Connected Components Labeling on Bitonal Images

Federico Bolelli, Stefano Allegretti, and Costantino Grana

Università degli Studi di Modena e Reggio Emilia, Italy
Dipartimento di Ingegneria "Enzo Ferrari"
{*name.surname*}@unimore.it

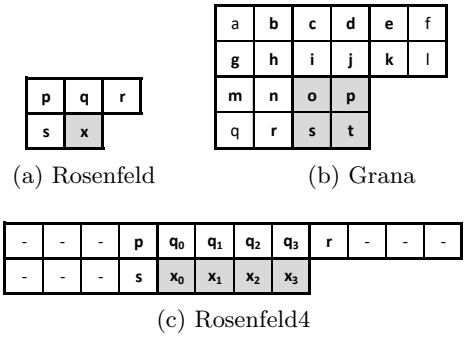
Abstract. Several algorithmic solutions for the optimization of Connected Components Labeling have been proposed in literature. Among them, one of the most effective is a block-based mask to drastically reduce the number of memory accesses during the labeling procedure. This paper proposes a systematic approach for labeling multiple pixels at once, automatically generating the actions to be performed on the current pixel/block given the mask values. The proposed strategy allows to extend existing techniques for the generation of optimal decision trees to much more complex masks, where the connectivity between pixels inside a block is not guaranteed. A showcase application, consisting in the design of an efficient CCL algorithm for bitonal images, demonstrates the effectiveness of our proposal in terms of speed and memory footprint.

Keywords: Connected Components Labeling · Bitonal Images · Optimal Decision Trees

1 Introduction

The task of labeling connected components, also known as Connected Components Labeling or CCL in short, aims at producing a description of the objects inside binary images, by generating a symbolic image where each pixel of a single connected component (object) is assigned a unique identifier. Objects inside binary images are usually defined according to the pixel neighborhood, which can be either *4-* or *8-neighborhood* for 2D-images. The rest of the paper will focus on the 8-neighborhood.

Connected components labeling represents a fundamental pre- and post-processing step for many Computer Vision and Image Processing pipelines [3, 6, 8, 11, 12, 14, 18, 25, 26, 28, 29, 31]. CCL has an exact output, and therefore different algorithmic solutions are only compared in term of speed and memory footprint. After the introduction of the task in the Sixties, several proposals were made in the course of decades to optimized its computational load, both for sequential [5, 20, 24, 32] and parallel architectures [1, 23, 27, 33]. Among the different algorithmic solutions, block-based scan approaches (i.e. label a block of 2×2 pixels at once) [9, 10, 17], decision trees [19, 32] and state prediction [15, 22] (i.e. reuse the information gathered during the previous step when labeling



(a) Rosenfeld (b) Grana

(c) Rosenfeld4

Fig. 1. Example of scan masks. Gray squares identify current pixels to be labeled using information extracted from white pixels. (a) and (b) are very common masks employed in CCL (c) is an extended version of the Rosenfeld mask that is proposed and analyzed in this paper. In this specific case, the current pixels, i.e. x_0, x_1, x_2, x_3, do not necessarily share the same label. Dashes identify meaningless pixels.

the current pixel/block) revealed to be some of the most valuable strategies, especially when combined together [4].

Binary images can be efficiently stored with only 1 bit per pixel ("1-bit graphics" format, or bitonal images). This representation is especially useful in embedded systems with limited resources, where memory usage must be reduced to a minimum. In banking, as an example, the bitonal image is the legally recognized standard for electronic check clearing in the United States and many other countries. Working with 1-bit per pixel images (also denoted as 1-bpp or bitonal images) allows to considerably reduce the amount of memory accesses; on the other hand, it also requires additional bitwise operations for retrieving single pixel values.

In the context of 1-bpp images, being able of labeling an entire byte as a single block would guarantee a significant performance improvement, without requiring to convert the input into a 1-byte per pixel image. Unfortunately, the assumption that foreground pixels are always connected inside a block does not hold in such a case, and algorithms proposed in literature for the automatic generation of binary decision trees are not feasible. This paper introduces a systematic approach for generating all the possible actions associated to a scanning mask, which is employed to design an extremely fast CCL algorithm for bitonal images capable of labeling four consecutive pixels at once.

The rest of this paper is organized as follows. Section 2 resumes the latest contributions on connected components labeling; the proposed strategy is described in Section 3, and the result is evaluated in Section 4. Finally, in Section 5 conclusions are drawn.

2 Related Work

Originally introduced by Rosenfeld and Pfaltz [30], connected components labeling has a very long story, full of different strategies and proposals. Since its first appearance in 1966, many papers showed algorithms to improve the efficiency of the task. Traditionally, the fastest CCL algorithms employ a two scan strategy. In the first scan, each pixel is assigned a provisional label determined using a mask of already visited pixels, such as the one in Fig. 1a, and possible equivalences between labels are recorded. Then, a representative label is established for each connected component, and the second scan replaces provisional labels with final ones.

Several strategies have been proposed for the resolution of label equivalence, and the most commonly seen in literature employ some variation of the union-find. The union-find data structure, first applied to CCL by Dillencourt *et al.* [13], provides two convenient procedures to deal with equivalence classes of labels: *Find*, which retrieves the representative label of an equivalence class, and *Union*, which merges two equivalence classes into one, ensuring that they share the same representative label.

After the introduction of union-find, a significant improvement was provided by Wu *et al.* in [32]. The authors proved an optimal strategy, based on a manually identified decision tree, to reduce the average number of load/store operations during the first scan of the input image, driven by the Rosenfeld mask (Fig. 1a). The resulting algorithm have been christened *Scan Array-based Union Find*, or SAUF in short.

In 2010, Grana *et al.* [17] introduced a major breakthrough, consisting in a 2×2 block-based approach (Fig. 1b). The problem was modeled as a *command execution metaphor*: values of pixels in the scanning mask constitute a *rule* (binary string), which is associated to a set of equivalent actions in an *OR*-decision table (Fig. 2). Given this decision table, an algorithm can simply read all the pixels inside the mask, identify the rule, and find the action to be performed in the corresponding column. In [19], a dynamic programming approach to convert *OR*-decision tables into optimal binary decision trees was proposed, in order to minimize the average number of conditions to be checked when choosing the correct action to be performed. The resulting algorithm is denoted as *Block-Based Decision Tree*, or BBDT. Many improvements were published since then [21].

In 2014, He *et al.* [22] demonstrated the possibility to use a finite state machine to summarize the value of already inspected pixels during the horizontal mask movement.

In [15], decision trees and configuration transitions are combined in a decision forest, where each previous pattern allows to "predict" some of the current configuration pixel values, thus allowing for automatic code generation. The first scan phase of the algorithm is ruled by a forest of decision trees connected into a single graph, where each tree derives from a reduction of the complete optimal decision tree.

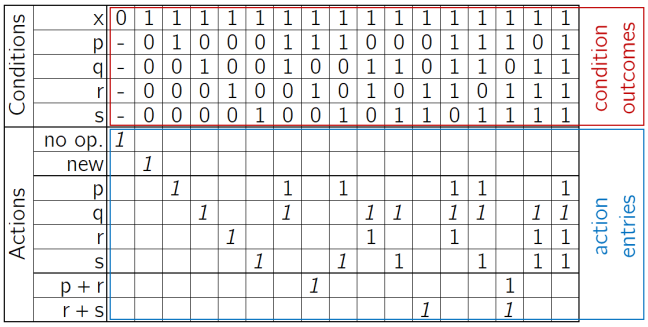


Fig. 2. *OR*-decision table for the Rosenfeld mask.

Additionally, in [7] Bolelli *et al.* demonstrated that switching from decision trees to Directed Rooted Acyclic Graphs (DRAGs) allows for a reduction of machine code footprint, thus lessening the impact on instruction cache.

Finally, in [4] authors managed to combine the block-based mask with state prediction and code compression: the resulting algorithm, known as *Spaghetti Labeling*, was modeled as a Directed Rooted Acyclic Graph with multiple entry points, automatically generated without manual intervention.

3 Method

In this section, the proposed method for labeling multiple pixels at once is presented. As usual, CCL is performed with two raster scans of the input, here briefly summarized.

The first scan employs a mask moving in discrete steps, which highlights the current pixel(s) to be labeled and its neighborhood, composed of already analyzed and labeled pixels; at each step, the current pixel is assigned a provisional label, and if it connects two or more connected components, their labels are recorded as equivalent by means of some variation of Union-Find [7]. This set of operations carried out for a certain mask position is known as *action*, and depends on the values of pixels inside the mask, which form a binary word known as *command* [17].

Then, the second scan simply replaces each provisional label with the chosen representative for the equivalence class, thus completing the task. While the second scan is usually fixed and nearly identical for most algorithms, the first scan is where algorithmic proposals differ the most: here several optimizations can take place, such as the aforementioned decision tree (decide the action without reading the whole command word), block-based strategy (label multiple connected pixels at once), prediction (avoid to re-read neighbor pixels known from the previous step), and compression (reduce machine code size by merging equivalent subtrees of a larger decision tree).

The new technique proposed with this work extends the block-based approach, by overcoming the limitation that all pixels to be labeled at once must

be connected. In fact, with respect to previous proposals [9, 10, 17], limited to a block size of at most 2×2 pixels, the devised algorithm can be applied to blocks of any shape.

In the following, the term *macroblock* identifies the pixels of the mask to be labeled during the current step. A macroblock can be divided into disjoint *blocks*, each of which always contains pixels connected to each other. This ensures that a block can always be assigned only one label.

On the other hand, it is not possible to assume that only one single action is performed on the current macroblock. Taking the mask of Fig. 1c as an example, the macroblock is composed by pixels x_0, x_1, x_2, and x_3 which can be divided into two different blocks: $x_0, x_1 \in X_0$ and $x_2, x_3 \in X_1$.

In this context, it may happen that, e.g., block X_0 requires a *new label*, while X_1 must be assigned the result of a *merge* —the union procedure of the union-find data structure— of two different existing label classes.

In literature, no attempt has been made to systematically generate the set of actions associated to a macroblock-based scan mask. The block-based mask proposed in [17] (Fig. 1b), for example, shares the same actions of the Rosenfeld mask, with the only addition of some merge operations that have no effect in a pixel-based context.

Let us start with some formal definitions. Be $I : \mathcal{L} \to \{B, F\}$ a binary image, *i.e.*, a function defined over a 2-dimensional square lattice $\mathcal{L}$, where pixels only assume two possible values, background (B) and foreground (F), usually represented by integers 0 and 1 respectively.

The 8-neighborhood of pixel $p = (p_r, p_c) \in \mathcal{L}$, denoted as $\mathcal{N}_8(p)$, is the set of pixels sharing an edge or vertex with p:

$$\mathcal{N}_8(p) = \{q = (q_r, q_c) \in \mathcal{L} \mid max(|p_r - q_r|, |p_c - q_c|) \leq 1\} \tag{1}$$

Given $S \subset \mathcal{L}$, pixels $p, q \in S$ are *connected in* S, denoted as $p \diamond_S q$, if a path of neighbor foreground pixels exists, all belonging to S and leading from p to q:

$$p \diamond_S q \Leftrightarrow \exists\{s_i \in S \mid I(s_i) = F \wedge s_1 = p, s_{n+1} = q, s_{i+1} \in \mathcal{N}_8(s_i), \forall i = 1, ..., n\} \tag{2}$$

Connectivity in S is an equivalence relation, since the properties of *reflexivity*, *symmetry* and *transitivity* hold. Equivalence classes of this relation are called *Connected Components (CCs)* of S. When $S = \mathcal{L}$, we omit the subscript in the notation $p \diamond q$, and CCs of $\mathcal{L}$ are referred to as just CCs.

To better detail the proposed algorithmic solution, we divide the pixels in the mask in two subsets:

- *Outer Pixels (P_O)*, pixels inside the mask but outside the macroblock. Outer pixels already have a provisional label, since they have already been analyzed by the algorithm.
- *Inner Pixel (P_I)*, pixels inside the macroblock. Inner pixels must be assigned a provisional label in the current step.

As an example, in Fig. 1c, $P_O = \{p, q_0, q_1, q_2, q_3, r, s\}$, while $P_I = \{x_0, x_1, x_2, x_3\}$. In order to proceed with the generation of the action set, the following operations are required for each configuration of the mask:

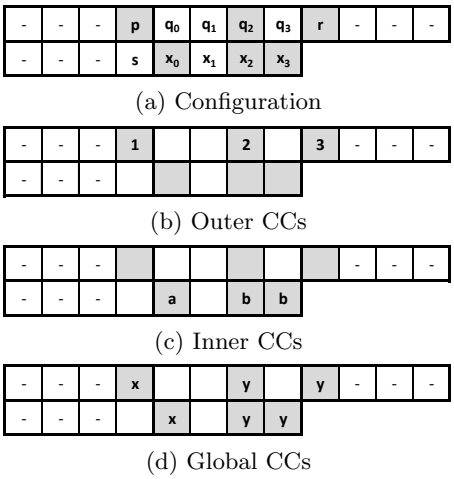


Fig. 3. Example of the proposed action(s)-generation algorithms when applied to the mask configuration depicted in (a). Gray squares identify foreground pixels inside the mask. The pixels to be labeled using the information extracted from all the others are x_0, x_1, x_2, and x_3. Together they constitute the inner part of the mask. Remaining pixels are the outer part of the mask. Dashes identify meaningless values.

- Identify the CCs of P_O; this set of outer connected components is denoted as C_O;
- Identify connected components of P_I; these inner connected components are denoted as C_I;
- Identify the connected components for the whole mask, i.e., CCs for $P_O \cup P_I$; these are denoted as C_T;
- For each $t \in C_T$, consider all inner pixels belonging to t (i.e. the set $t \cap P_I$); all these pixels must be assigned the same label l. This label is determined analyzing the set of outer connected components contained in t, denoted as $C_O^t = \{c \in C_O \mid c \subset t\}$. If $C_O^t = \varnothing$, then a new label is created for l. If $\#C_O^t = 1$, then l can be assigned the label of any pixel of $c \in C_O^t$. Finally, if $\#C_O^t > 1$, all CCs in C_O^t must be merged, meaning that their labels are marked as equivalent and l is set to any of them (typically the smallest).

It is important to stress that $C_T \neq C_O \cup C_I$, and that the external components are defined without considering connections through pixels currently under examination (the pixels of the macroblock). The same goes for internal components, where we do not consider connections due to external pixels. Those connections are considered only for C_T.

An example of action generation is reported in Fig. 3. In Fig. 3a, a mask configuration is shown: the gray squares represent foreground pixels, x_0, x_1, x_2, x_3 are the pixels in the "current" macroblock, and dashes identify meaningless pixels. The process starts with the detection of the connected components in the outer part of the mask, i.e. ignoring the current macroblock (Fig. 3b). In this

specific example, three different objects are in C_O, respectively named 1, 2, 3. Then, inner connected components are identified inside the macroblock: a and b as in Fig. 3c. Finally global connected components (C_T), x and y, are depicted in Fig. 3d. In particular, CC x contains both the inner component a and the outer component 1, so from this configuration we can derive the first operation to be performed, $a = 1$, that easily translates into action $x_0 = p$. This action means: *assign to* x_0 *the same label previously assigned to* p. Moreover, CC y contains inner component b and outer components 2 and 3. In this case the operation is $b = 2 + 3$: *assign to all the pixels of connected component* b *the result of the merge between components* 2 *and* 3. Translating this operation into an action, we obtain $x_2\ x_3 = q_2 + r$, that is *assign to pixels* x_2 *and* x_3 *the merge of labels previously assigned to pixels* q_2 *and* r.

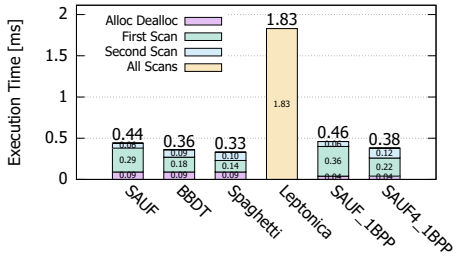
In order to give the reader an additional example, we can consider the same configuration of Fig. 3a and add q_1 as foreground. In this case, outer and inner components are the same of Fig. 3b and Fig. 3c, but component 2 is made of two pixels (q_1 and q_2) instead of just one. On the other hand, we obtain just one single global component. This causes the algorithm to generate the action $x_0\ x_2\ x_3 = p + q_1\ q_2 + r$: x_0, x_2, *and* x_3 *must be assigned the result of the merge between* p, *one between* q_1 *and* q_2, *and* r. Actually, the *or* between q_1 and q_2 is responsible for the generation of two equivalent actions, $x_0\ x_2\ x_3 = p + q_1 + r$ and $x_0\ x_2\ x_3 = p + q_2 + r$. Most equivalence cases can be resolved with the block-based approach described in the following; the others are treated during the generation of the optimal decision tree, as explained in [19]. Basically, each global connected component generates one or multiple equivalent actions, responsible for the labeling of all pixels belonging to one or more internal components.

3.1 Reducing Actions with Blocks

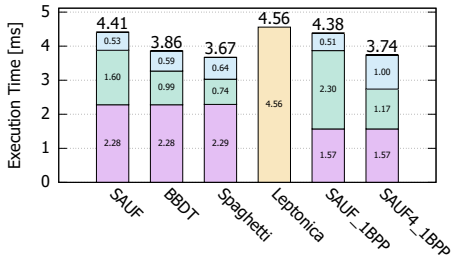
The number of actions generated with the proposed approach grows very quickly as the mask size increases, making the generation of the optimal decision tree extremely hard or even impossible. In order to reduce the number of actions and simplify the problem, we introduce a block-based approach. As described above, macroblocks are divided into blocks, and pixel-based actions are replaced with block-based ones, eliminating possible duplicates. This way, many of the previous actions translate into the same one, and can be removed.

As an example, let us consider the following pixel-based actions: $x_0 = q_0$ and $x_1 = q_0\ q_1$, the latter actually representing the two equivalent actions $x_1 = q_0$ and $x_1 = q_1$. Since x_0 and x_1 are always connected, they can be viewed as part of the single block X_0, and the same applies to q_0 and q_1, which are part of the block Q_0. Thus all the three aforementioned pixel-based actions can be fused into $X_0 = Q_0$, producing the same outcome.

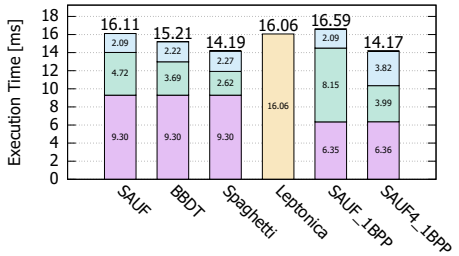
As previously described in literature [17, 4], when working with block-based actions the second scan of an algorithm requires a slight overhead to correctly handle blocks, i.e. assigning the block label to all foreground pixels belonging to that block. Obviously, the reduction in the number of actions can be more or less significant depending on the mask features. For what concerns the mask of



(a) *Fingerprints*



(b) *Medical*



(c) *Tobacco800*

Fig. 4. Average run-time tests with steps in ms (lower is better).

Fig. 1c, actions reduce of about 80% (from 413 to 85 actions) when moving to the block-based approach. After generating all the possible actions associated to a scan mask and the corresponding *OR*-decision table, the algorithm described in [19] can be employed for the generation of an optimal decision tree, which maps the mask configuration to an action, minimizing the average number of load/store operations required.

The described approach has been employed to generate an optimal decision tree for the mask of Fig. 1c, which constitutes the core of a new CCL algorithm, specifically designed for 1-bpp images. Since it shares the general structure of SAUF, but operates on 4-pixel macroblocks, it is referred to as SAUF4_1BPP.

4 Experimental Results

The performance evaluation of the proposed algorithm has been carried out with an extended version of the YACCLAB benchmark [7, 16], an open source C++ framework specifically designed to test CCL algorithms. In order to incorporate a standard well-known implementation of bitonal images, the benchmark has been integrated with Leptonica, an open-source image processing library employed in several projects (e.g. Tesseract OCR by Google). The extended version of the YACCLAB benchmark, including the proposed algorithm implementation, is available at https://github.com/prittt/YACCLAB.

Experimental results discussed in the following were obtained on an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz with Windows 10.0.17134 (64 bit) OS and MSVC 19.15.26730 compiler. Our proposal is evaluated on three datasets: *Fingerprints*, *Medical* and *Tobacco800*, which cover the most common CCL application fields, a full description can be found in [2]. Fig. 4 highlights how the performance of algorithms is influenced by the different phases they are composed of: memory management, first scan and second scan.

The selected algorithms for comparison are SAUF, BBDT, Spaghetti, and the CCL implementation available in Leptonica. The first three algorithms, mentioned in Section 2, represent the state of the art regarding 1-byte per pixel images; for a fair comparison, their first scan times also include a conversion of the input to their preferred format. SAUF_1BPP and SAUF4_1BPP, finally, are the 1-bpp algorithms introduced in this paper. The former is a simple adaptation of SAUF, which iterates over the eight pixels stored in each byte; the latter employs a decision tree generated starting from the mask of Fig. 1c, employing the action generation algorithm introduced with this paper. All the algorithms employ the classic union-find label solver [32].

The Leptonica algorithm is based on a seedfill approach which, as can be observed, is extremely inefficient when connected components extend vertically (e.g. *Fingerprints*), causing a series of non cache-friendly memory accesses. On the other hand, when small size CCs constitute the images, Leptonica has comparable performance with SAUF_1BPP.

The main proposal of this work, SAUF4_1BPP, considerably exceeds the performance of Leptonica, with a speedup ranging from 1.13 to 4.81 depending on the dataset, and thus represents the currently most efficient CCL algorithms designed to work on bitonal images. Moreover, SAUF4_1BPP has comparable performance to Spaghetti (current state of the art for CCL on binary images), when the latter needs a prior conversion of the input. However, SAUF_1BPP only requires about $1/9\times$ memory for the input data, making it an excellent choice for use cases where memory size is constrained.

5 Conclusion

An effective solution to automatically map a connected components labeling scan mask configuration with the actions to be performed has been presented, which

allows to label multiple pixels at each mask shift. A CCL algorithm generated using this systematic approach is presented, which outperforms competitors on bitonal images, confirming the effectiveness of the method.

References

1. Allegretti, S., Bolelli, F., Cancilla, M., Pollastri, F., Canalini, L., Grana, C.: How does Connected Components Labeling with Decision Trees perform on GPUs? In: Computer Analysis of Images and Patterns. vol. 11678, pp. 39–51 (2019)
2. Allegretti, S., Bolelli, F., Grana, C.: Optimized Block-Based Algorithms to Label Connected Components on GPUs. IEEE Transactions on Parallel and Distributed Systems pp. 423–438 (2019). https://doi.org/10.1109/TPDS.2019.2934683
3. Allegretti, S., Bolelli, F., Pollastri, F., Longhitano, S., Pellacani, G., Grana, C.: Supporting Skin Lesion Diagnosis with Content-Based Image Retrieval. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (Jan 2021)
4. Bolelli, F., Allegretti, S., Baraldi, L., Grana, C.: Spaghetti Labeling: Directed Acyclic Graphs for Block-Based Connected Components Labeling. IEEE Transactions on Image Processing **29**(1), 1999–2012 (2019)
5. Bolelli, F., Baraldi, L., Cancilla, M., Grana, C.: Connected Components Labeling on DRAGs. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 121–126 (2018)
6. Bolelli, F., Borghi, G., Grana, C.: XDOCS: An Application to Index Historical Documents. In: Digital Libraries and Multimedia Archives. pp. 151–162. Springer (2018)
7. Bolelli, F., Cancilla, M., Baraldi, L., Grana, C.: Towards reliable experiments on the performance of Connected Components Labeling algorithms. Journal of Real-Time Image Processing **17**(2), 229–244 (2018)
8. Canalini, L., Pollastri, F., Bolelli, F., Cancilla, M., Allegretti, S., Grana, C.: Skin Lesion Segmentation Ensemble with Diverse Training Strategies. In: Computer Analysis of Images and Patterns. pp. 89–101 (2019)
9. Chang, W.Y., Chiu, C.C.: An efficient scan algorithm for block-based connected component labeling. In: 22nd Mediterranean Conference on Control and Automation. pp. 1008–1013 (2014)
10. Chang, W.Y., Chiu, C.C., Yang, J.H.: Block-Based Connected-Component Labeling Algorithm Using Binary Decision Trees. Sensors **15**(9), 23763–23787 (2015)
11. Cipriano, M., Allegretti, S., Bolelli, F., Di Bartolomeo, M., Pollastri, F., Pellacani, A., Minafra, P., Anesi, A., Grana, C.: Deep Segmentation of the Mandibular Canal: a New 3D Annotated Dataset of CBCT Volumes. IEEE Access pp. 1–11 (2022)
12. Cipriano, M., Allegretti, S., Bolelli, F., Pollastri, F., Grana, C.: Improving Segmentation of the Inferior Alveolar Nerve through Deep Label Propagation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–10. IEEE (2022)
13. Dillencourt, M.B., Samet, H., Tamminen, M.: A General Approach to Connected-Component Labeling for Arbitrary Image Representations. Journal of the ACM **39**(2), 253–280 (1992)
14. Fabbri, M., Brasó, G., Maugeri, G., Cetintas, O., Gasparini, R., Ošep, A., Calderara, S., Leal-Taixé, L., Cucchiara, R.: MOTSynth: How Can Synthetic Data Help Pedestrian Detection and Tracking? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10849–10859 (2021)

15. Grana, C., Baraldi, L., Bolelli, F.: Optimized Connected Components Labeling with Pixel Prediction. In: ACIVS. pp. 431–440 (2016)
16. Grana, C., Bolelli, F., Baraldi, L., Vezzani, R.: YACCLAB - Yet Another Connected Components Labeling Benchmark. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 3109–3114 (2016)
17. Grana, C., Borghesani, D., Cucchiara, R.: Optimized Block-based Connected Components Labeling with Decision Trees. IEEE Transactions on Image Processing **19**(6), 1596–1609 (2010)
18. Grana, C., Borghesani, D., Cucchiara, R.: Automatic segmentation of digitalized historical manuscripts. Multimedia Tools and Applications **55**(3), 483–506 (2011)
19. Grana, C., Montangero, M., Borghesani, D.: Optimal decision trees for local image processing algorithms. Pattern Recognition Letters **33**(16), 2302–2310 (2012)
20. He, L., Chao, Y., Suzuki, K.: A Linear-Time Two-Scan Labeling Algorithm. In: 2007 IEEE International Conference on Image Processing. pp. 241–244 (2007)
21. He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., Chao, Y.: The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition **70**, 25–43 (2017)
22. He, L., Zhao, X., Chao, Y., Suzuki, K.: Configuration-Transition-Based Connected-Component Labeling. IEEE Transactions on Image Processing **23**(2), 943–951 (2014)
23. Hennequin, A., Lacassagne, L., Cabaret, L., Meunier, Q.: A new Direct Connected Component Labeling and Analysis Algorithms for GPUs . In: 2018 Conference on Design and Architectures for Signal and Image Processing (DASIP). pp. 76–81. IEEE (2018)
24. Lacassagne, L., Zavidovique, B.: Light speed labeling: efficient connected component labeling on risc architectures. Journal of Real-Time Image Processing **6**(2), 117–135 (2011)
25. Laradji, I.H., Rostamzadeh, N., Pinheiro, P.O., Vazquez, D., Schmidt, M.: Where are the Blobs: Counting by Localization with Point Supervision. In: Computer Vision – ECCV 2018. pp. 547–562 (2018)
26. Pham, H.V., Bhaduri, B., Tangella, K., Best-Popescu, C., Popescu, G.: Real time blood testing using quantitative phase imaging. PloS one **8**(2), e55676 (2013)
27. Playne, D., Hawick, K.: A new algorithm for parallel connected-component labelling on gpus. IEEE Transactions on Parallel and Distributed Systems **29**(6), 1217–1230 (2018). https://doi.org/10.1109/TPDS.2018.2799216
28. Pollastri, F., Bolelli, F., Paredes, R., Grana, C.: Augmenting Data with GANs to Segment Melanoma Skin Lesions. Multimedia Tools and Applications **79**(21-22), 15575–15592 (2019)
29. Porrello, A., Abati, D., Calderara, S., Cucchiara, R.: Classifying signals on irregular domains via convolutional cluster pooling. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 1388–1397. PMLR (2019)
30. Rosenfeld, A., Pfaltz, J.L.: Sequential Operations in Digital Picture Processing. J. ACM **13**(4), 471–494 (1966)
31. Uslu, F., Bharath, A.A.: A recursive Bayesian approach to describe retinal vasculature geometry. Pattern Recognition **87**, 157–169 (2019)
32. Wu, K., Otoo, E., Suzuki, K.: Two Strategies to Speed up Connected Component Labeling Algorithms. Pattern Analysis Application **0**(LBNL-59102) (2005)
33. Zavalishin, S., Safonov, I., Bekhtin, Y., Kurilin, I.: Block Equivalence Algorithm for Labeling 2D and 3D Images on GPU. EI **2016**(2), 1–7 (2016)