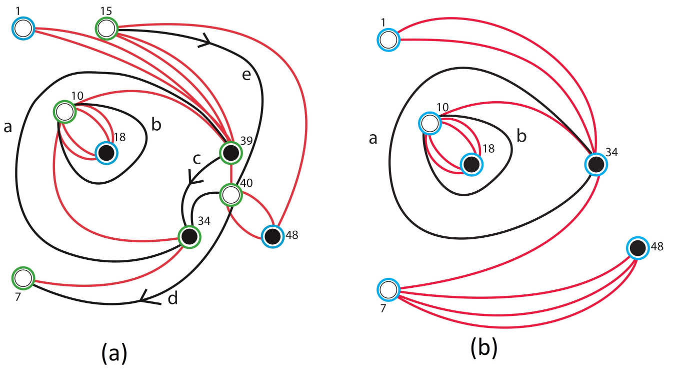

Figure 3. (a) The second level of the pyramid. (b) The third level (top) of the pyramid.

III. Parallelism of the PCCL Algorithm

In this section the parallelism of the proposed algorithm is investigated. Due to the size of an input image which is already known, the structure of the corresponding combinatorial map is available. It means the incident darts to each vertex are known. Note that, Without such assumptions, defining the relationship between vertex's number and its incident darts is also straight forward and takes only a few parallel steps.

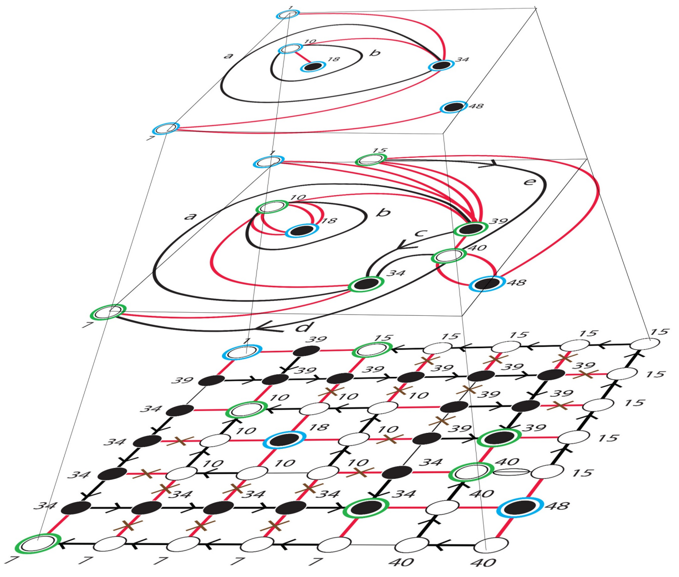

Figure 4. The constructed irregular graph pyramid with its three levels.

A. Identifying the edge color

To identify the color of each edge, its two corresponding vertices are compared. If they are the same the edge color is black and otherwise red. This is a local process for each dart (in Parallel).

B. Selecting isolated vertices

In this stage, the color of all incident edges to a vertex are checked. Vertices of the corners of the grid, boundary sides and those located inside have 2, 3 and 4 incident edges respectively. These maximum 4 values are compared together. If all of the incident edges to a vertex are red, then the vertex is the isolated one. This can be done at the same time for all vertices.

C. Selecting the CKs

Assigning a direction to each possible black edge defines the CK and the direction of the contraction. To this aim, it needs to check first the right (up) and second down (left) adjacent vertex for each black (white) vertex. Hence, for each vertex only two sequential comparisons are needed and therefore all of the computations perform in parallel.

D. Defining the surviving vertices

To see a vertex is a surviving or not, two conditions must be checked. First, a vertex must not be an isolated one. Second, it has not any outgoing black edges (it was not selected as CK in previous step). All of the vertices perform these two sequential $O(1)$ checking procedure in parallel.

E. Removing the redundant edges

Proposition 1: Each face does not have both redundant vertical and horizontal black edges together.

Proof 1: If we have two vertical and horizontal black edges in a same face then they must be connected and therefore one of them must be a CK (see Fig. 5-b).
Fig. 5-a shows the dependency between the set of darts around an edge in a 4-neighboring grid. By contracting (or removing) an edge(d,α(d)) the value of the other 4 incident darts, namely (σ(d), σ^{-1}(d), σ(α(d)), σ^{-1}(α(d))) need to be updated. Generally, if two edges do not have any of these 4 darts in share, they can be contracted (or removed) in parallel. Moreover, there is a special configuration of edges which the edges can be contracted (removed) in parallel even though they share two darts of above set. This situation happens when the CKs (removal edges) are placed in a exclusive path (a line).

Proposition 2: All the connected CKs (redundant edges) in a exclusive path (a line) are contracted (removed) in parallel.

Proof 2: Assume a set of connected redundant edges is given to be removed (see Fig. 5) (c) and (d) for black and red edges, respectively). To do the removal task, every redundant edge updates only two darts of 4 dependency dart set (σ^{-1}(d), σ^{-1}(α(d))) in red edges or (σ(d), σ(α(d))) in black ones. By doing this, there will be no remaining dart to be updated in connected redundant edges. Moreover, since there is also no overlapping between the removal darts, therefore, all can be removed in parallel.

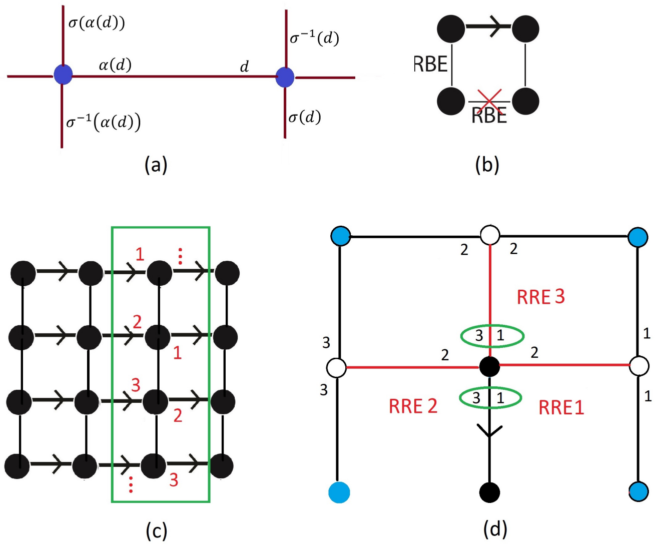

Figure 5. (a) The dart dependency set. (b) A black face has no two adjacent redundant black edges. (c) An example of redundant black edges. (d) An example of redundant red edges and a vertex incident these three edges.

F. Contracting the selected CKs

By defining the Rule.1 for contracting, all the CKs align in an exclusive path toward their surviving vertices. By propagating the labels before the contraction, the labels propagate from the surviving vertex to all corresponding non-surviving vertice(s) in the exclusive path of CKs. This is the only sequential part of the algorithm which in the worst case takes 2N steps in a N by N binary image. However, in future work we are aiming at using a static built pyramid to overcome such sequential obstacle.

G. Investigating the worst-case

The irregular graph pyramid is constructed in two steps till one reaches to the top of the pyramid. In constructing the proposed irregular pyramid, all the steps (except the propagation task) are implemented locally around the vertices in parallel way. Therefore, the parallel complexity of all the steps (except the propagation task) is $O(log(image-size))$ as the general complexity of the irregular pyramid. The complexity of the sequential label propagation step is depended into the largest length of the exclusive paths. Since the exclusive paths are disjoint, thus, each surviving vertex's label propagates in independent path and in parallel way. Therefore, the largest exclusive path indicates the sequential complexity of the algorithm. If the largest object in the $N \times N$-sized image contains the whole image, the length of the corresponding exclusive path is $2N$. Therefore, the sequential complexity in such worst case is $O(N))$. However, if we can independently select the CKs in a line, then, the complexity becomes $log(N)$. Next section explains the GPU implementation of the proposed algorithm.

IV. GPU implementation and the Experimental results

In the proposed algorithm the three levels of the pyramid are computed. Note that, each level has 5 sequential steps which individually are fully in parallel (except the step 4, propagating the surviving vertices's labels). Moreover, each available thread is dedicated to each vertex. In a GPU the shared memory (SM) is always faster than the global memory, therefore, the local processes are mostly employed by the shared memory. In fact, since all the processes which are performed in a vertex are local and independent, if a processing element exist for each vertex the algorithm is terminated in near the logarithmic complexity. As a result, the bottleneck of the algorithm is the capacity of the shared memory.
Consequently, in large images the number of vertices may be become greater than the number of the threads in the selected GPU. Therefore, in such a case, the threads must be waited until a busy SM finishes its task and is replaced by another vertex.
Our experiments were performed on the SUPERMICRO motherboard with Intel, E5-2697 v3, 128 GB DDR4 2133MHZ RAM and NVIDIA GeForce GTX 2080 TI.
In the experimental tests the image sets of the University of Southern California [19], ColumbiaUtrecht Reflectance and Texture Database [20] are used which the former includes different categories such as landscape, text image, fingerprint and the latter consists of seven textural images.
The result of experimental tests on the above GPU and the comparison with the other state-of-the-arts in [4] shows that our proposed PCCL algorithm outperforms the others and especially for big images (see Fig. 6).

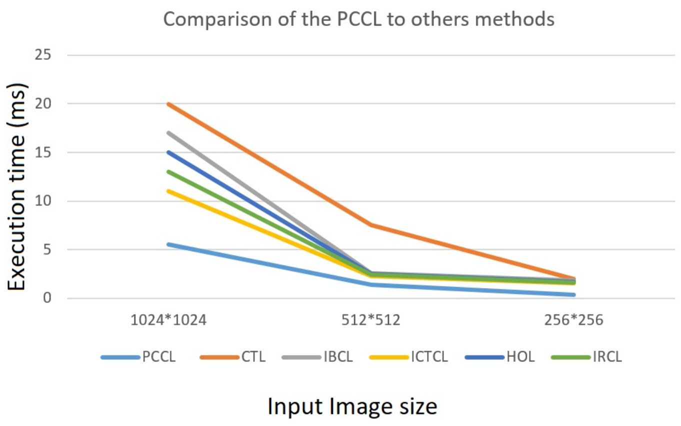

Figure 6. The comparision of the connected component labeling execution time over different image size. The proposed PCCL algorithm and the other state-of-the-art methods in [4].

V. Conclusion

The paper proposes a new parallel pyramidal connected component labeling. The main advantage of proposed algorithm is that the height of the constructed pyramid is bounded to three. Moreover, its complexity is $O(log(image - size))$ in a $N \times N$-sized binary image if a processing element exists for each vertex. In fact, the bottleneck is the number of the independent processing elements in shared memory at each employed hardware. In addition, the proposed algorithm preserves the topology of the image and represents the region adjacency graph between connected components including the inclusion relationships. The worst case is when a connected component has a maximum $(2N)$ path length and therefore its complexity is $O(N)$. Moreover, using the combinatorial pyramid structure provides further developments for higher dimensions (nD) and employing such algorithm to segment the gray-scale and color images in future. Finally, the parallel implementation of the algorithm on GPU shows that the experimental results outperforms the state of the art in connected component labeling field.

VI. References

References

[1] Q. Gu, T. Takaki, and I. Ishii, "A fast multi-object extraction algorithm based on cell-based connected components labeling," IEICE transactions on information and systems, vol. 95, no. 2, pp. 636–645, 2012.

[2] F. Zhao and Z.-y. Zhang, "Hardware acceleration based connected component labeling algorithm in real-time atr system," in Fifth International Conference on Machine Vision: Algorithms, Pattern Recognition, and Basic Technologies, vol. 8784. International Society for Optics and Photonics, 2013, p. 87841S.

[3] P. Sutheebanjard, "Decision tree for 3d connected components labeling," in International Symposium on Information Technologies in Medicine and Education, vol. 2. IEEE, 2012, pp. 709–713.

[4] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," Pattern Recognition, vol. 70, pp. 25–43, 2017.

[5] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," Computer Vision and Image Understanding, vol. 93, no. 2, pp. 206–220, 2004.

[6] J. Martín-Herrero, "Hybrid object labelling in digital images," Machine Vision and Applications, vol. 18, no. 1, pp. 1–15, 2007.

[7] L. He, Y. Chao, and K. Suzuki, "Two efficient label-equivalence-based connected-component labeling algorithms for 3d binary images," IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2122–2134, 2011.

[8] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," Computer Vision and Image Understanding, vol. 89, no. 1, pp. 1–23, 2003.

[9] U. H. Hernandez-Belmonte, V. Ayala-Ramirez, and R. E. Sanchez-Yanez, "A comparative review of two-pass connected component labeling algorithms," in Mexican International Conference on Artificial Intelligence. Springer, 2011, pp. 452–462.

[10] W. G. Kropatsch, Y. Haxhimusa, and P. Lienhardt, "Hierarchies relating topology and geometry," in Cognitive Vision Systems. Springer, 2006, pp. 199–220.

[11] Y. Haxhimusa, R. Glantz, M. Saib, G. Langs, and W. G. Kropatsch, "Logarithmic tapering graph pyramid," in Joint Pattern Recognition Symposium. Springer, 2002, pp. 117–124.

[12] L. Brun and W. Kropatsch, "Introduction to combinatorial pyramids," in Digital and Image Geometry. Springer, 2001, pp. 108–128.

[13] L. Brun and W. Kropatsch, "Combinatorial pyramids," in Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429), vol. 2. IEEE, 2003, pp. II–33.

[14] T. Wang, G. Dai, B. Ni, D. Xu, and F. Siewe, "A distance measure between labeled combinatorial maps," Computer Vision and Image Understanding, vol. 116, no. 12, pp. 1168–1177, 2012.

[15] M. Banaeyan, H. Huber, W. G. Kropatsch, and R. Barth, "A novel concept for smart camera image stitching," in Proceedings of the 21st Computer Vision Winter Workshop 2016, L. Čehovin, R. Mandeljc, and V. Štruc, Eds., 2016, pp. 1–9.

[16] M. Cerman, I. Janusch, R. Gonzalez-Diaz, and W. G. Kropatsch, "Topology-based image segmentation using lbp pyramids," Machine Vision and Applications, pp. 1–14, 2016.

[17] R. González Díaz, W. G. Kropatsch, M. Cerman, and J. Lamar León, "Characterizing configurations of critical points through LBP extended abstract," Image-A: Applicable Mathematics in Image Engineering, 4 (7), 2015.

[18] F. Torres and W. G. Kropatsch, "Canonical encoding of the combinatorial pyramid," in Proceedings of the 19th Computer Vision Winter Workshop, 2014, pp. 118–125.

[19] "SIPI Image Database," http://sipi.usc.edu/database/, Accessed: 2021-02-01.

[20] "Department of Computer Science, Columbia University," http://www.cs.columbia.edu/, Accessed: 2021-02-01.