Connected component labeling techniques on modern architectures

Costantino Grana, Daniele Borghesani, Rita Cucchiara

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Modena e Reggio Emilia, Via Vignolese 905/b, 41100 Modena, Italy
{costantino.grana, daniele.borghesani, rita.cucchiara}@unimore.it

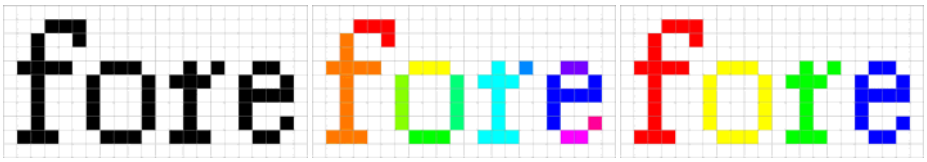
Abstract. In this paper we present an overview of the historical evolution of connected component labeling algorithms, and in particular the ones applied on images stored in raster scan order. This brief survey aims at providing a comprehensive comparison of their performance on modern architectures, since the high availability of memory and the presence of caches make some solutions more suitable and fast. Moreover we propose a new strategy for label propagation based on a 2x2 blocks, which allows to improve the performance of many existing algorithms. The tests are conducted on high resolution images obtained from digitized historical manuscripts and a set of transformations is applied in order to show the algorithms behavior at different image resolutions and with a varying number of labels.

Keywords: connected component labeling, comparison, union-find.

1 Introduction

Connected component labeling is a fundamental task in several computer vision applications. It is used as a first step in the task chain in many problems, e.g for assigning labels to segmented visual objects, and for this reason a fast and efficient algorithm is undoubtedly very useful. A lot of techniques have been proposed in literature in the past; most of them referred to specific hardware architectures to take advantage of their characteristics, but nowadays, modern architectures do not suffer anymore of such limitations that constitute a design priority of some of these algorithms. In this paper, a brief survey of traditional and new labeling techniques is presented and a comparison of some labeling techniques is reported in order to find out the real performances of these proposals on modern computer architectures.

Moreover, Intel has released a precious set of libraries as an open source project named OpenCV. These libraries contain an implementation of all the main algorithms useful in computer vision applications and include two strategies for connected component analysis: a contour tracing (cvFindContours) followed by a contour filling (cvDrawContours), or a flood fill approach (cvFloodFill) which can be applied sequentially to all foreground pixels. We will also consider these two approaches in the comparison.



a) *b)* *c)*

Fig. 1. Example of binary image depicting text (*a*), its labeling considering 4-connectivity (*b*), and 8-connectivity (*c*).

Beside an extensive review of labeling algorithms, the main contribution of this work is a new block based scanning strategy, which allows to substantially improve the performance of the most common class of algorithms, namely the raster scan one.

We will exclude two wide classes of algorithms from our analysis. The first one is the class of parallel algorithms which has been extensively studied up to the first half of the '90s. These algorithms were aimed to specific massively parallel architectures and do not readily apply to current common workstations, which provide more and more parallelism (instruction level, thread level and so on), but substantially different from the parallelism exploited in those algorithms. The second class is given by algorithms suitable for hierarchical image representations (for example quadtrees) initially studied for accessing large images stored in secondary memory. We excluded them because the vast majority of images is currently stored in sequential fashion, since they can often be fully loaded in main memory.

After formalizing the basic concepts needed, we review of some of the most used labeling algorithms, the newest ones and then we detail our proposal. The different algorithms performance are evaluated on a high resolution image dataset, composed of documental images with a large number of labels. Different modifications are performed to test these algorithms in several situations in order to show which is the most effective algorithm in different conditions.

2 Neighborhood and Connectivity

Two pixels are said to be 4-*neighbors* if only one of their image coordinates differs of at most one, that is if they share a side when viewed on a grid. They are said to be *8-neighbors* if one or both their image coordinates differ of at most one, that is if they share a side or a corner when viewed on a grid.

A subset of a digitized picture, whose pixels share a common property, is called *connected* if for any two points P and Q of the subset there exists a sequence of points $P = P_0, P_i, \ldots, P_{n-1}, P_n = Q$ of the subset such that P_i is a neighbor of $P_{i-1}, 1 \le i \le n$ [1].

The common choice in binary images, where the property of interest is to be part of the "foreground" with respect to the "background", is to choose 8-connectivity, that is connectivity with 8-neighbors, for the foreground regions, and 4-connectivity for background regions. This usually better matches our usual perception of distinct objects, as in Fig. 1.

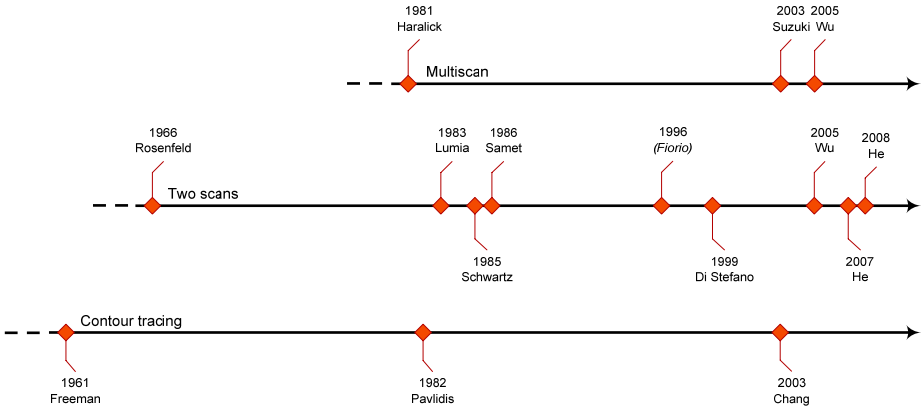


Fig. 2. Timeline showing the evolution of the labeling algorithms.

In binary images, the “labeling” procedure is the process of adding a “label” (an integer number) to all foreground pixels, guaranteeing that two points have the same label if and only if they belong to the same connected component.

3 The evolution of labeling algorithms

The problem of labeling has been deeply studied since the beginning of Computer Vision science. In the following we try to provide a historical view of the different approaches, discussing how they contributed to current approaches and if their purposes are still applicable to modern architectures.

The first work proposed for image labeling date back to Rosenfeld *et al.* in 1966 [1], and this can be considered the very classical approach to labeling. It is based on a raster scan of the image and, rather than generate an auxiliary picture, the “redundancies” of the labels are stored in an equivalences table with all the neighborhood references. The redundancies are solved processing the table by repeatedly using an unspecified sorting algorithm and removing redundant entries, consequently requiring an high amount of CPU power. Finally the resulting labels are updated in an output image with a single pass, exploiting the solved equivalences table.

A problem of the original algorithm is the use of a second image to store labels and of another structure to store equivalences. To tackle this problem an improvement has been proposed by Haralick *et al.* [2]. This algorithm does not use any equivalences table and no extra space, by iteratively performing forward and backward raster scan passes over the output image to solve the equivalences exploiting only local neighborhood information. This technique clearly turns out to be very expensive when the size of the binary image to analyze increases.

Lumia *et al.* [3] observe that both previous algorithms perform poorly on ’83 virtual memory computers because of page faults, so they mix the two approaches trying to keep the equivalences table as small as possible, saving memory usage. In

this algorithm a forward and a backward scan are sufficient to complete the labeling, but at the end of each row the collected equivalences are solved and another pass immediately updates that row labels. This suggests that four passes over the data are indeed used by this algorithm. The technique to solve label equivalences is left unspecified.

Schwartz *et al.*[4] further explored on this approach, in order to avoid the storage of the output image, which would have required too much memory. Thus they use a sort of run length based approach (without naming it so), which produces a compact representation of the label equivalences. In this way, after a forward and a backward scan, they can output an auxiliary structure which can be used to infer a pixel label.

Samet and Tamminen [5] are the first researchers who clearly named the equivalence resolution problem as the *disjoint-set union problem*. This is an important achievement, since a quasi linear solution for this problem is available: the so called *union-find* algorithm, from the name of the basic operations involved. The algorithm is executed in two passes. The first pass creates an intermediate file consisting of image elements and equivalence classes while the second pass processes this file in reverse order, and assigns final labels to each image element. Their proposal is definitely complex, since it also targets quad-tree based image representations and is aimed at not keeping the equivalences in memory. In particular in [6] a general definition of this algorithm for arbitrary image representations has been proposed.

The Union-Find algorithm is the basis of a more modern approach for label resolution. As a new pixel is computed, the equivalence label is resolved: while the previous approaches generally performed first a collection of labels and at the end the resolution and the Union of equivalence classes, this new approach guarantees that at each pixel the structure is up to date.

A relevant paper in this evolution is [7] where Di Stefano and Bulgarelli proposed an online label resolution algorithm with an array-based structure to store the label equivalences. The array-based data structure has the advantage to reduce the memory required and to speed up the retrieval of elements without the use of pointer dereferencing. They do not explicitly name their equivalences resolution algorithm as Union-Find, and their solution requires multiple searches over the array at every Union operation.

In 2003, Suzuki [8] resumed Haralick's approach, including a small equivalence array and he provided a linear-time algorithm that in most cases requires 4 passes. The label resolution is performed exploiting array-based data structures, and each foreground pixel takes the minimum class of the neighboring foreground pixels classes. An important addition to this proposal is provided in an appendix in the form of a LUT of all possible neighborhoods, which allows to reduce computational times and costs by avoiding unnecessary Union operations.

In the same year, Chang *et al.* [9] proposed a radically different approach to connected components labeling. Their approach is an improvement of [10] and [11], and it is based on a single pass over the image exploiting contour tracing technique for internal and external contours, with a filling procedure for the internal pixels. This technique proved to be very fast, even because the filling is cache-friendly for images stored in a raster scan order, and the algorithm can also naturally output the connected components contours.

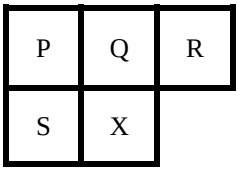
In 2005, Wu in [12] proposed a strategy to increase the performances of the Suzuki's approach. He exploited a decision tree to minimize the number of neighboring pixels to be visited in order to evaluate the label of the current pixel. In fact in a 8-connected components neighborhood, often only one pixel is needed to determine the label of the new one. In the same paper, Wu proposed another strategy to improve the Union-Find algorithm of Fiorio and Gustedt [13] exploiting an array-based data structure. For each equivalence array a path compression is performed to compute the root, in order to directly keep the minimum equivalent label within each equivalence array.

In 2007, He (in collaboration with Suzuki) proposed another fast approach in the form of a two scan algorithm [14]. The data structure used to manage the label resolution is implemented using three arrays in order to link the sets of equivalent classes without the use of pointers. By using this data structure, two algorithms have then been proposed: in [15] a run-based first scan is employed, while in [16] a decision tree is used to optimize the neighborhood exploration and to apply merging only when needed. The He and the Chang proposals can be considered the state-of-the-art methods for connected components labeling, being the latest evolution of two different approaches to solve the problem, and obtaining similar performances in terms of computation time. With the dataset used by He in [16], the Chang algorithm was shown to be slightly slower.

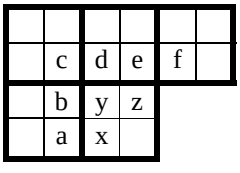
4 Speeding up neighbors computation

The algorithms analyzed so far differ each other on the way neighboring pixels are analyzed, how many passes are performed and in the way the resolution of equivalences is managed. While the number of passes depends on the underline idea of the algorithm, and the label resolution is based on a limited amount of data structures and optimization proposed in literature, there is still something to say about the neighborhood computation. In [16], besides the efficient data structure used for label resolution, He proposed an optimization of the neighborhood computation deeply minimizing the number of pixel needed to access.

In this paper, we provide another optimization for the neighboring computation based on a very straightforward observation: when using 8-connection, the pixels of a 2x2 square are all connected to each other. This implies that they will share the same label at the end of the computation. For this reason we propose to logically scan the image moving on a 2x2 pixel grid. To allow this, we have to provide rules for the connectivity of 2x2 blocks.



Blocks



Pixels in blocks

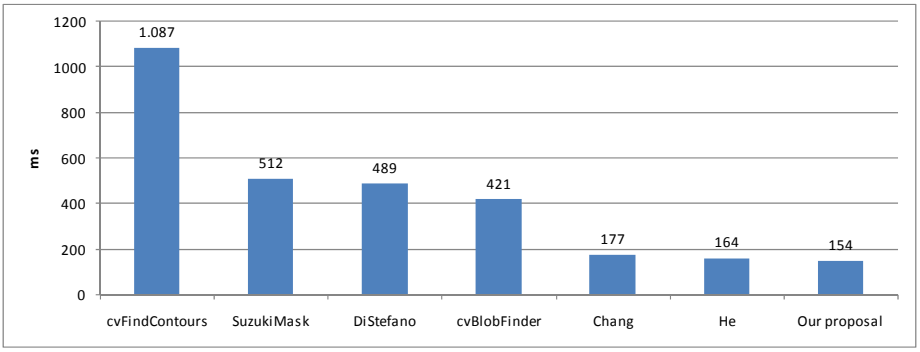


Fig. 3. Results of Test 1.

Referring to the figure, we can define the following rules:

- *P* is connected to *X* if *c* and *y* are foreground pixels
- *Q* is connected to *X* if (*d* or *e*) and (*y* or *z*) are foreground pixels
- *R* is connected to *X* if *f* and *z* are foreground pixels
- *S* is connected to *X* if (*a* or *b*) and (*y* or *x*) are foreground pixels

By applying these connectivity rules, we obtain two advantages: the first one is that the number of provisional labels created during the first scan, is roughly reduced by a factor of four, and the second is that we need to apply much less unions, since equivalences are implicitly solved within the blocks. Another advantage is that a single label is stored for the whole block. On the contrary the same pixel needs to be checked multiple times, but this is easily solved by the use of local variables and caching, and the second scan requires to access again the original image to check which pixels in the block require their label to be set. Overall the advantages greatly overcome the additional work required in the following stage.

This method may be applied to different connected component labeling algorithms, and, depending from the algorithms, can improve performances from 10% to 20% based on the way they consider the neighborhood of the current pixel.

5 Comparison

The main focus of this comparison is to evaluate the performance of the algorithms under stress, that is when working with high resolution images with thousands of labels. Besides, we also tested their scalability, varying the image sizes and the number of labels.

To this purpose, we produce three datasets coming from the binarized version of high resolution documentary images. The first dataset is composed by 615 images, with a resolution of 3840x2886 pixels. For each algorithm, a mean value of the processing times will indicate which one has the best overall performance. The second dataset is composed by 3,173 images derived by the first dataset by a sequence of 10 subsequent dilations with a 3x3 pixels square structuring element. In this way we preserve the image size (total amount of pixel processed) but we decrease the

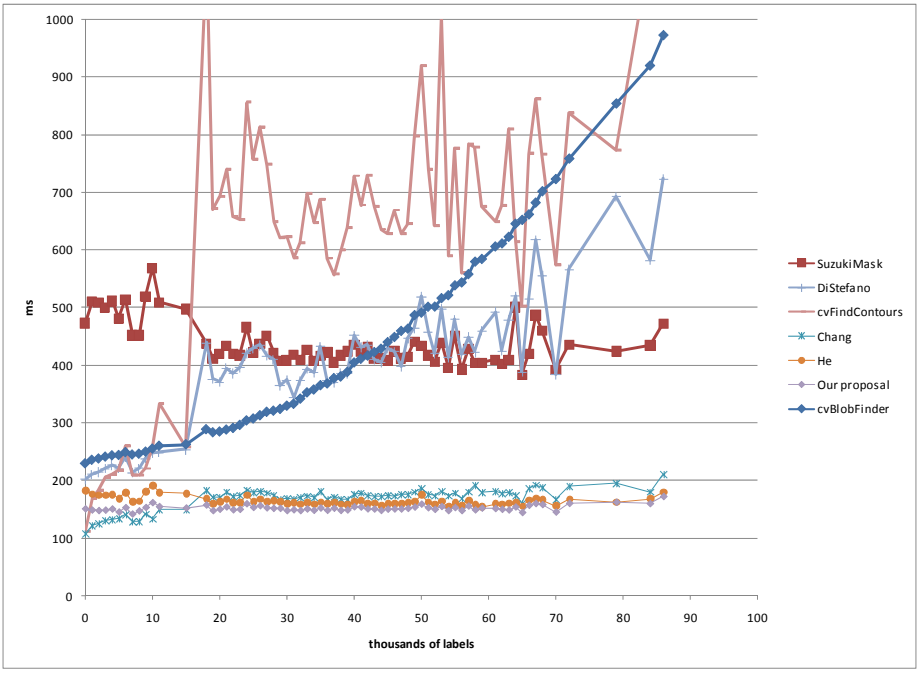


Fig. 4. Results of Test 2.

number of labels thanks to the dilations (that merges little by little an increasing amount of blobs). This test will show which algorithm has the best scalability varying the number of labels, and at the same time which algorithm performs better with lower and higher amount of labels. Finally the third dataset is composed by 3807 images obtained from a 160x120 downscaled version of the first dataset, increasingly upscaled with 11 4:3 formats (up to the original image size). This dataset will be useful to evaluate the scalability of these algorithms with a small fixed number of blobs, but a larger number of pixels.

In all tests we applied our 2x2 block optimization to a raster scan algorithm which uses He's technique for handling equivalences and applies a *union* operation every time two different labeled blocks are connected.

The results of the first test are shown in Fig. 3. On high resolution images, our approach provides the best performances, by using the block optimization. Suzuki and DiStefano proposals are superior to the OpenCV standard contour tracing method, but cannot beat the other technique based on flood fill. After our proposal, the overall best techniques are Chang's and He's.

The results of the second test are shown in Fig. 4. Even in this case, the performace of OpenCV algorithms, as well as Suzuki and DiStefano ones, result to be not quite good (even if their proposal proves to have a good scalability). Chang's and He's algorithms and our proposal are still the approaches with the best performances. It is important to highlight that the behaviour of these algorithms is somehow different below and above the 150 labels: in this case the OpenCV contour tracing technique

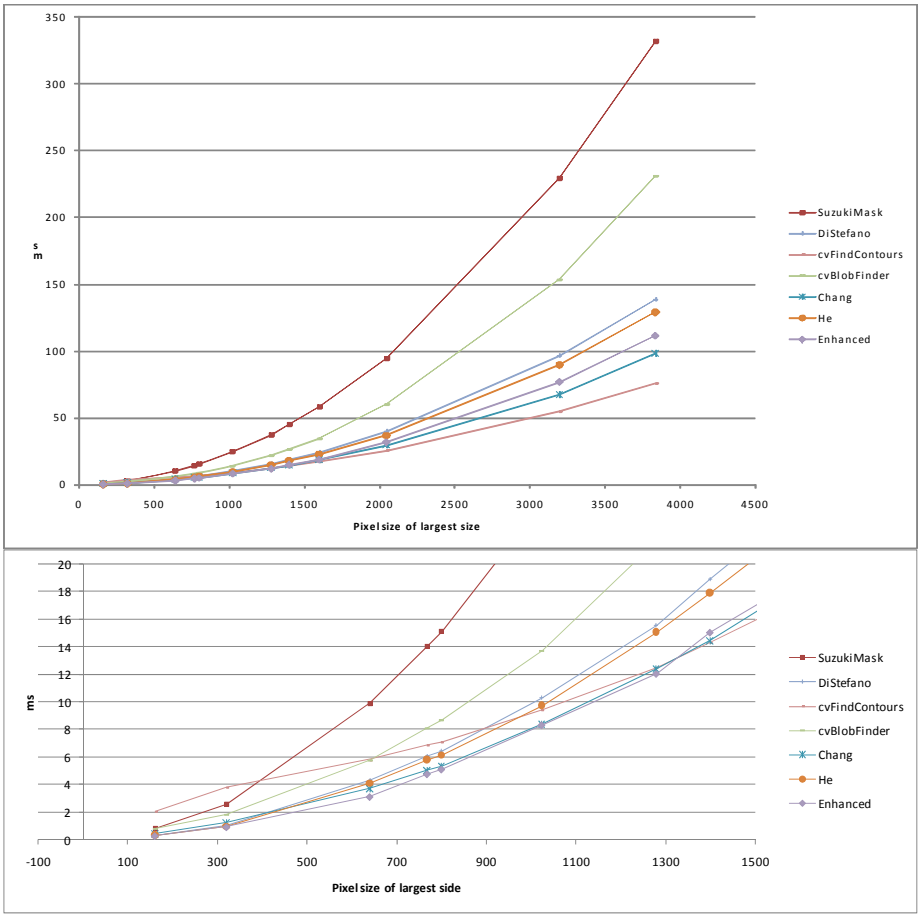


Fig. 5. Results of Test 3. The lower figure shows an enlargement of the area of images with widths below 1500 pixels.

stays close to the other techniques. Chang algorithm is a clear winner with less then 10000 labels, while our proposal has the best performance in other cases.

Finally the results of the latest test are shown in Fig. 5. In this case, where the average number of labels is 473, OpenCV contour tracing proved to have a great scalability increasing the size of the image, while Chang and our method still perform very well. Nevertheless, zooming in at lower images sizes, up to 1024x768, we can notice that our approach and Chang's provide the best performance.

6 Conclusions

We have given a comprehensive overview of the different strategies which have been proposed for the connected component labeling problem, pointing out relations and evolution of the single optimization proposals.

A new strategy for label propagation has been proposed, based on a 2x2 block subdivision. This strategy allows to improve the performance of many existing algorithms, given that the specific connection rules are satisfied.

Experimental results have stressed a few points of the different algorithms, in particular showing how the Cheng approach is a clear winner when the number of labels is small, compared to the image size, while our proposal can obtain around 10% of speedup when the number of labels is high.

References

1. Rosenfeld, A., Pfaltz, J.L.: Sequential Operations in Digital Picture Processing. Journal of the ACM, vol. 13, n. 4, pp. 471--494 (1966)
2. Haralick, R.M.: Some neighborhood operations. Real Time Parallel Computing: Image Analysis, Plenum Press, New York, pp. 11--35 (1981)
3. Lumia, R., Shapiro, L.G., and Zuniga, O.A.: A New Connected Components Algorithm for Virtual Memory Computers. Computer Vision Graphics and Image Processing, vol. 22, n. 2, pp. 287--300 (1983)
4. Schwartz, J.T., Sharjr, M., Siegel, A.: An efficient algorithm for finding connected components in a binary image. Robotics Research Technical Report 38. New York Univ. New York (1985)
5. Samet, H., Tamminen, M.: An Improved Approach to connected component labeling of images. In: International Conference on Computer Vision And Pattern Recognition, pp. 312--318 (1986)
6. Dillencourt, M.B., Samet, H., Tamminen, M.: A general approach to connected-component labeling for arbitrary image representations. Journal of the ACM, vol. 39, n. 2, pp. 253--280 (1992)
7. Di Stefano, L., Bulgarelli, A.: A simple and efficient connected components labeling algorithm. In: 10th International Conference on Image Analysis and Processing, pp. 322--327 (1999)
8. Suzuki, K., Horiba, I., Sugie, N.: Linear-time connected-component labeling based on sequential local operations. Comput. Vis. Image Underst., vol. 89, n. 1, pp. 1--23 (2003)
9. Chang, F., Chen C.J.,: A component-labeling algorithm using contour tracing technique. In 7th International Conference on Document Analysis and Recognition, pp. 741--745, (2003)
10. Freeman, H.: Techniques for the Digital Computer Analysis of Chain-Encoded Arbitrary Plane Curves. In: 17th National Electronics Conference, pp. 412--432. (1961)
11. Pavlidis, T.: Algorithms for graphics and image processing. Computer Science Press, Rockville MD (1982)
12. Wu, K., Otoo, E., Shoshani, A.: Optimizing connected component labeling algorithms. In: SPIE Conference on Medical Imaging, vol. 5747, pp. 1965--1976 (2005)
13. Fiorio, C., Gustedt, J.: Two Linear Time Union-Find Strategies for Image Processing. Theor. Comput. Sci. 154, pp. 165--181 (1996)
14. He, L., Chao, Y., Suzuki, K.: A Linear-Time Two-Scan Labeling Algorithm. In: IEEE International Conference on Image Processing, vol. 5, pp. 241--244, (2007)
15. He, L., Chao, Y., Suzuki, K.: A Run-Based Two-Scan Labeling Algorithm. IEEE Transactions on Image Processing, vol. 17, n. 5, pp. 749--756 (2008)
16. He, L., Chao, T., Suzuki, K., Wu, K.: Fast connected-component labeling. Pattern Recognition, In Press (2008)