REPORT

Abstract

Current Background Subtraction (BGS) algorithms are mostly pixel-based methods. We propose an Interest-Point(IP)-based BGS algorithm applicable in IP-based Computer Vision applications. Based on a block-wise processing strategy, the frames are divided into blocks of the same size. IPs inside each block are together Events. Throughout the frame sequence, the algorithm stores the Events in each block as well as the numbers of their occurrences (Repetition Index (*RI*)) in a Binary Tree. The *RI* is used to classify Events as either background or foreground. The background Events appear significantly more often than foreground Events. Events with an RI greater than a certain threshold are classified as background, the rest as foreground. This Event classification is used to label IPs of frames into the foreground and background IPs. Experimental results quantitatively show that the proposed algorithm delivers a good subtraction rate in comparison with other BGS approaches. Moreover, it creates a map of the background usable for further processing, it is robust to changes in illumination and can keep itself updated to changes in the background.

*Key Words*: Background Subtraction; Foreground Detection; Interest Points.

# 1 Introduction

The segmentation of areas of an image related to moving objects, foreground, from the areas related to static objects in the scene, background, is called Background Subtraction (BGS) when the processed image is captured by static cameras [1]. This is the earliest stage of many Computer Vision (CV) applications such as human motion analysis, automated surveillance, video indexing, and vehicle navigation. Therefore, it exhibits a strong influence on further processing [2]. Accordingly, a great deal of research has been conducted on BGS over the past few years and many algorithms have been proposed, most of them pixel-based. They rely on the difference between pixels either *individually* [3], using the pixels’ illumination, or *regionally*, using the texture of a group of pixels in the form of blocks [4] or clusters [5], to model and update the background [6].

In contrast, Interest Points (IPs) have not so far been used for BGS. As the most lightweight way of object representation [7], they represent well-defined features of the image such as corners. This makes them superior to other object representation methods, such as Kernel and Silhouette, in terms of the speed, accuracy and robustness [8]. IPs have delivered a high level of descriptive power and robustness to illumination changes [9]. Many remarkable IP detector and local feature descriptors such as the Scale Invariant Feature Transform (SIFT)

2.2 Technical Description of Algorithm:

For the purpose of block-wise processing, the image is divided into blocks of *N*×*M* pixels (Fig. 1). In terms of hardware implementation and efficiency, it would be better to select *N* and *M* as a power of 2 because the memory blocks are counted based on powers of 2 too. Fig. 1 shows a frame of 320 × 240 pixels along side its blocks in yellow colour and the structure of a block. The extracted IPs of any frame are assigned to appropriate blocks based on their location (*x,y*) in the image plane. Consequently, the blocks related to the smoother areas of the image have no IPs, some blocks have a few IPs, and those associated with the highly textured areas of image have higher number of IPs.

To store the Events of blocks for consecutive frames, firstly we need to define a unique tag for any Event using its composing IPs. This tag is composed of some numbers, corresponding to the location of IP in the block, separated by commas e.g. {9*,*21*,*39}. To do this, the 2D coordinates of pixels in the block is mapped into a 1D coordinate by numbering pixels from 0 at the top left corner of the block, and then counting along each row from left to right to *N*×*M-1* at the bottom right corner (Fig. 1(b)).

Now, the Events as well as their number of repetitions should be stored. To preserve the speed and efficiency, the Binary Search Tree (BST) [40], which is a fast way of storing, sorting and searching, is used. On this basis, a Binary Tree (BT) is created for each block and its Events throughout the frames are stored in the tree as well as the numbers of their occurrences (Repetition Index (*RI*)). If any Event is happening for the first time, a new node with *RI* of 1 is created in the tree. Otherwise, the node corresponding to Event is found and its *RI* is increased by 1. So, the BTs of blocks summarize the Events and *RI*s of blocks.

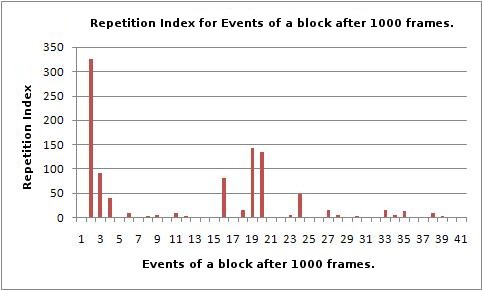


Figure 2: Repetition Index for Events of a block after 1000 frames.

As stated before, the background IPs appear as specific Events regularly at fixed blocks. Regardless the rare changes in the position of these IPs in blocks, due to changes in illumination and low resolution, they appear significantly more often than a threshold criterion. As can be seen in Table 1, the non-dominant Events created by foreground IPs, have *RI* value less than the threshold. Therefore, the algorithm classifies any Event as dominant if its *RI* is equal to or greater than a threshold. The algorithm is summarized in Algorithm 1.

Algorithm 1 IP-Based BGS Algorithm

1: Input: IPs of frames *k* = 1*,...,K*.

2: Output: background IPs & foreground IPs of each frame.

3: *Initialization:*

4: for frame *k* = 1 do

5: Divide image frame into *B* blocks.

6: Create a Binary Tree (BT) for each block.

7: Create the lists of background events (BG-E-list) and background IPs (BG-IP-list).

8: end for

9: *Background Modelling & Subtraction:*

10: for every frame *k >* 1 do

11: *Background Modelling:*

12: for each IP *i* do

13: Assign IP *i* to the corresponding block based on its (*x,y*) coordinate.

14: end for

15: for each block *j* do

16: Create an Event *Em* from its assigned IPs.

17: Search Binary Tree *BTj* for Event *Em*.

18: if *Em* ∈ *BTj* then

19: Increase Repetition Index of Event *Em* by 1 (*RIm* = *RIm* + 1).

20: if *RIm > threshold* then 21: Add Event *Em* to BG-E-list.

22: Add IPs of Event *Em* to BG-IP-list.

23: end if

24: else

25: Add a new node in *BTj* and set its *RI* to 0.

26: end if 27: end for

28: *Background Subtraction:*

29: for each IP *i* do

30: if IP *i* ∈ BG-IP-list then

31: IP *i* ⇒ background IP.

32: else

33: IP *i* ⇒ foreground IP.

34: end if

35: end for

36: *k* = *k* + 1

37: end for

Table 1 and Fig. 2 show the stored Events for block 102 (7th row and column) in its BT after 1000 frames. As the algorithm assumes, only a few Events appear dominant. On the other hand, the non-dominant Events have been created by the foreground IPs when they have met this block. The dominant Events have been marked

√ with the ” ” sign in the ”*D*” column of the table. IPs which create these dominant Events are classified as background IPs (bold numbers in ”Event” columns of table). They are stored in a list of background IPs.

Fig. 3 shows the real image with its extracted IPs (FAST corner IPs in this case), foreground and background IPs in red and blue, respectively.

Table 1: The stored Events of block 102 if image for 1000 frames.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Event | RI | D | # | Event | RI | D | # | Event | RI | D |
| 1 | (363,110) | 1 | √ | 15 | (280,110,353) | 1 | √ | 29 | (303,171) | 1 |  |
| 2 | (303,14) | 325 | √ | 16 | (285,110) | 82 |  | 30 | (303,353,14) | 2 |  |
| 3 | (285,14) | 92 | √ | 17 | (285,11) | 1 | √ | 31 | (303,353) | 1 |  |
| 4 | (285) | 40 |  | 18 | (285,12) | 16 | √ | 32 | (303,394) | 1 | √ |
| 5 | (125,333) | 1 |  | 19 | (303,110) | 143 | √ | 33 | (323,14) | 13 |  |
| 6 | (110,353) | 9 |  | 20 | (303) | 134 |  | 34 | (323) | 6 | √ |
| 7 | (110) | 1 |  | 21 | (285,9) | 1 |  | 35 | (305,14) | 14 |  |
| 8 | (124,353,14) | 2 |  | 22 | (285,14,394) | 1 |  | 36 | (305,12) | 1 |  |
| 9 | (144,353) | 4 |  | 23 | (303,11) | 5 | √ | 37 | (323,110,353) | 1 | √ |
| 10 | (144,12,353) | 1 | √ | 24 | (303,12) | 49 |  | 38 | (323,110) | 10 |  |
| 11 | (144,353,14) | 10 |  | 25 | (303,110,353) | 1 |  | 39 | (343,14) | 3 |  |
| 12 | (280,110) | 2 |  | 26 | (303,110,394) | 1 | √ | 40 | (323,353) | 1 |  |
| 13 | (222,110,353) | 1 |  | 27 | (305,110) | 16 |  | 41 | (353,14) | 1 |  |
| 14 | (280,171) | 1 |  | 28 | (305) | 4 |  |  |  |  |  |

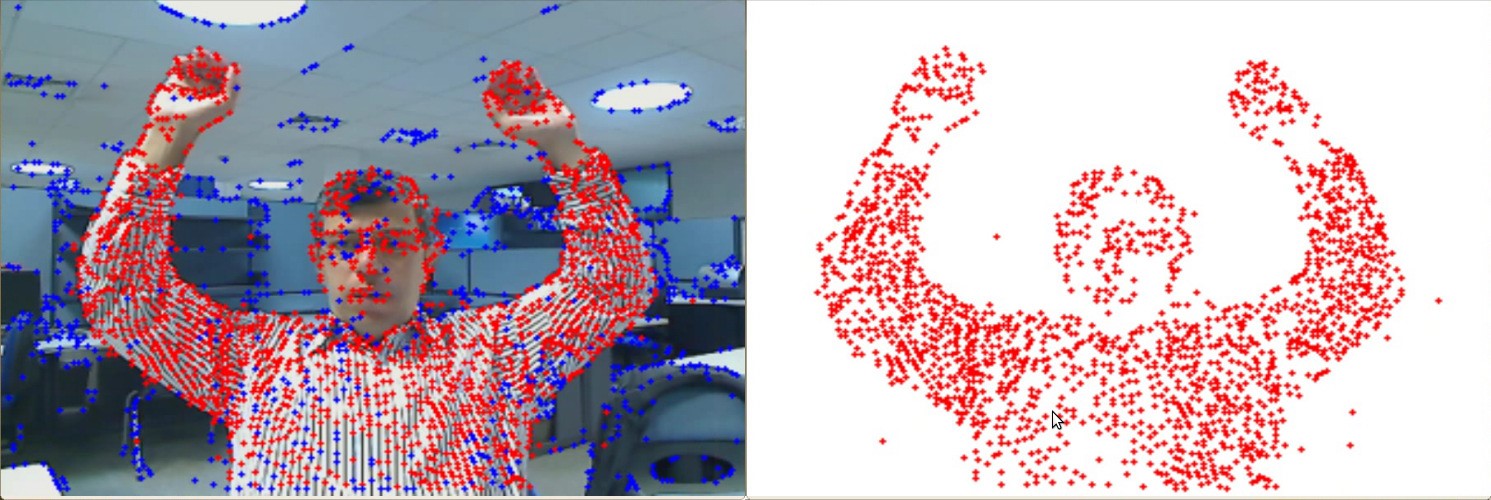


Figure 3: The real image with its IPs, red: foreground IPs, blue: background IPs.

REFERENCES

1. Herrero, S., Bescos, J.: Background subtraction techniques: systematic evaluation and comparative anal-´ ysis. In: Advanced Concepts for Intelligent Vision Systems, Springer (2009) 33–42
2. Moeslund, T.B., Hilton, A., Kruger, V.: A survey of advances in vision-based human motion capture and¨ analysis. Computer vision and image understanding 104(2) (2006) 90–126
3. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. Volume 2., IEEE (1999)
4. Fang, X., Xiong, W., Hu, B., Wang, L.: A moving object detection algorithm based on color information. In: Journal of Physics: Conference Series. Volume 48., IOP Publishing (2006) 384
5. Bhaskar, H., Mihaylova, L., Achim, A.: Video foreground detection based on symmetric alpha-stable mixture models. Circuits and Systems for Video Technology, IEEE Transactions on 20(8) (2010) 1133– 1138
6. Varcheie, P.D.Z., Sills-Lavoie, M., Bilodeau, G.A.: An efficient region-based background subtraction technique. In: Computer and Robot Vision, 2008. CRV’08. Canadian Conference on, IEEE (2008) 71–78
7. Aanæs, H., Dahl, A.L., Pedersen, K.S.: Interesting interest points. International Journal of Computer Vision 97(1) (2012) 18–35
8. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. Acm Computing Surveys (CSUR) 38(4) (2006) 13
9. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 2548–2555
10. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision 60(2) (2004) 91–110