A Fast and Precise Blob Detection

빈흐타한
숭실대학교

Thanh Binh Nguyen, Sun-Tae Chung
School of Electronics Engineering, Soongsil University

Abstract

Blob detection is an essential ingredient process in some computer applications such as intelligent visual surveillance. However, previous blob detection algorithms are still computationally heavy so that supporting real-time multi-channel intelligent visual surveillance in a workstation or even one-channel real-time visual surveillance in a embedded system using them turns out prohibitively difficult. In this paper, we propose a fast and precise blob detection algorithm for visual surveillance. Blob detection in visual surveillance goes through several processing steps: foreground mask extraction, foreground mask correction, and connected component labeling. Foreground mask correction necessary for a precise detection is usually accomplished using morphological operations like opening and closing. Morphological operations are computationally expensive and moreover, they are difficult to run in parallel with connected component labeling routine since they need much different processing from what connected component labeling does. In this paper, we first develop a fast and precise foreground mask correction method utilizing on neighbor pixel checking which is also employed in connected component labeling so that the developed foreground mask correction method can be incorporated into connected component labeling routine. Through experiments, it is verified that our proposed blob detection algorithm based on the foreground mask correction method developed in this paper shows better processing speed and more precise blob detection.

Ⅰ. Introduction

Blob detection is the foremost important process in various computer vision applications such as intelligent visual surveillance. It is now well known that the performance of intelligent visual surveillance heavily depends on how precisely and fastly interesting objects are detected [1].

Blob detection is usually processed through several steps: foreground mask extraction, foreground mask correction, and blob segmentation through connected component labeling. Foreground mask correction has usually been accomplished using morphological operations like opening and closing as a preprocessing step [3,4,5]. However, morphological operations are computational expensive. Moreover, they are difficult to be computed in parallel with connected component labeling procedure since they need much different processing from what connected component labeling does.

In this paper, we propose a fast and precise blob detection algorithm for visual surveillance. We first develop a blob correction method which can be efficiently combined with the connected component labeling procedure. NFPP(Neighbor

Foreground Pixel Propagation) is designed, utilizes 8-neighbors checking which is also employed in connected component labeling procedure (CCL) so that NFPP can be incorporated into CCL processing routine. Then, foreground mask correction is accomplished while CCL is processing, which can save the processing time much more compared with a conventional blob correction method using morphology operations, which needs to be processed separately before CCL processing. Through the experiments, it is shown that the proposed blob detection algorithm based on NFPP performs better than the conventional blob detection algorithm with respect to the processing time and the preciseness in blob detection.

The rest of the paper is organized as follows. Section 2 introduces the basic technical background about blob detection, morphology and connected component labeling. Section 3 describes our proposed blob detection algorithm. There, we first explains our developed blob correction method, NFPP which is essential in achieving faster and more precise than blob detection processing. Experiment results are discussed in Section 4, and finally the conclusion is presented in Section 5.

Ⅱ. Backgrounds

2.1 Blob Detection in Motion Analysis

Blob detection in intelligent visual surveillance usually go through several steps that are shown in Fig. 1.

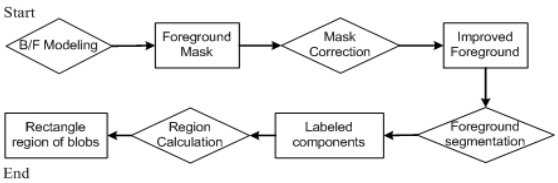


Fig. 1. General work flow of motion detection.

Fig. 2 shows example images related with blob detection steps.

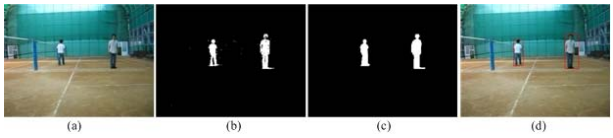


Fig. 2. Original frame image (a), foreground binary image (b), foreground image after optimize (c) and labeled foreground (d)

Foreground masks are the set of all foreground pixels, and are represented as binary images where white (value 1) pixels are foreground pixels and black (value 0) pixels are background pixels. Foreground pixels are extracted from the incoming current image frame through matching it with the background model [2]. For foreground mask correction which may need to fill holes or eliminate jutted pixels or a small pixel region, morphological operations such as opening and closing are usually employed as preprocessing procedures. To detect interesting objects, one needs to segment blobs that is defined as a set of all connected foreground pixels. After separate blobs are segmented, rectangular region enclosing each blob tightly are calculated for later processing like object tracking.

2.2 Morphology

The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple "probe" is called structuring element, and is itself a binary image. The popular structure elements include disk, square, and cross-shaped element of 3x3 size.

The basic operations of binary morphology are dilation, erosion, closing, and opening. A dilation operation enlarges a region, while an erosion

makes it smaller. A closing operation, defined as an operation of applying erosion after dilation, can close up internal holes in a region and eliminate "bays" along the boundary and a opening, defined as an operation of dilation after erosion, can get rid of small portions of the region that jut out from the boundary into the background region.

2.3 Connected Component Labeling with Union-Find Structure (CCLUF)

Connected components labeling scans an image and groups its pixels into components based on pixel connectivity by separated labels. In a foreground mask image, the connectivity exists between foregrounds pixel by 8-neighborhood relation of the pixel. Extracting and labeling of various disjoint and connected components in an image is central to many automated computer vision applications including intelligent visual surveillance.

Many algorithms have been proposed to deal with connected component labeling problem [6,7,8,9,10]. The promising efficient algorithms usually employ two-pass procedures utilizing union-find structure[9,10] which is also adopted in this paper. The *union-find* algorithm, which dynamically constructs and manipulates the equivalence classes efficiently by tree structures. Each disjoint set is stored as a tree structure in which a node of the tree represents a label and points to its one parent node. This is accomplished with only a vector array PARENT whose subscripts are the set of possible labels and whose values are the labels of the parent nodes. A parent value of zero means that this node is the root of the tree.

Fig. 3 shows the union-find structure for two label sets.

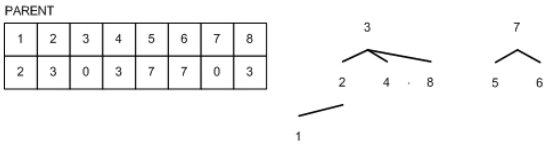


Fig. 3. The union-find structure for two label sets.

In the below, we briefly explain the two-pass connected component labeling with union-find structure.

In the 1st pass, the algorithm scans through the binary image from left to right and row by row. It assigned the smallest label among those of the scanned foreground pixel neighbors, and each such equivalence class found is entered in the union-find structure. At the end of the 1st pass, each equivalence class has been completely determined and has a unique label, which is the root of its tree in the union-find structure. The second pass through the image then performs a replacement of the temporary label into a final label which is done by assigning to each pixel the label of its equivalence class.

Fig. 4 shows an example input binary image where f means foreground pixel. Fig. 5 shows the results of the 1st pass. Fig. 6 shows the final labeling after second pass.

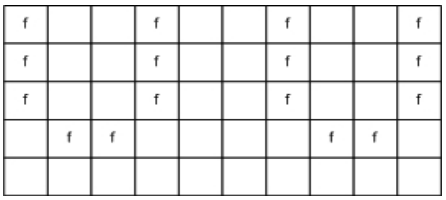
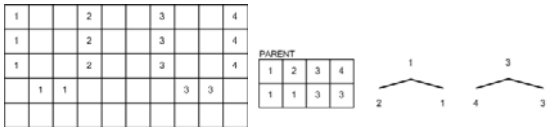


Fig. 4. Foreground mask where f means foreground pixel



(a) (b)

Fig. 5. The results of the 1st pass.

1			1			3			3
1			1			3			3
1			1			3			3
	1	1					3	3	

Fig. 6. Final result

Ⅲ. The Proposed Approach: CLNF

From the background review in Section 2, one can easily understand that it is difficult to process morphological operations like closing and opening in parallel with connected component labeling algorithm. Usual way to do processing both operations is to process sequentially: morphological operations, first and then connected component labeling, next But, morphological operations are computationally expensive.

Thus, we first develop a foreground mask correction method which can be incorporated into a connected component labeling routine and mask correction operations like filling up holes or eliminating jutted pixels or isolated pixels can be processed while connected component labeling routine is processing.

3.1 Neighbor Foreground Pixel Propagation (NFPP)

A hole in the foreground region is highly likely to be a set of unextracted foreground pixels region and jutted pixels or isolated pixels in the foreground mask is also highly likely to be wrongly extracted as foreground pixels. One can observe in the foreground mask that a pixel belonging to a hole is likely to have more foreground pixel neighbors and on the other hand, a jutted pixel or an isolated pixel in a foreground mask is likely to have less foreground pixel neighbors.

Here, we define the foreground neighborhood probability of a pixel X, P(X) as the ratio of the number of the foreground neighbor pixels over the number of all neighbors. In this paper, we use 8-neighborhood as in Fig. 7.

1	2	3
4	*	5
6	7	8

Fig. 7. 8-Neighbors

From this observation, we develop a foreground mask correction method, "Neighbor Foreground Pixel Propagation (NFPP)".

Neighbor Foreground Pixel Propagation algorithm

In binary image where pixels have two states, value 1 (foreground pixel) and value 0 (background pixel), Pixel state is propagated to its neighbors to the right and below, depending on its foreground neighborhood probability. That is, When one scans the binary image from left to right in row by row, a state of a pixel X will be changed depending on its foreground neighborhood probability as follows.

$$state(X) = \begin{cases} 1\ ; & P(X) \geq \frac{1}{2} \\ 0\ ; & others \end{cases}$$

Processing of NFPP method is graphically illustrated Fig. 8.

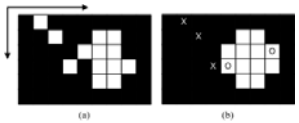


Fig. 8. (a) Foreground mask, (b) The result after propagation where X are spots that were changed to background (Black), O are elements that were turned into foreground (White).

Fig. 9 shows the result of applying NFPP method to a foreground mask which was actually obtained

using Gaussian Mixture Background Model [2]. One can see that the blob after NFPP looks clearer and plumper than before.

Fig. 9. (a) Foreground mask, (b) Result after NFPP

NFPP is a simple but shows a good result in blob correction. In addition to a good blob correction property, another more important property of NFPP preprocessing method is that it can be processed while CCl algorithm is being processed since both NFPP and CCL utilize pixel neighbor checking. This property can save computational time a lot in blob detection. Based on this property, we propose CLNF(CCLUF with NFPP) approach in blob detection, which leads to a faster and preciser blob detection than conventional blob detection algorithms.

3.2 CLNF Approach (CCLUF with NFPP)

3.2.1 Outline

In this paper, we propose a blob detection algorithm, CLNF which combines CClUF(Connected Component Labeling with Union and Find Structure) and NFPP(Neighbor Foreground Pixel propagation). Work flow of CLNF approach is shown in Fig. 10 with two passes (scans).

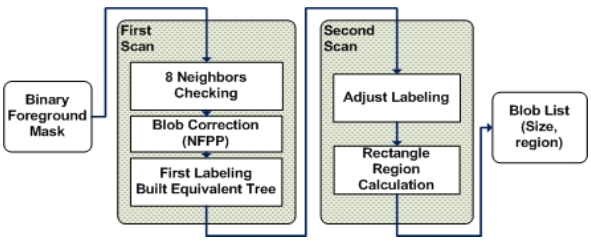


Fig. 10. Work flow of CLNF

While first labeling and making Union-Find structure is doing in the 1st pass (scan), a blob correction using NFPP is also processed. In the 2nd pass(scan), temporarily assigned labels of the pixels are adjusted into a correct label using the accomplished union-find structure and finally rectangle blob regions are calculated using the labelled connected component.

3.2.2 Pseudo Code

For clearer explanation of our proposed blob detection algorithm, CLNF, the pseudo codes of CLNF are sketched in the below.

Pseudo code for 1st scan

```
-NewLabel: the label value assigned to current pixel
-TargetLabel: the Label which will be replaced by NewLabel
-Image[]: binary image
-Labels[]: label map of Image[]
-LabelTree[]: PARENT array in Union-Find Structure explained in Section 2.2
        current max label :=0;
With each Image[X]
Start
  With each of 8 neighbors of a pixel X
  Start
    - Count Foreground Pixel
    - To a foreground pixel neighbor, assign NewLabel as the smallest label among its left or above foreground pixel neighbors' labels
    - In case it has two different neighbor labels, assign TargetLabel as the bigger label
  End
  If foreground neighborhood probability 〉= 1/2
  Then
    Image[X] = Foreground;          // Foreground =1
  Else
    Image[X] = Background;          // Background=0;
  End
  If Image[X] is foreground
  Then
    If NewLabel and TargetLabel assigned
    Then
      Recalculate new parent NewLabel for TargetLabel
      LabelTree[TargetLabel] = NewLabel
      Labels[X] = NewLabel
    Else
      Increase current max label by 1;
      Labels[X] = Current max label;
    End
  End
End
```

Pseudocode for 2nd scan

```
-sizeBlobs[]: list of each blob size
-blobsRectangles[]: list of blob rectangle regions
With each Labels[X]
Start
  Relabel Labels[X] by its parent note in tree LabelTree[]
  Increase sizeBlobs[Labels[X]] by 1;
  Recalculate blobsRectangles[Labels[X]]
End
```

Ⅳ. Experimental Results

4.1 The preciseness of NFPP

We compare NFPP and morphology by relying on the preciseness of blob. Fig. 11 shows the result that NFPP is simpler but more effective in foreground blob correction than morphological operations.

Foreground mask in Fig.11(a) was obtained using Gaussian Mixture Background Model, and morphology result in Fig. 11(b) and 11(c) is processed using 3x3 kernels.

Fig. 11. Foreground mask (a), result after morphology (Open) (b), after morphology (Open + Close) (c), after NFPP (d)

Certainly, if one chooses a more suitable kernel size for morphology, one can get a similar clear blob as in NFPP, also. However, in that case one must pay more computational cost.

4.2 Comparision about processing time

In order to compare the computational cost between the conventional blob detection algorithm using morphology (CONV, hereafter) and the proposed blob detection algorithm, CLNF, we did two experiments.

First experiment calculates the processing time for blob detection in one frame Fig. 11(a) for both CONV and CLNF. And the second experiment calculates the processing time for continuous blob detection in movies. The resolution of each image frame is 320x240 and sizes of movie1 and movie2 are 4.67 MB(221 frames) and 245MB(1117 frames). Both results of experiments have been done using a Windows PC with a 3GHz Pentium 4 Core of 3GHz and 3GB main memory is showed in Table 1a and 1b, respectively.

	time(sec)	movie1(sec)	movie2(sec)
CONV	0.017714	4.297988	21.053498
CLNF	0.003722	0.874097	4.111587

(a) (b)

Table 1.(a) Comparison of the processing time between CONV and CLNF for one frame of Fig. 11(a). (b) for two movies.

The results of Table 1a and 1b certainly shows that the proposed blob detection algorithm is faster than the conventional blob detection algorithm using morphological operations for foreground mask correction.

V. CONCLUSION

In this paper, we proposed a fast and precise blob detection algorithm for visual surveillance. The main idea of the proposed approach is to develop a blob correction method which can be efficiently processed in parallel with the connected component labeling algorithm. NFPP, designed in this paper utilizes 8-neighbors checking which is also employed in CCL so that it can be incorporated in parallel into CCL processing. The experiment results show the effectiveness of the proposed blob detection algorithm with respect to the processing time and the preciseness in blob detection.

ACKNOWLEDGMENT

The authors would like to acknowledge the following support for the accomplishment of this

work: Soongsil University Research Fund and BK21 of Korea.

▌REFERENCES ▐

[1] J. Sai and J. Ariyama, Computer Vision Workload Analysis: Case Study of Video Surveillance Systems, Springer-Verlag: Tokyo, Japan, 2000.

[2] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," ICPR 2004, pp. 28-31.

[3] P. Soille, "Opening and Closing," in Morphological Image Analysis: Principles and Applications, 2nd ed. Berlin, Germany: Springer, 2003, ch. 4, pp. 105-137.

[4] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, "Morphological Image Processing," in Digital Image Processing using MATLAB. Upper Saddle River, NJ: Pearson Prentice Hall, 2004, pp. 334-377.

[5] E. R. Dougherty and R. A. Lotufo, Hands-on Morphological Image Processing. Bellingham, Wash.: SPIE Optical Engineering Press, 2003.

[6] Rosenfeld, A., Pfaltz, J.L. "Sequential Operations in Digital Processing", JACM, 13, 417-494, 1966.

[7] R. H. Haralick, Some neighborhood operations, In M. Onoe, K. Preston, and A. Rosenfeld, 19(Eds.) Real Time/Parallel Computing Image Analysis, 1981, Plenum Press, New York.

[8] R. Lumia, L. Shapiro, and O. Zuniga, A New Connected Components Algorithm for Virtual Memory Computers, Computer Vision, Graphics, and Image Processing, 22(1983) 287-300.

[9] C. Fiorio and J. Gustedt, Two linear time Union-Find strategies for image processing, Theo-retical Computer Science, 154(1996) 165-181.

[10] R. E. Tarjan, Efficiency of a Good But Not Linear Set Union Algorithm, Journal of the As-sociation for Computing Machinery, 22(1975) 215-225.