FAST BLOCK BASED CONNECTED COMPONENTS LABELING

Costantino Grana, Daniele Borghesani, Rita Cucchiara

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Modena e Reggio Emilia, Italy

ABSTRACT

In this paper we present a new optimization technique for the neighborhood computation in connected component labeling focused on images stored in raster scan order. This new technique is based on a 2x2 square block analysis of the image, and it exploits the fact that, 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. To prove the effectiveness of our proposal, we show a comprehensive comparison of the most used and advanced connected components labeling techniques presented so far. 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.

Index Terms— connected component labeling, comparison, union-find

1. INTRODUCTION

Labeling is a fundamental task in several computer vision application. It is used as basic algorithm to assign 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. In this paper, a new block optimization technique is proposed to speed up the neighborhood computation, and the significant performance improvement is proved through a comparison with the most effective labeling techniques proposed so far. Notice that we chose to include within the comparison also the connected components labeling approaches implemented in the OpenCV, the very popular open source project started by Intel containing the majority of computer vision algorithms. In particular, two strategies for connected component analysis are included: 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.

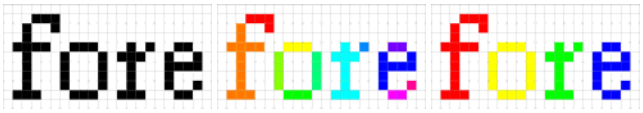
We will exclude two wide classes of algorithms from our analysis. The first one is the class of parallel algorithms which has been extensively studied up to the first half of the '90s. These algorithms were aimed to specific massively parallel architectures and do not readily apply to current common workstations, which provide more and more parallelism, but substantially differ from the target of those algorithms. The second class is given by algorithms suitable for hierarchical image representations (for example quadtrees). 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, a brief overview of the most used labeling algorithms is presented, and then we detail our proposal. The different algorithms performance (in terms of execution time, since every algorithm provides exactly the same result) 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. RELATED WORK

The problem of labeling has been deeply studied since the beginning of Computer Vision science. We can identify three main strategies to accomplish this task.

The most common is the two-scan approach. The classic approach to connected components labeling belongs to this category, and it was proposed by Rosenfeld *et al.* in 1966 [1]. It is based on a raster scan of the image and the "redundancies" of the labels are stored in an equivalences table with all the neighborhood references to be solved by repeatedly processing it. Lumia *et al.* [2] proposed a way to keep the equivalence table as small as possible to be more effective with '83 virtual memory computers, solving the equivalences at the end of each row. In 1986 Samet and Tamminen [3] 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. Quickly the Union-Find algorithm became the basis of the most modern approaches for label resolution, as well as the adoption of more efficient array-based data structures to store equivalences. Optimizations over this approach have been proposed by Di Stefano and Bulgarelli [4] that suggested an online label resolution algorithm but requiring multiple searches over the array at every Union operation. In [5] Wu proposed a strategy which exploited an array-based representation of the disjoint set data structure which also included path compression. He *et al.* proposed a more efficient



a) *b)* *c)*

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

data structure to manage the label resolution, implemented in the form of three arrays in order to link the sets of equivalent classes without dealing with lists and pointers [6]. By using this data structure, two algorithms have then been proposed: in [7] a run-based first scan is employed, while in [8] a decision tree is used to optimize the neighborhood exploration and to apply merging only when needed.

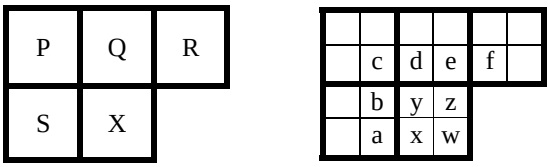
A second category is composed by algorithms that perform a multipass strategy upon the image. The archetype of this approach was proposed by Haralick *et al.* [9]. This algorithm does not use any equivalences table and no extra space: it performs iteratively forward and backward raster scan passes over the output image to solve the equivalences exploiting only local neighborhood information. In 2003, Suzuki [10] exploited a similar approach but including a small equivalence array. He provided a linear-time algorithm that in most cases requires no more than 4 passes. The label resolution uses 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. Finally a decision tree has been proposed by Wu in [5] to improve Suzuki's approach in order to minimize neighborhood computation.

The last category is composed by all the techniques exploiting a contour tracing procedure. In particular a very fast approach is proposed by Chang *et al.* [11] 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.

3. 2X2 BLOCK-BASED NEIGHBORS COMPUTATION

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.



Blocks Pixels in blocks

Fig. 2. Block arrangement and pixel naming convention in our proposal.

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.

The algorithms analyzed so far differ each other based 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 [8], besides the efficient data structure used for label resolution, He proposed an optimization of the neighborhood computation deeply minimizing the number of pixel which needed to be accessed.

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.

Referring to Fig. 2, we can define the following rules to define blocks connectivity:

1. P is connected to X if c and y are foreground pixels
2. Q is connected to X if (d or e) and (y or z) are foreground pixels
3. R is connected to X if f and z are foreground pixels
4. S is connected to X if (a or b) and (y or x) are foreground pixels

Given these rules, it is possible to reduce the number of accesses to the current block's pixels, by checking the connectivity giving priority to the most used pixels. In particular if pixel y is a foreground pixel, we can avoid to check x and z in rules 1, 2 and 4 (z shall be checked for rule 3). If pixel y is a background pixel we can avoid any check on c and if z is a foreground pixel we need to check rules 2 and 3, then go to x in rule 4. If also z is a background pixel, only rule 4 needs checking, so x first and then a or b. Finally w should be checked to see if it is background. If all pixels are background, the block is skipped, otherwise if all rules say the block is isolated we create a new label. If any of the rules provides connection, we need to find the minimum and perform union operations between different classes. A pseudo code version of the algorithm is given in Fig. 3.

```
if (y==F) {
   if (c==F) lp = P.GetLabel();
   if (d==F || e==F) lq = Q.GetLabel();
   if (a==F || b==F) ls = S.GetLabel();
   if (z==F) {
      if (f==F) lr = R.GetLabel();
   }
}
else if (z==F) {
   if (d==F || e==F) lq = Q.GetLabel();
   if (f==F) lr = R.GetLabel();
   if (x==F) {
      if (a==F || b==F) ls = S.GetLabel();
   }
}
else if (x==F) {
   if (a==F || b==F) ls = S.GetLabel();
}
else if (w!=F) {
   continue; // this block is empty
}
// solve equivalences or create a new label
```

Fig. 4. C++ style code of the connectivity check for a 2x2 block. The collected labels are then used to merge classes if needed.

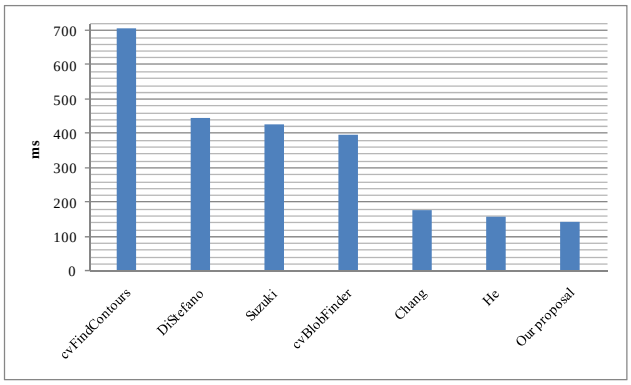
By applying these connectivity rules, we obtain two advantages: the first one is that the number of provisional labels created during the first scan, is roughly reduced by a factor of four, and the second is that we need to apply much less unions, since classes equivalence is implicitly solved within 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.

4. COMPARISON

The majority of algorithms presented so far has been tested with relatively small images with few labels. In this paper, we mainly aimed at evaluating the performance of these algorithm with high resolution images with thousands of labels. Then we tested their scalability, in order to evaluate what is their behavior varying the image sizes and the number of labels.

We used a large dataset composed by the Otsu-binarized versions of 615 images of high resolution (3840x2886) illuminated manuscripts pages, with gothic text, pictures and great floral decorations. This dataset gives us the possibility to test the connected components labeling capabilities with a lot of complex patterns at different sizes, turning out to be very stressful. For each algorithm, a mean value of the processing times will indicate which one has the best overall performance. From the original dataset, we derived a second dataset composed of 3,173 images obtained by 10 subsequent dilations. In this way we preserve the image size (total amount of pixel processed) but we decrease the 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.



The tests upon the first dataset in Fig. 4 show that our approach based on the proposed block optimization provides the best performance, especially on high resolution images with a lot of labels (that is the focus of this paper). Suzuki and DiStefano

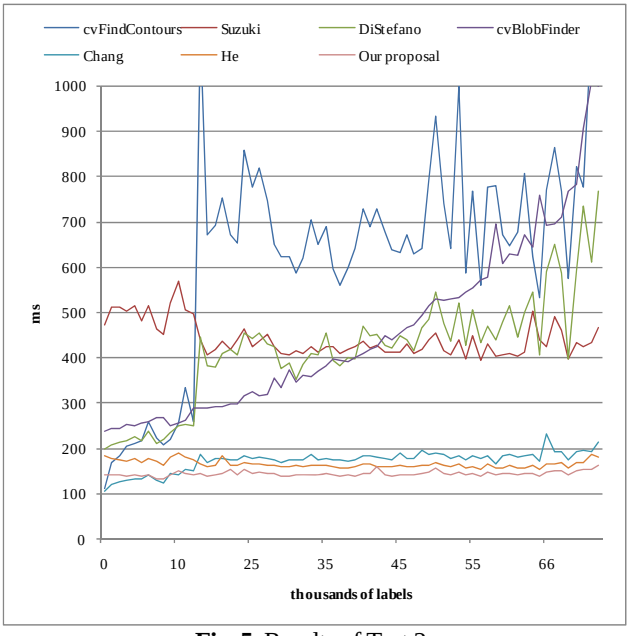
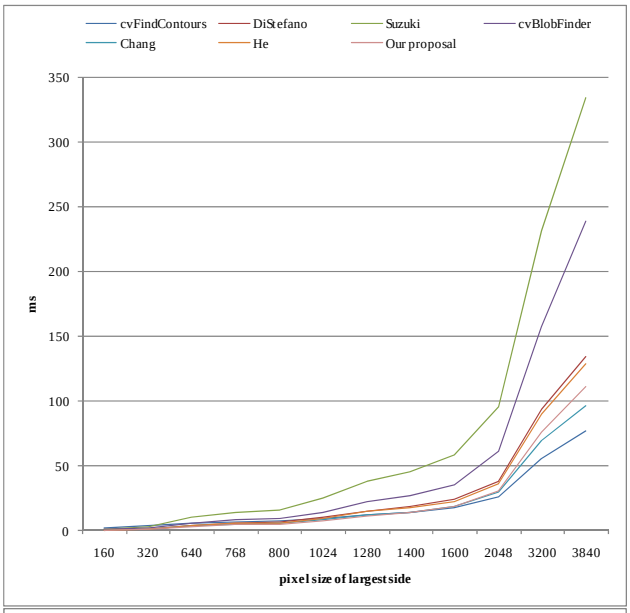
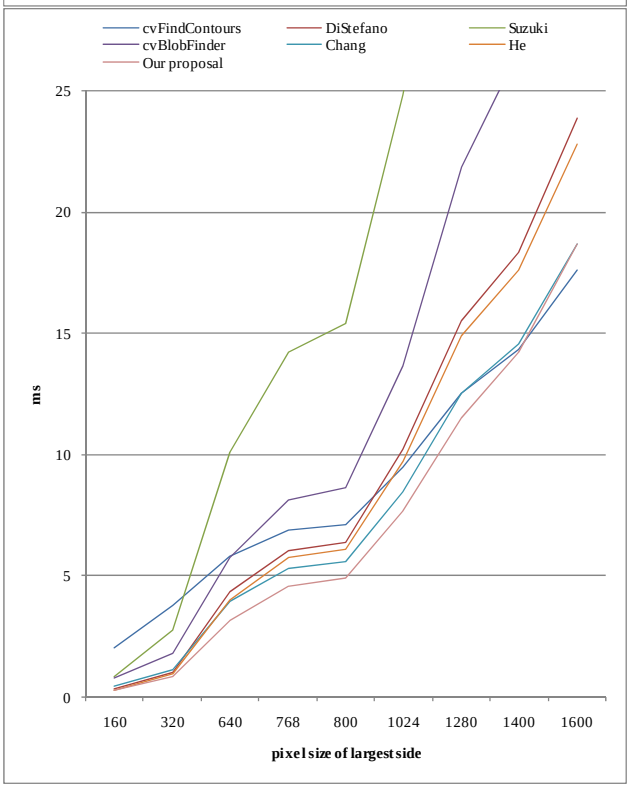


Fig. 5. Results of Test 2.





proposals are superior to the OpenCV standard contour tracing method, but cannot beat the other technique based on flood fill. The quality of Chang's and He's proposal (that can be considered the state of the art approaches so far) is demostrated by their results, that are close to our improved proposal.

Looking at the scalability of these algorithm varying the number of labels (Fig. 5), we can see that only Chang's and He's algorithms show a progression similar to our proposal, while the OpenCV algorithm present the worst progression. On the contrary, considering a lower number of labels (namely less than 150), the OpenCV contour tracing technique presents very good performance, aligning to the best techniques.

Finally looking at the scalability of these algorithms varying the size of the image (Fig. 6), we can see that, with an average number of labels is 473, OpenCV contour tracing presents the overall best progression. Nevertheless, zooming in at lower images sizes, up to 1024x768, we can notice that our approach and Chang's provide the best performance.

5. CONCLUSIONS

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.

6. 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. 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)
3. 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)
4. 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)
5. Wu, K., Otoo, E., Shoshani, A.: Optimizing connected component labeling algorithms. In: SPIE Conference on Medical Imaging, vol. 5747, pp. 1965--1976 (2005)
6. 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)
7. 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)
8. He, L., Chao, T., Suzuki, K., Wu, K.: Fast connected-component labeling. Pattern Recognition, In Press (2008)
9. Haralick, R.M.: Some neighborhood operations. Real Time Parallel Computing: Image Analysis, Plenum Press, New York, pp. 11--35 (1981)
10. 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)
11. 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)