A Single-Scan Algorithm for Connected Components Labelling in a Traffic Monitoring Application

Alessandro Bevilacqua, Alessandro Lanza,
Giorgio Baccarani, and Riccardo Rovatti

ARCES - Centre of Excellence,
DEIS (Department of Electronics, Computer Science and Systems)
University of Bologna
Via Toffano, 2
40125 Bologna, ITALY
{abevilacqua, alanza, gbaccarani, rrovatti}@deis.unibo.it
WWW home page: http://www.arces.unibo.it/wp/home.htm

Abstract. This paper presents a fast algorithm based on sequential local operations which aims at labelling connected components in binary images. While classical algorithms scan the image twice and utilize an equivalence table to store and resolve label redundancies, our method performs just a single scan, relying on the idea of labelling a whole blob at a time. In this way, we avoid label redundancies. As a consequence, the use of both equivalence tables and algorithms to resolve them becomes unnecessary. This leads our labelling algorithm to attain even more significant performances in the case of images characterized by blobs generating a large number of label equivalences. The proposed labelling algorithm has been successfully utilized in our visual surveillance system.

1 Introduction

The final goal of our research is to develop an effective visual surveillance application for traffic monitoring purposes. The target sequences are outdoor grey level sequences taken by a single stationary camera. The application should work at a high frame utilizing as simple as possible image processing operations.

Our traffic monitoring system, as well as all visual surveillance applications, mostly relies on its first processing step which consists in detecting moving objects. At the end of this phase it is necessary to identify each moving object, or *blob*, assigning a unique label to all the pixels belonging to it.

In this paper we describe a fast single-scan labelling algorithm based on sequential local operations. The classical labelling methods perform two forward raster-scans of the image. The first scan assigns a temporary label to each foreground pixel, based on the labels of its already visited neighbours, and registers label equivalences. Hence, label equivalences are *resolved*, that is are processed to

J. Bigun and T. Gustavsson (Eds.): SCIA 2003, LNCS 2749, pp. 677–684, 2003.

determine equivalence classes. Finally, the second scan replaces each provisional label with the identifier of its corresponding equivalence class. Our algorithm performs just a single scan of the image, labelling a whole blob at a time, thus avoiding the handling of label redundancies. Therefore, the efficiency of our method becomes independent of the number of label conflicts. We have realized that images characterized by blobs with irregular borders can generate many label equivalences. In this case the performances of our algorithm becomes even more significant, letting it be preferred to the classical ones.

This paper is organized as follows. In the next section the overall visual surveillance system is briefly illustrated. In Sect. 3 an introduction to the connected components labelling theory is presented. Section 4 presents an overview of different labelling algorithms. Section 5 illustrates our single-scan labelling algorithm and Sect. 6 draws conclusions.

2 The Motion Detection System

The framework of our motion detection algorithm is described in Fig. 1. Since

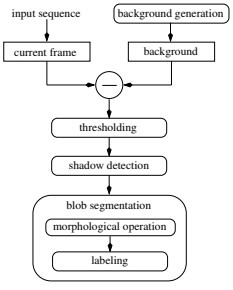

Fig. 1. The framework of our motion detection algorithm

the purpose of our Smart Surveillance System [1] (or briefly, *S-CUBE*) consists in traffic monitoring, hence the starting point is the extraction of vehicles and pedestrians from the background. The system relies on a stationary video camera and includes a background generation and updating algorithm [2] and a shadow detection module [3]. The input of the system is a grey level sequence representing a daytime traffic scene in which blobs are made of vehicles, humans, shadows or all of them. The algorithm processes one frame at a time and returns the segmented blobs as the final output.

After that the system has generated a background through a bootstrap procedure and has performed the arithmetic subtraction between the generated background (Fig. 2, on the left) and the current frame (Fig. 2, in the middle), a suitable threshold T has to be chosen and applied in order to detect moving pixels. The output of this step, called *background subtraction*, consists of a noisy

Fig. 2. The generated background (left), a sample frame (middle) and the thresholded background subtraction (right)

binary image (Fig. 2, on the right) which retains most of the true moving pixels together with false signals due to noise, moving shadows and uninteresting moving objects, such as hedges or trees. Hence, the subsequent shadow detection module removes pixels belonging to moving shadows from the binary image (Fig. 3, on the left). Then, morphological operations [4] perform both denoising

Fig. 3. The binary image after shadow removal (left), after morphological operations (middle) and the labelled image (right)

and pixel grouping (Fig. 3, in the middle). At last, each connected component is identified with a unique label (Fig. 3, on the right) by means of the labelling method presented in this paper.

Currently, the first step of our segmentation module can generate blobs characterized by jagged borders, when trying to achieve a high resolution. We have realized that the classical labelling algorithms are not efficient with this kind of blobs because they have to handle a large number of label equivalences. The algorithm we have designed could guarantee a high efficiency even in the presence of this unfavourable situation.

3 Connected Components Labelling

Two pixels, P and Q, belonging to an image I are said to be *connected* if there exists a sequence of pixels $P_0 = P, P_1, \ldots, P_{n-1}, P_n = Q$ such that P_i is a neighbour of P_{i-1}, $\forall\ 1 \leq i \leq n$. Therefore, the definition of pixel connection relies on that of pixel neighbourhood (Fig. 4): 4-neighbourhood leads to 4-connection, 8-neighbourhood leads to 8-connection. A subset of an image is said to be *con-*

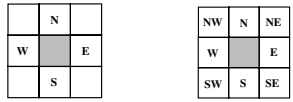

Fig. 4. Neighbouroods: 4-neighbourood (left) and 8-neighbourood (right)

nected if any two points of it are connected. Now, let I be a binary image and F and B the complementary subsets of I corresponding respectively to foreground and background pixels. The *connected components* of I are the maximal size connected subsets of F. It is possible to give a different definition of connected components for binary images on the basis of the concept of relation. If we consider I as a set of pixels, the relation among them *to be connected to* is evidently an equivalence relation (reflexive, symmetric and transitive). The connected components of I are the equivalence classes of F. In practice, given a binary image, the labelling algorithm consists in generating a new image in which a unique label is assigned to all the pixels belonging to the same connected component and different labels are associated with different components (Fig. 5).

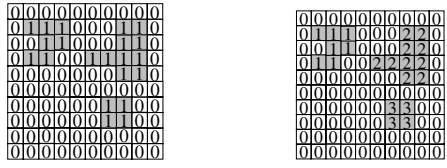

Fig. 5. A toy example of the labelling task: input (left) and output (right) images

As for background pixels, they are typically left unchanged by the labelling process, though in principle it is possible to mark them with a new label value. Hence, assuming that I has n connected components, in the labelled image we find $n+1$ labels, one of them, l_B, assigned to the pixels in B and the remaining n used to mark the pixels in F as belonging to a distinct connected component. It is worth noticing that the meaning of the relation among pixels expressed through having the same label value in the labelled image depends on the label value: two pixels marked with l_B belong to B, but they are not necessarily connected, two pixels marked with $l_i \neq l_b$ belong to F and are connected.

4 Previous Works

The classical sequential labelling algorithm dates back to the early days of computer vision ([5, 6]) and relies on two subsequent forward raster-scans of the image. During the first scan, provisional labels are assigned to foreground pixels based on the values of their already visited neighbours. If a foreground pixel

with two foreground neighbours carrying different labels is found, these labels are stored into an equivalence table. At the end of the first scan label equivalences registered in the equivalence table are resolved to determine equivalence classes. During the second scan each temporary label is replaced by the label assigned to its equivalence class. The way in which label equivalences are resolved can have a dramatic effect upon the execution time of this algorithm. Modifications to this algorithm include one proposed in [7], where label equivalences are processed during the first raster-scan, precisely at the end of each row. Haralick and Shapiro in [8] handle equivalences by means of a graph structure where the nodes are the labels found in the first scan and the edges connect pairs of equivalent labels. Equivalence classes can be seen as the connected components of the graph, and connected components can be determined by a standard depth-first search algorithm. Gonzales and Woods in [9] propose to use a boolean $n \times n$ matrix, B, to represent the equivalence relation between labels (n is the number of labels, $B(i,j) = 1$ means that labels i and j are equivalent). During the first scan B is filled up, then label equivalences are resolved by means of general and formal tools for handling equivalence relations. Klette and Zamperoni in [10] use an equivalence table composed by 2-tuples, each storing a pair of equivalent labels. As well as in [8], authors in [11] represent the label equivalence relation with a graph structure, but suggest to implement each equivalence class with a cyclic graph. Di Stefano and Bulgarelli in [12] describe an algorithm where label equivalences are processed during the first pass in order to determine the correct state of equivalence classes at each time of the scan. This approach allows the check for a conflict to be carried out on class identifiers rather than on labels. In [13], authors choose a "divide and conquer" approach, partitioning the input binary image into $N \times N$ rectangles. They perform local equivalences resolution on each subset, keeping track of the global equivalences with list pointers to equivalence lists. Local equivalences are resolved using the same method proposed in [9]. Finally, the algorithm developed in [14] utilizes a one-dimensional table, called by the authors the *label connection table*, to store label equivalences. The algorithm repeats passes through the image in forward and backward raster directions alternately, using the label connection table, until no provisional labels change.

5 A Single-Scan Labelling Algorithm

5.1 The Algorithm

The basic idea which originated our algorithm has been prompted by the outcome of the intermediate steps of S-CUBE. The key concept is to avoid dealing with the problem of label equivalences resolution. As a matter of fact, whatever method is employed to obtain equivalence classes its efficiency strictly depends on the number of redundant labels and consequently on the shape of the connected components to be labelled. Blobs characterized by irregular borders, like the ones occasionally generated by our segmentation module, can produce a large number of equivalences. Therefore this kind of blobs have a negative effect upon

the running time of the classic labelling methods. In order to avoid assigning redundant labels, our algorithm labels an entire blob at a time, thus performing just a single forward raster-scan of the image. Let I be the input binary image to be labelled (Fig. 6, A), O the output labelled image (Fig. 6, D), F and B the

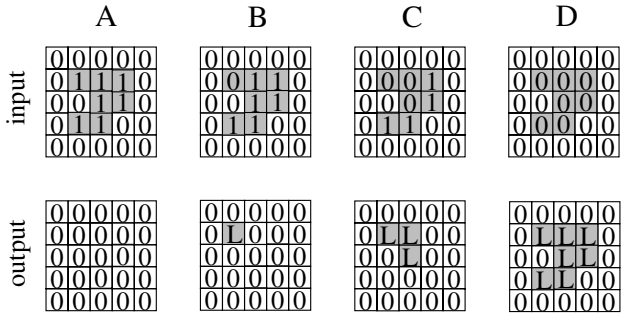

Fig. 6. The steps of the single-scan algorithm on a toy image

complementary subsets of I corresponding respectively to foreground and background pixels, x the current pixel to be labelled and L the label assigned to the current blob. Figure 6 represents the steps of the proposed single-scan labelling algorithm performed on a toy input binary image. In Fig. 7 we briefly outline the kernel of the algorithm, in a pseudo-code. In the proposed algorithm the image

```
1.   for each pixel x ∈ I do
2.       if (x ∈ F) then
3.           v_start ← 0;  v_end ← 1;  v_tmp ← 0
4.           store_coordinates(x, V)
5.           L ← L + 1
6.           while (v_end > v_start) do
7.               v_tmp ← 0
8.               for i from v_start to (v_end − 1) do
9.                   label(x, O, L)
10.                  erase(x, I)
11.                  check_neighbours(x, I)
12.              v_start ← v_end;  v_end ← v_end + v_tmp
```

Fig. 7. The kernel of the proposed algorithm

has been sorted lexicographically and we utilize a one-dimensional array $V[\cdot]$ to store the coordinates of all the pixels belonging to the current blob. The main loop of the algorithm (Fig. 7) performs a single forward raster-scan of the input image I. When the first pixel of a blob is found (line 2), its coordinates are stored in $V[\cdot]$ (line 4), and the label counter L is increased (line 5). Here, the *explosion*

loops relative to the current blob start (line 6). For each pixel whose coordinates are stored in $V[\cdot]$ between *v_start* and *v_end* -1 (line 8) the algorithm firstly assigns the current label value L to the corresponding pixel in the output image (line 9), secondly erases the pixel in the input image making it a background pixel (line 10) and finally checks in the input image the neighbours of the pixel belonging to F (line 11), storing in $V[\cdot]$ their coordinates. This last operation is performed by the *check_neighbours* function, outlined in Fig. 8. The explosion

```
1.    check_neighbours(x, I)
2.    BEGIN
3.        for each y neighbour of x do
4.            if (y ∈ F) then
5.                store_coordinates(y, V)
6.    END
```

Fig. 8. The *check_neighbours* function

loops end when all the pixels of the current blob have been found and labelled. Hence, the forward raster-scan of the input image resumes from the pixel where it stopped, thus labelling the entire image.

5.2 Performance in Terms of Computational Complexity

As regards the asymptotic computational complexity, the task of labelling an input binary image of N pixels is evidently an $\Omega(N)$ problem. As a matter of fact, it results necessary to examine all the pixels of the input image in order to assign them a label. It turns out easily that the proposed algorithm's worst case complexity is $O(N)$. In fact, independently of the possible occurrences of the starting data, the algorithm executes a fixed number of instructions for each foreground pixel of the input image, whether it is the first pixel belonging to a new blob (Fig. 7, lines 3–5) or it is any of the other foreground pixels (Fig. 7, lines 9–11). As a consequence, the algorithm is optimal.

As far as the complexity of the classic methods is concerned, one must distinguish between best case and worst case complexity. The best case occurs when input images contain blobs showing particular shapes and regular borders. On the contrary, the worst case occurs when either blobs show generic shapes or jagged borders. As a matter of fact, the classic algorithms have an asymptotical complexity of $O(N)$ as well, sometimes resulting to be more efficient than the proposed algorithm due to the lower number of performed instructions. As for the worst case condition, it has been shown that the classic algorithms' complexity goes worse than $O(N)$ (e.g. $O(N^3)$ in [5] and $O(N^2)$ in [12]). As regards our system, blobs often show jagged borders when the algorithm tries to achieve a high resolution. Such images are yield the worst case condition as far as the classic methods are concerned. With such images, the proposed algorithm results to perform more efficiently than the classic ones.

6 Conclusions and Future Works

In this paper we have described a single-scan labelling algorithm based on sequential local operations. Our approach is very different from the classical two-scan labelling algorithms, whose performances are strictly dependent on the number of label equivalences generated. The strength of the proposed algorithm relies on its efficiency, worthy in case of blobs with jagged borders. Besides, our algorithm results to be easily implemented in a recursive manner. As far as future works are concerned, the recursive implementation as well as the use of the size-thresholding operators is being developed.

References

1. Bevilacqua, A., Roffilli, M.: Robust denoising and moving shadows detection in traffic scenes. In: Proceedings of the Tech. Sketches of the 25^{th} IEEE Conference on Computer Vision and Pattern Recognition (CVPR01). (2001) 1–4
2. Bevilacqua, A.: A novel background initialization method in visual surveillance. In: IAPR Workshop on Machine Vision Applications (MVA02). (2002) 614–617
3. Bevilacqua, A.: Effective shadow detection in traffic monitoring applications. Journal of WSCG **11** (2003) 57–64
4. Bevilacqua, A.: Effective object segmentation in a traffic monitoring application. In: Proceedings of the 3^{rd} Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP02). (2002) 125–130
5. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Journal of the ACM **13** (1966) 471–494
6. Rosenfeld, A., Kak, A.C.: Digital Picture Processing. Volume 2. Academic Press (1982)
7. Hattori, T.: A high-speed pipeline processor for regional labeling based on a new algorithm. In: Proceedings of the 10^{th} International Conference on Pattern Recognition (ICPR90). (1990) 494–496
8. Haralick, R.M., Shapiro, G.L.: Computer and Robot Vision. Volume 1. Addison-Wesley (1992)
9. Gonzales, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley (1992)
10. Klette, R., Zamperoni, P.: Handbook of Image Processing Operators. (1996)
11. Lacassagne, L., Milgram, M., Garda, P.: Motion detection, labeling, data association and tracking, in real-time on RISC computer. In: Proceedings of the 10^{th} International Conference on Image Analysis and Processing (ICIAP99). (1999) 520–525
12. Stefano, L.D., Bulgarelli, A.: A simple and efficient connected components labeling algorithm. In: Proceedings of the 10^{th} International Conference on Image Analysis and Processing (ICIAP99). (1999) 322–327
13. Park, J.M., Looney, G.C., Chen, H.C.: Fast connected-component labeling algorithm using a divide and conquer technique. In: Proceedings of the 15^{th} International Conference on Computers and their Applications (CATA00). (2000) 373–376
14. Suzuki, K., Horiba, I., Sugie, N.: Fast connected-component labeling based on sequential local operations in the course of forward raster scan followed by backward raster scan. In: Proceedings of the 15^{th} International Conference on Pattern Recognition (ICPR00). Volume 2. (2000) 434–437