Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/081L41

A Traversing and Merging Algorithm of Blobs in Moving Object Detection

Yong Zhou[*] *and Yi Li*

School of Computer Science and Technology , China University of Mining and Technology, Xuzhou Jiangsu 221116, P. R. China

Received: 17 May. 2013, Revised: 11 Sep. 2013, Accepted: 12 Sep. 2013
Published online: 1 Apr. 2014

Abstract: In the frame of surveillance video, the colour of both moving objects and background distribute randomly. Consequently, the foreground image obtained by background subtraction or temporal differencing always contains many discrete blobs, which are isolated but actually belong to the same object. These blobs influence the efficiency of later process such as recognition, classification, and activity analysis. In order to organize the foreground blobs, a traversing and merging algorithm of blobs is proposed in this paper. The experimental results show that the proposed algorithm can merge blob effectively.

Keywords: Moving Object Detection, Background Subtraction, Target Blob, Traversing, Merging.

1 Introduction

Moving object detection is widely used in the video surveillance system. There are three conventional approaches to detect moving objects: background subtraction, temporal differencing and optical flow. Background subtraction is extremely sensitive to dynamic scene changes because of lighting and extraneous events (i.e.. stationary objects move) [1,2]. Temporal differencing is adaptive to dynamic environments, but it is difficult to extract all relevant feature pixels [3]. Optical flow can be used to detect independently moving objects in the presence of camera motion; however, most methods are computationally complex, and cannot be applied to full-frame video streams in real-time without specialized hardware [1,4].

Aiming at the problems above, adaptive background subtraction and three-frame differencing algorithm have been combined to overcome the shortcoming of each other [1]. In [5], a novel nonparametric background model was proposed to handle the case that the background is cluttered and not completely static. In [6], an on-line auto-regressive model was introduced to capture and predict the behavior the dynamic scenes. Literature [7] first formed a spatio-temporal entropy image (STEI) by measuring color variation in multiple successive frames, then employed the morphological methodology to extract salient motions or moving objects from STEI. Haritaoglu [9] studied the detection, tracing, activity analysis method for human body in outdoor environment.

In these studies above, various algorithms were proposed and successfully applied to moving object detection problems. However, as far as we know, there have little discussions about the fragmented foreground blobs. Generally, discrete and isolated blobs in the foreground image influence the efficiency of later process such as recognition, classification, and activity analysis. To address this problem, we proposed an algorithm for blob traversing and merging in this paper.

2 Adaptive Background Subtraction

Moving objects often result in a change of gray value on a pixel when pass through this pixel (assuming that the moving objects and the background are distinct in gray value), as shown in Fig. 2.1,. This fact is the theory basis of background subtraction and temporal differencing algorithm. Hence, the moving objects will expose themselves in the foreground which is a set of absolute difference values between current frame and background.

To make the presentation clear, a few definitions are first introduced.

[*] Corresponding author e-mail: yzhou@cumt.edu.cn

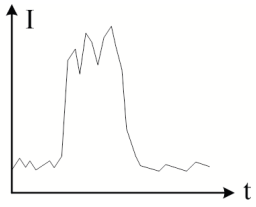

Figure 2.1:Moving objects passing through a pixel cause a pulse of gray value

Definition *Target Pixel*: Pixel whose gray value is different from the background it covered is called *Target Pixel*.

Definition *Target Blob*: The connected region consisted of target pixels is called Target Blob.

Definition *Foreground Image*: A binarize image where the target pixels have a gray value of 255, and the background pixels have a gray value of 0.

On the basis of the definitions above, let $I_n(x)$ and $B_n(x)$ represent the gray values of moving object and background at a pixel position x in the n^{th} frame respectively, then the *Target Blobs* can be obtain by

$$O_n = \{x : |I_n(x) - B_n(x)| > T_n(x), x \in A\} \tag{1}$$

where $T_n(x)$ is a threshold to describe a statistically significant change of gray values at the pixel position x, and A represents the set of pixel coordinate in one frame.

According to (1), we can compute the *Foreground Ima e* by using

$$F_n(x) = \begin{cases} 0, & x \in A - O_n, \\ 255, & x \in O_n, \end{cases} \tag{2}$$

In order to adapt to the lighting change and background disturbance, both the background model $B_n(x)$ and the difference threshold $T_n(x)$ should be updated over time as:

$$B_{n+1}(x) = \begin{cases} \lambda B_n(x) + (1-\lambda) I_n(x), & x \in A - O_n, \\ B_n(x), & x \in O_n, \end{cases} \tag{3}$$

$$T_{n+1}(x) = \begin{cases} \lambda T_n(x) + (1-\lambda)(5 * |I_n(x) - B_n(x)|), & x \in A - O_n, \\ T_n(x), & x \in O_n, \end{cases} \tag{4}$$

where λ is a time constant that specifies how fast new information supplants old observations, and $0 \leq \lambda \leq 1$. We can get from [3] and [4] that, the bigger λ is, the faster $B_n(x)$ and $T_n(x)$ update. In addition, the restricted condition $X \in A - O_n$ or $x \in O_n$ indicates that whether $B_n(x)$ and $T_n(x)$ should be updated or not. The initial value of $B_0(\bullet)$ is set to be the first frame, namely $B_0(\bullet) = I_0(\bullet)$, and $T_0(\bullet)$ is set to be a predefined non-zero threshold.

Table 1:Blob Descriptor

Items	**Information**
ID	Identifier of the blob
TPCount	Count of Target Pixels within the blob
Left	Left boundary of the blob
Right	Right boundary of the blob
Top	Top boundary of the blob
Bottom	Bottom boundary of the blob
x	Horizontal coordinate of the blob centroid
y	Vertical coordinate of the blob centroid

3 Traversing and Merging Algorithm of Blobs

Section II introduced the adaptive background subtraction algorithm. In this section, we describe the proposed traversing and merging algorithm of blobs.

In order to obtain the information of blobs, such as location, shape and so on, it is inevitable to traverse the *Foreground Image (FI)*. Since we have no priori knowledge about blobs, it is necessary to search from the front (at the first row and the first column) of *FI*. The searching procedure switch to blob traversing algorithm once a new *Target Pixel (TP)* is encountered. When the traversing work end, the working flow will switch back to the searching procedure which will be terminated as soon as it reaches the end (at the last row and the last column) of *FI*.

The *TP* searching module maintains a blob list. As shown in Table 1, each element (referenced as blob descriptor) in the list is used for storing the description information about a single blob. At the beginning of blob traversing algorithm, a blob descriptor is created, and it will be updated during the process of traversing. Finally all the descriptor are inserted into the blob list in order. The order is determined by the size of *BCA* which is computed by (5) (see section *B*).

A.Target Blob Traversing

In the *Target Pixel* searching process, the first found *TP* is treated as the root of the blob, and its 8 neighbor nodes are treated as child nodes. As Fig. 3.1shown, the root node is identified by 0, and the child nodes are identified according to their orientation to the root node.

The *TP* found by the searching subroutine is insert into a *Queue of Sibling Node (QoSN)* firstly. Before switching to blob traversing routine, a new *Blob Descriptor* is created. While the working flow switches to traversing subroutine, the traversing algorithm takes one node out of *QoSN* (treated as root node), marks it as an accessed node in the *Foreground Image*, and updates the *Blob Descriptor*; then checks its child nodes, and inserts new *TP* into *QoSN*. After all of the new *TP* inserted, the traversing algorithm moves to the first valid child of

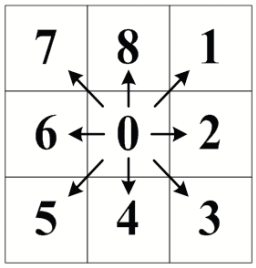

Figure 3.1:Number of child node

current root node, repeats the procedure above until the *QoSN* is empty. The whole process is illustrated in Fig. 3.2, where the 0 node is root node. When the traversing algorithm moves to one of its child nodes, only those nodes with shadows need to be checked.

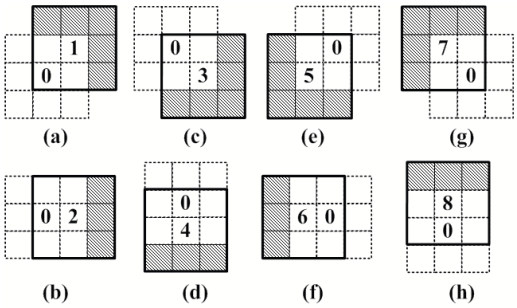

Figure 3.2:The blob expanding process

In order to demonstrate the run of the traversing algorithm, we mark each *TP* with a triple, i.e. Current-ID, Parent-ID, Sibling-Order, where three triple represent the number of current node, the number of parent node, and the order current node in siblings, respectively. Since the root node of *Target Blob* has no parent and is unique, the Current-ID is 1, the Parent-ID and Sibling-Order both are 0.

A 8×8 *Foreground Image* is shown in Fig. 3.3. The line numbers are listed in the first row and column. Each cell in the matrix represents a pixel. Those cells with shadow represent the *Target Pixels* which are labeled with triples. The rectange surrounding the *Target Blob* marks the covering area of the blob.

B.Target Blob Merging

When the blob traversing process ended, we have much information about the blobs, which includes the quantity of blobs, blob size, boundary, centroid. However, as mentioned before, these blobs are isolated and discrete. Hence, we have to merge these blobs in the next.

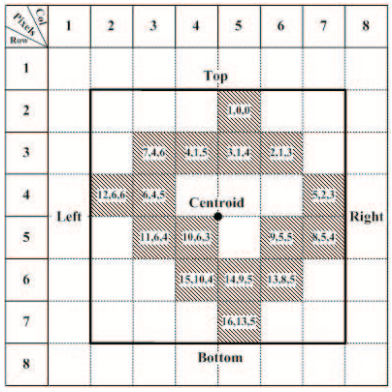

Figure 3.3:Illustration of the blob traversing algorithm

First of all, we give the criterion of blob merging as follow:

Definition (*Blob Covering Area*):The region coverd by the circumscribed rectangle of blob *A* is called the *Blob Covering Area (BCA)* of *A* (see Fig. 4.1). The size of *BCA* is computed by (5). If the centroid (computed by (6) and (7)) of blob *B* is within the extent of the *Blob Covering Area* of *A*, *B* is considered to be covered by *A*.

The size of *Blob Covering Area* is computed as follow:

$$BCA = |Right - Left| \times |Bottom - Top| \tag{5}$$

where *Left*, *Right t*, *Top* and *Bottom* represent the boundary of *BCA* respectively.

The centroid of blob is computed by:

$$cx = \frac{1}{N} \sum_{i=L}^{R} \sum_{j=T}^{B} x(i,j) \tag{6}$$

$$cy = \frac{1}{N} \sum_{i=L}^{R} \sum_{j=T}^{B} y(i,j) \tag{7}$$

$$x(i,j) = \begin{cases} 0, (i,j) \notin O_n, \\ i, (i,j) \in O_n, \end{cases} \tag{8}$$

$$y(i,j) = \begin{cases} 0, (i,j) \notin O_n, \\ j, (i,j) \in O_n, \end{cases} \tag{9}$$

where *L*, *R*, *T* and *B* represent the left, right, top and bottom boundary of the blob, N denotes the count of *TPs* within the blob, cx and cy denote the horizontal and vertical coordinate of the centroid.

Criterion of blob merging: Asuming there are two blobs in the *Foreground Image*. According to the *BCA* size, merge them if the smaller one is covered by the bigger one, otherwise, they are considered to be independent.

Fig. 3.4 demonstrates the blob merging process. The blob searching and traversing routines detect two blobs in Fig. 3.4(a), which are marked by blue rectangle. Red

(a) Blob before merged (b) Blob after merged

Figure 3.4:Illustration of the blob merging algorithm

points are the centroids of blobs. The result of blob merging algorithm is shown in Fig. 3.4(b), where two blobs have been merged.

C.Algorithm Summary

Part 1. Steps of blob traversing algorithm

1)According to (1) and (2), compute the Foreground Image (*FI*) ;

2)Search the *FI* for *Target Pixel (TP)*, if a *TP* is found, goto next step; otherwise, goto step 6);

3)Mark the *TP* as a visited node in *FI*, insert it into the *Queue of Sibling Node (QoSN)*, and create a new *Blob Descriptor (BD)*;

4)Check the *QoSN*, if it is empty, insert *BD* into the *List of Blob (LoB)* and goto step 2); otherwise, pop one node out of *QoSN*, regard it as *Current Node (CN)*, and update *BD* according the information carried by *CN*;

5)Check the neighbor nodes of *CN*, insert new TP into *QoSN*, mark it as visited node in FI, then goto step 4);

6)Terminate the algorithm.

Part 2. *Steps of blob merging algorithm*

1)Check the List of Blob, if it is empty, goto step 5);

2)Take the smallest blob out of *LoB* (according to their *BCA* size), and denote as *B*. if its centroid is covered by any rest blob in *LoB*, denote the firstly found blob as *A*, goto next step; otherwise, goto step 4);

3)Update the *Blob Descriptor* of *A*, if the *BCA* size of *A* change, reinsert *A* into the *LoB* at the proper location according to its *BCA* size, and goto step 2) ;

4)Insert blob *B* into the *List of Independent Blob (LoIB)*, goto step 1);

5)Terminate the algorithm;

4 Experiment

A video in indoor environment is chosen as the data source. In this video, the moving object is a pen. The C++ implementation of our algorithm uses OpenCV library, which is compiled under VC++ 2005 IDE. The system configuration is a PC with Core 2 E7500 CPU and 2GB memory.

As shown in Fig. 4.1, the first row is the results of blob searching and traversing algorithm. The detected blobs are marked by red rectangles. The second row is the results of blob merging algorithm.

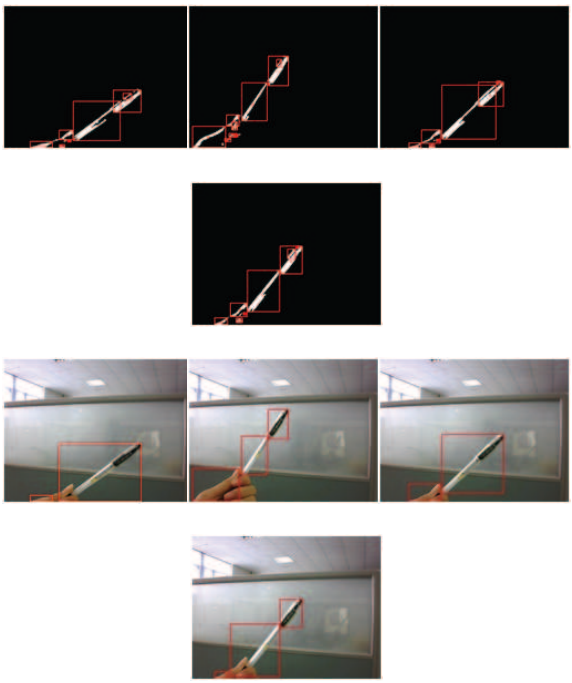

Figure 4.1:Experimental results of proposed algorithm

5 Conclusion

The foreground image obtained through background subtraction or temporal differencing always contains many isolated and discrete blobs, which actually belong to the same object. As a result, the later process is influenced greatly. In this paper, we proposed a blob traversing and merging algorithm. The blob traversing algorithm is used for acquiring the information about blobs in detail. On the basis, The blob merging algorithm is proposed to merge these discrete blobs. The experimental results show that our algorithm could merge the blobs effectively.

References

[1] Robert T. Collins, Alan J. Lipton. A System for Video Surveillance and Monitoring. VSAM Final Report. Carnegie Mellon University. Technical Report, CMU-RI-00-12, (2000).

[2] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multi-sensor video surveillance. In Proceedings of the 1998 DARPA Image Understanding Work-shop, **1**, 3-24 (1998).

[3] Alan J. Lipton, H. Fujiyoshi, and R. S. Patil. Moving target classification and tracking from real-time video. IEEE Workshop Applications of Computer Vision, **8**, 8-14 (1998).
[4] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. International Journal of Computer Vision, **12**, 42-77 (1994).
[5] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In ECCV, 751-757 (2000).
[6] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In CVPR, (2003).
[7] Yufei Ma, Hong-jiang, Zhang. Detecting Motion Object by Spatio-Temporal Entropy. IEEE International Conference on Multimedia and Expo (ICME 2001). Waseda University, Tokyo, Japan, **8**, 22-25 (2001).
[8] Khedr, Ahmed M., Osamy, Walid. Target Tracking Mechanism for Cluster Based Sensor Networks. Applied Mathematics and Information Science, **1**, 287-303 (2007).
[9] Ismail Haritaoglu, Davlid Harwood, Larry S Davis. W4: A Real Time Surveillance of People and their Activities. IEEE Transaction on Pattern Analysis and Machine Intelligence, **22**, 809-830 (2000).

Yong Zhou was born in Xuzhou, Jiangsu, China, in September 1974. He received the Ph.D. degree in control theory and control engineering from China University of Mining and Technology, Xuzhou, Jiangsu, in 2006. His research interests include data mining, genetic algorithm, artificial intelligence and wireless sensor network. He has published more than 20 papers in these areas. He is currently an Assistant Professor in School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu, China..

Yi Li was born in Xuzhou, Jiangsu, China, in November 1986. He received the Master Degree in Computer Applications Technology from China University of Mining and Technology, Xuzhou, Jiangsu, in 2012. His research interests include data mining and artificial intelligence.