Proceedings of the 33rd Chinese Control Conference
July 28-30, 2014, Nanjing, China

Block Compressed Sensing Based Background Subtraction for Embedded Smart Camera

LUO Rujun, WANG Yiyin , CHEN Cailian, YANG Bo, GUAN Xinping

Department of Automation, Shanghai Jiao Tong University,
and Key Laboratory of System Control and Information Processing,
Ministry of Education of China, Shanghai 200240
E-mail: {junjun, yiyinwang, cailianchen, bo.yang, xpguan}@sjtu.edu.cn

Abstract: Embedded smart camera networks represent an emerging direction of next generation surveillance systems. A big challenge to implement computer vision applications on embedded cameras is the limit of memory and computational capacity. Since background subtraction algorithms play a fundamental yet significant role of most computer vision applications, their memory requirements and computational efficiency should be taken into account in the design. In this paper, we propose an efficient hierarchical light-weight background subtraction approach by combining the pixel-level and the block-level background subtraction modules into a single framework so that it is capable of dealing with dynamic background scenes. Block compressed sensing theory is for the block-level module design to save memory and improve computational efficiency. Moreover, considering the continuity of foreground objects, a novel integral filter is designed for the pixel-level module to eliminate perturbations efficiently. Experimental results on various videos demonstrate superior performance of the proposed algorithm. The proposed light-weight algorithm only requires about 6.5 bytes per pixel, and is applicable for embedded smart cameras. Furthermore, as each block is processed independently, it can be implemented in parallel.

Key Words: Background subtraction, Hierarchical, Block compressed sensing, Light-weight, Real-time

1 Introduction

In recent years, embedded smart camera networks provide a promising way for intelligent surveillance systems. In most computer vision applications, foreground detection plays a basic yet significant role. However, many existing foreground detection algorithms [1] are too "heavy" (large memory requirements or low computational efficiency) to apply in the embedded platforms. Moreover, among these algorithms, the background subtraction method is the mostly used for its simplicity and efficiency. Therefore, a robust light-weight background subtraction algorithm is critical for embedded computer vision applications.

The basic idea of background subtraction is to automatically detect foreground objects by comparing current frame with a background model. However, this procedure rarely works in real world applications because the background model is always contaminated by perturbations in natural scenes, such as shadows, illumination changes, rippling water, swaying leaves and camera noises. Thus, in order to obtain a robust background model, a number of sophisticated methods based on complex statistical models have been proposed. For instance, the methods based on the Gaussian Mixture Model (GMM) [2][3] model every pixel with three to five Gaussian distributions, which is capable to deal with subtle illumination changes. Codewords-based methods [4][5] form codewords for each pixel to capture different values within a fixed period, which averagely require 6.5 codewords per pixel. Nevertheless, these sophisticated modeling techniques suffer from memory crises and they are not applicable for embedded smart cameras.

Recently, a few embedded platforms oriented background subtraction methods have been proposed [6-9]. Casares et al. [6][7] use four counters to record changes of each pixel in a fixed period and apply the statistical result to update the background model. This algorithm is fast and light-weight. Manzanera et al. [8] propose an efficient background subtraction algorithm based upon a sigma-delta filter. The sigma-delta detection filter approximates the background model as a simple nonlinear recursive module. This kind of approximation has taken advantage of the characteristics of embedded platforms without a floating point unit. Another popular real-time background subtraction method is the visual background extractor (ViBE) proposed by Barnich et al. [9]. ViBE is well-known for its random update strategy and low computational load. However, ViBE needs to store more than 20 background reference images (also called background samples), which is memory-consuming. Furthermore, most of the above methods are in the pixel level, and pixel-level methods are not robust enough against dynamic background scenarios.

In this paper, we propose a hierarchical light-weight subtractor based on block compressed sensing theories (BCS). Our contributions are three-fold: (a) Considering that pixel-level methods are sensitive to unavoidable perturbations in natural scenes, we combine block-level and pixel-level modules into a single system. (b) To break the "bottleneck" of the limited memory, we employ the block compressed sensing into the embedded system in the block level. We classify blocks in the compressive sensing (CS) domain instead of the original domain. This allows us to store data with less memory. In the block level,we use a modified random update strategy with only one-sixth of memory of ViBE. (c) Further utilizing the spatial information of each block, we design a light-weight filter, called the integral filter, to extract foreground pixels from foreground blocks. The experimental results on real data demonstrate superior performance of the proposed method.

This work was partially supported by Ministry of Science and Technology of China under National Basic Research Project 2010CB731803, by the NSF of China under the grants 61221003, 61290322, 61174127, 61273181, 61301223, and by Science and Technology Commission of Shanghai Municipal, China under the grants 13ZR1421800 and 13QA1401900.

The remainder of this paper is organized as follows. Section 2 introduces mathematic theories of CS. The main idea of the proposed algorithm is described in Section 3. Section 4 represents the experimental studies for evaluating the performance of the proposed Block Compressed Sensing based Background Subtraction (BCSBS) method and other five background subtraction methods including GMM1 [2], GMM2 [3], CodeBook [4], LW [6] and ViBE [9]. Conclusion is given in Section 5.

2 Preliminaries

Before illustrating the proposed algorithm, we present some preliminaries on CS [10][11]. CS is developed recently in signal processing, and has gained increasing popularity in computer vision applications.

2.1 Random Projection

Generally speaking, the image in the discrete cosine transform (DCT) or the wavelet basis domain is compressible, which implies that we can use less memory to store data. Here we introduce CS into the proposed background subtraction algorithm and detect foreground objects in the compressive sensing domain.

A random measurement matrix $\boldsymbol{R} \in \mathcal{R}^{m \times n}$ projects data from high-dimension image representation $\boldsymbol{x} \in \mathcal{R}^n$ to a low-dimension vector $\boldsymbol{y} \in \mathcal{R}^m$ by

$$\boldsymbol{y}=\boldsymbol{R}\boldsymbol{x} \tag{1}$$

where $m \ll n$ and $\boldsymbol{R}$ satisfies the Restricted Isometry Property (RIP).

Given a sparse signal such as a natural image, it has been demonstrated that a small number of randomly generated linear measurements can still preserve most of the salient information of the original signal [11]. Compressive sensing theories also guarantee the reconstruction of the original signal with a small error. Supported by the solid mathematic theories of CS, we can analyze image data in the CS domain instead of the high dimensional space, which reduces computation and saves memory. The setting of the random matrix $\boldsymbol{R}$ is stated as follows and the selection of m will be discussed in Section 4.1.

2.2 Random Measurement Matrix

Random measurement matrix plays an important role in compressive sensing and should be modeled properly. A typical measurement matrix satisfying the RIP is the random Gaussian matrix $\boldsymbol{R} \in \mathcal{R}^{m \times n}$ where $r_{i,j} \sim N(0,1)$. However, the random Gaussian matrix is not suitable for embedded platforms without a float point operation unit. In this paper, we adopt a circulant matrix [11][12] as the measurement matrix:

$$\boldsymbol{R} = \begin{bmatrix} a_n & a_{n-1} & \cdots & a_1 \\ a_1 & a_n & \cdots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & \cdots & a_m \end{bmatrix} \tag{2}$$

where $a_i \in \{-1, +1\}$ obeys the Bernoulli distribution.

It is demonstrated in [13] that a circulant matrix could ensure sparse reconstruction by l_1-minimazation in the sparsity up to a log-factor in the ambient dimension. In addition, the circulant matrix is generated by circularly shifting the elements of the first row of $\boldsymbol{R}$, which is proper for the implementation of embedded platforms.

3 Block Compressed Sensing Based Subtractor

In this section, we present the proposed background subtraction algorithm in details. Firstly, we detect foreground objects in the compressed sensing domain coarsely. Furthermore, we extract foreground objects in the pixel level through the integral filter to achieve a more accurate output. The flowchart of the BCSBS algorithm is shown in Fig. 1.

3.1 Previous Processing

In the pre-processing module, we segment every frame into non-overlapping blocks with size of 8×8 pixels (Note that 8×8 is the default block size in JPEG.). Then we compute projections based on CS for each block. So far, we have mapped a 64×1 ($8 \times 8 = 64$) vector of each block into a $m \times 1$ projected vector. Note that m should follow RIP, meanwhile small enough for memory saving and computational efficiency. Actually, results in Section 4 show that the proposed method performs well when $m = 8$ per block. The measurement matrix we have mentioned at Section 2.2 is generated as a global parameter and used for all blocks. For more details, we formulate this procedure at frame t as

$$\begin{aligned} \boldsymbol{B}_t &= \{\boldsymbol{b}_{t,i} | i = 1, 2, ..., N_b\} \\ \boldsymbol{Y}_t &= \{\boldsymbol{y}_{t,i} | i = 1, 2, ..., N_b\} \end{aligned} \tag{3}$$

where $\boldsymbol{Y}_t = \boldsymbol{R}\boldsymbol{B}_t$, N_b is the total number of blocks per frame. $\boldsymbol{B}_t$ is the block representation at frame t and $\boldsymbol{b}_{t,i}$ is the vector representation of the ith block of the tth frame. $\boldsymbol{y}_{t,i}$ is the projected vector of $\boldsymbol{b}_{t,i}$ and $\boldsymbol{Y}_t$ is the projected image representation of the block image $\boldsymbol{B}_t$.

Before carrying out all the modules of the proposed background subtraction method, we need to initialize the background model firstly. Details of how to obtain the initial background model are described in Algorithm 1. In the block level, the background model $\bar{\boldsymbol{B}}_t$ is sample-based, which is formed by a number of background samples. The background samples generated from a single frame constitute the initial background model in this level, which is similar to the initialization process of ViBE. Actually, since we

Algorithm 1 Initialization of the background model

Let $t = 0$ index the first frame and $\boldsymbol{I}_0$ denote the first frame
Make $\bar{\boldsymbol{Y}}_0$ and $\boldsymbol{M}_0$ empty and be the Initialization of $\bar{\boldsymbol{Y}}_t$ and $\boldsymbol{M}_t$
Let $N(p)$ denote a spatial neighborhood of a pixel p
Let $\bar{\boldsymbol{I}}_0^k$ denote the kth background sample at the first frame
on $k = 1 : N_s$ **do**
    on every pixel $p \in \boldsymbol{I}_0$, p is located at (i, j) **do**
        Location $(i_1, j_1) \in N(p)$ is chosen randomly by a uniform law
        Set $\bar{\boldsymbol{I}}_0^k(i, j) = \boldsymbol{I}_0(i_1, j_1)$
    end on
    $\boldsymbol{M}_0 = \boldsymbol{M}_0 + \bar{\boldsymbol{I}}_0^k$
    Set $\bar{\boldsymbol{B}}_0^k$ as the block representation of $\bar{\boldsymbol{I}}_0^k$
    Set $\bar{\boldsymbol{Y}}_0^k$ as the projected representation of $\bar{\boldsymbol{B}}_0^k$
end on
Then $\bar{\boldsymbol{Y}}_0 = \{\bar{\boldsymbol{Y}}_0^k | k = 1, ..., N_s\}$
Set $\boldsymbol{M}_0 = \frac{\boldsymbol{M}_0}{N_s}$

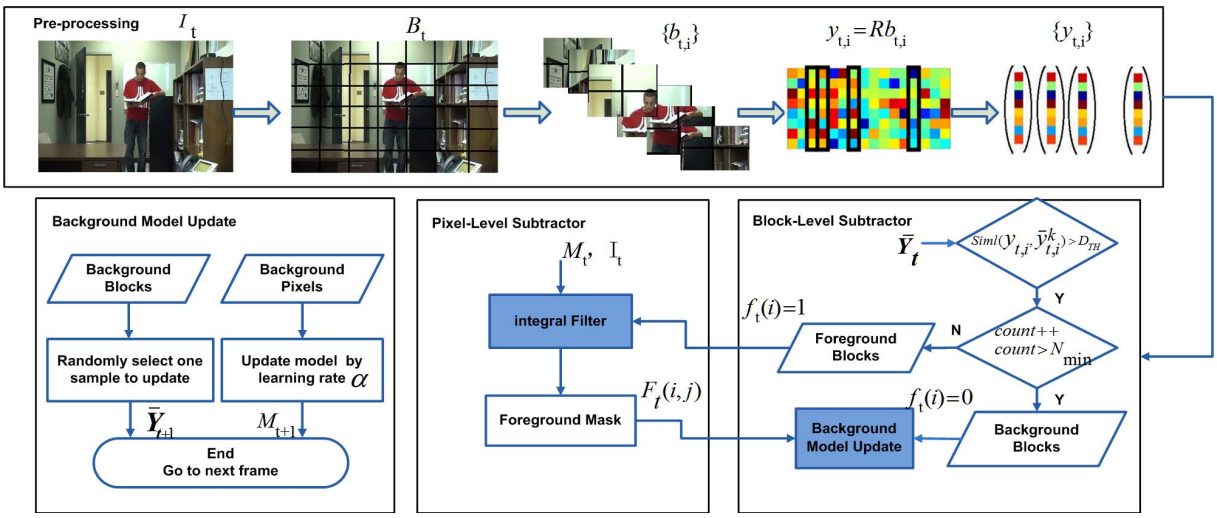


Fig. 1: Flow chart of the proposed BCSBS algorithm.

detect foreground blocks in CS domain, the actual background model in the block level should be the projected representation $\bar{\boldsymbol{Y}}_t$ of $\bar{\boldsymbol{B}}_t$. Moreover, the average of the initial background samples in the block level is taken as the initial background model $\boldsymbol{M}_t$ for the pixel-level subtractor. Here, $\bar{\boldsymbol{Y}}_t$ and $\bar{\boldsymbol{B}}_t$ are defined as follows, respectively.

$$\begin{aligned}\bar{\boldsymbol{B}}_t &= \{\bar{\boldsymbol{B}}_t^k | k = 1, .., N_s\}, \bar{\boldsymbol{B}}_t^k = \{\bar{\boldsymbol{b}}_{t,i}^k | i = 1, ..., N_b\} \\ \bar{\boldsymbol{Y}}_t &= \{\bar{\boldsymbol{Y}}_t^k | k = 1, .., N_s\}, \bar{\boldsymbol{Y}}_t^k = \{\bar{\boldsymbol{y}}_{t,i}^k | i = 1, ..., N_b\}\end{aligned} \tag{4}$$

where N_s is the total number of background samples. $\bar{\boldsymbol{B}}_t$ is the block representation of the background sample set at frame t, and $\bar{\boldsymbol{B}}_t^k$ is the kth sample of $\bar{\boldsymbol{B}}_t$. $\bar{\boldsymbol{Y}}_t$ and $\bar{\boldsymbol{Y}}_t^k$ are respectively the projected representation of $\bar{\boldsymbol{B}}_t$ and $\bar{\boldsymbol{B}}_t^k$.

3.2 Block-level Subtractor

In the block level, the background subtraction is carried out coarsely with the random update strategy similar to ViBE [9]. We detect foreground objects in the CS domain instead of the original image domain and in the block level instead of the pixel level.

For a new incoming frame $\boldsymbol{I}_t$ at frame t, we firstly apply the pre-processing module to it and obtain the projected representation $\boldsymbol{Y}_t = \{\boldsymbol{y}_{t,i} | i = 1, 2, ..., N_b\}$. The ith block is classified as the background block if at least $N_{\min}$ corresponding projections $\{\bar{\boldsymbol{y}}_{t,i}^k\}$ of the projected background sample set $\bar{\boldsymbol{Y}}_t$ are similar to $\boldsymbol{y}_{t,i}$. Here, $N_{\min}$ denotes the minimum number and the similarity is defined by the cosine similarity with a certain threshold. The block-level subtractor formulas as

$$Siml(\boldsymbol{y}_{t,i}, \bar{\boldsymbol{y}}_{t,i}^k) = \frac{{\boldsymbol{y}_{t,i}}^T \bar{\boldsymbol{y}}_{t,i}^k}{\|\boldsymbol{y}_{t,i}\| \left\|\bar{\boldsymbol{y}}_{t,i}^k\right\|}, k = 1, 2, ..., N_s \tag{5}$$

$$\boldsymbol{f}_t(i) = \begin{cases} 0 & \#\{Siml(\boldsymbol{y}_{t,i}, \bar{\boldsymbol{y}}_{t,i}^k) > D_{TH}\} > N_{\min} \\ 1 & others \end{cases} \tag{6}$$

where $\boldsymbol{f}_t \in \mathcal{R}^{N_b}$ is the foreground block mask, and when $\boldsymbol{f}_t(i) = 1$, it indicates that the ith block of $\boldsymbol{I}_t$ is the foreground block.

The selection of threshold D_{TH}, sample number N_s and cardinality $N_{\min}$ depends on the complexity of testing videos. Further discussions of how to adjust these parameters are presented in Section 4.1.

The background model update strategy in the block level is shown in Algorithm 2. Compared to the first-in-first-out update strategy, this technique is simple and effective. Samples are discarded randomly according to a uniform probability density function with property of monotonic decay, which was demonstrated in ViBE [9].

Algorithm 2 Background update strategy in the block level

if $\boldsymbol{f}_t(i) = 0$ **then**
  $s = random(1, N_s)$
  Using $\boldsymbol{y}_{t,i}$ to replace the selected sample $\bar{\boldsymbol{y}}_{t,i}^s$ of $\bar{\boldsymbol{Y}}_t$ directly.
  Using $\boldsymbol{b}_{t,i}$ to update M_t by a learning rate α.
else
  Go to the pixel-level subtractor in the following section.
end if

3.3 Pixel-level Subtractor

Though results of the block-level subtractor are sufficient for some applications like intruder detections, many applications require higher accuracy. In this section, we will extract more details of foreground objects in the pixel level only for the foreground blocks.

To suppress more disturbances without introducing significant computational burden, we design a filter (named as integral filter) inspired by the FAST corner detection method [14]. The integral filter, shown in Fig. 2, takes the local continuity of foreground pixels into consideration, and calculates the local continuity in the way of calculating the integral image.

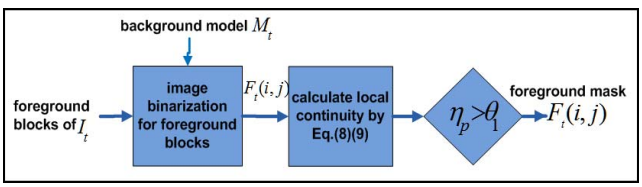


Fig. 2: Flow chart of the Integral Filter

We define a local index η to measure the local continuity of each pixel for each foreground block. For instance, a pixel $p \in \{\text{foreground blocks}\}$ is classified as foreground pixel if its local index is larger than a threshold θ_1. Here, the local continuity is defined as the total number of the candidate foreground pixels within a window around pixel p.

Firstly, we generate candidate foreground mask $\boldsymbol{F}_t$ at frame t by comparing $\boldsymbol{I}_t$ to the background model $\boldsymbol{M_t}$. Take pixel p located at (i,j) for instance, and the candidate foreground mask $\boldsymbol{F}_t$ is formulated by

$$\boldsymbol{F}_t(i,j) = \begin{cases} 1 & abs(\boldsymbol{I}_t(i,j) - \boldsymbol{M}_t(i,j)) \geq \delta \\ 0 & others \end{cases} \tag{7}$$

when $\boldsymbol{F}_t(i,j) = 1$, it implies that pixel p is a candidate foreground pixel.

Furthermore, the integral continuity value, denoted as $\boldsymbol{H}$, is calculated by (8). Note that we can conveniently obtain the local continuity η_p for pixel p as the implementation of computing integral image, which is efficient. If the size of the filter window is 3×3, then η_p can be written as (9).

$$\boldsymbol{H}(i,j) = \sum_{k \leq i} \sum_{l \leq j} \boldsymbol{F}_t(k,l) \tag{8}$$

$$\begin{aligned} \eta_p = \boldsymbol{H}(i+1, j+1) + \boldsymbol{H}(i-1, j-1) - \\ \boldsymbol{H}(i+1, j-1) - \boldsymbol{H}(i-1, j+1) \end{aligned} \tag{9}$$

Now we update the candidate foreground mask $\boldsymbol{F}_t$ to achieve a more precise foreground mask by

$$\boldsymbol{F}_t(i,j) = \begin{cases} 1 & \eta_p \geq \theta_1 \\ 0 & others \end{cases} \tag{10}$$

After processing the integral filter, more noises are eliminated and we can obtain a more specific foreground mask $\boldsymbol{F}_t$. With this final foreground mask $\boldsymbol{F}_t$, we could update the background model for the pixel-level subtractor by

$$\boldsymbol{M}_{t+1} = (1-\alpha)\boldsymbol{M}_t + \alpha(\boldsymbol{I}_t - \boldsymbol{I}_t \odot \boldsymbol{F}_t) \tag{11}$$

where α is the learning rate, $\odot$ symbol is to explicitly denote element-wise multiplication.

So far, we have described all the major steps of the proposed algorithm. Evaluations of the BCSBS algorithm will be discussed and compared with other five popular background subtraction methods in next section.

4 Experiments

In this section, we evaluate the performance of the BCSBS algorithm and compare it with other five background subtraction algorithms, including GMM1 [2], GMM2 [3], CodeBook [4], LW [6] and ViBE [9]. Although a globally accepted standardized evaluation framework is missing, receiver operating characteristic (ROC) curve is one of the most widely used metrics to assess performance for a binary classifier [15]. Background subtraction method is some kind of a binary classifier, so we employ ROC curve to evaluate the performance of different parameter combinations. The horizontal and vertical axis of ROC curve represent the probability of false alarm (PFA) and the probability of detection (PD), respectively. PFA and PD are formulated as

$$PFA = \frac{FP}{FP+TN}, PD = \frac{TP}{TP+FN} \tag{12}$$

where FP is the total number of the false positive pixel, TP is the total number of the true positive pixel, FN is the total number of the false negative pixel and TN is the total number of the true negative pixel.

Another widely used metric is percentage of correct classification (PCC) [15]. In our experiments, PCC is used to evaluate the accuracy of these comparative methods on various videos. PCC is defined as

$$PCC = \frac{TP+TN}{TP+TN+FP+FN} \tag{13}$$

Moreover, given that memory is constrained for embedded smart cameras, we also compared the proposed algorithm with others in terms of the memory requirement per frame.

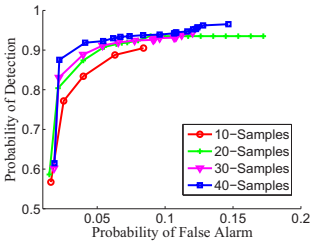
Five challenging videos are used to assess the performance of different methods. These videos are named as "office", "canoe", "traffic", "sofa" and "highway" respectively, provided by www.changedetection.net. All the experiments are implemented on 2.5GHz Core i5CPU 2GB of DDR3 either by VS2010 or by Matlab2010.

4.1 Parameter Settings

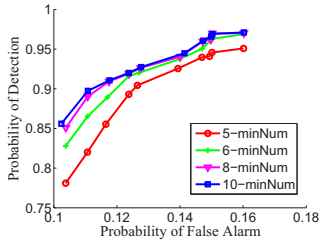
From the previous discussions, we can observe that there are three key parameters deciding the performance of the proposed algorithm:

- N_s – the total number of background samples ;
- $N_{\min}$ – the minimum number of similar block samples for selecting the candidate foreground blocks;
- m – the length of the projected vector.

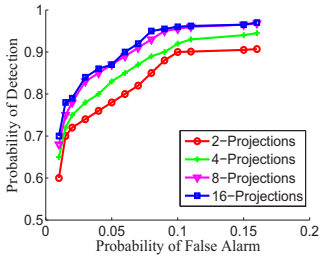
In order to determine an optimal parameter combination for the BCSBS algorithm, we analyse the ROC curve with $N_{\min}$ chosen from 2 to 80 on N_S ranging from 5 to 100. Meanwhile, others are fixed: $m = 8$, block size $= 8 \times 8$, $D_{TH} = 0.8$, $\theta_1 = 5$, $\alpha = 0.005$, $\delta = 30$ for the "office" video and $\delta = 50$ for the "canoe" video.



(a) minimum number $N_{\min}$

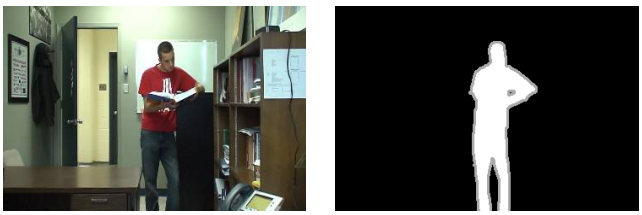


(b) background sample number N_S

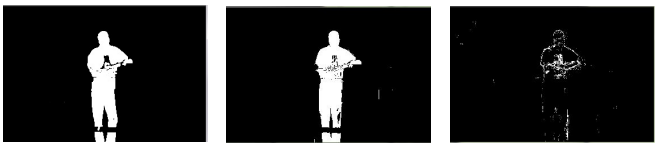


(c) length of the projected vector m

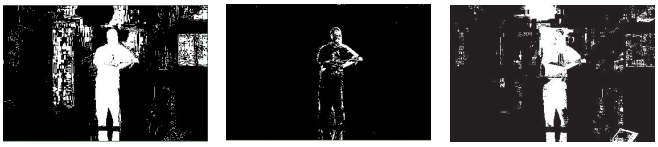
Fig. 3: ROC curve of numerous parameter combinations. (a) $N_{\min}$ ranges from 2 to N_s with step of 2. (b) N_s ranges from 10 to 100 with step of 10. (c) m is respectively set as 2, 4, 8 and 16.



(a) Frame 1498 (b) Ground Truth

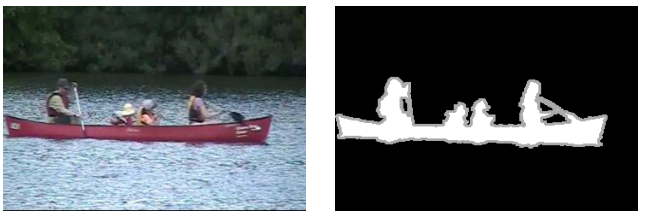


(c) Prop. method (d) GMM1 (e) GMM2



(f) CodeBook (g) ViBE (h) LW

Fig. 4: Foreground detection results of different algorithms on the "office" video at frame 1498 where a person has remained still for more than 150 frames.



(a) Frame 955 (b) Ground Truth

(c) Prop. method (d) GMM1 (e) GMM2

(f) CodeBook (g) ViBE (h) LW

Fig. 5: Foreground detection results of different algorithms on the "canoe" video at frame 955 with swaying trees and rippling water.

Since there are more than 100 combinations between $N_{\min}$ and N_S, it is not suitable to draw all of them in one ROC curve figure. Instead, we analyse the impact of different $N_{\min}$ on the ROC curve with $N_S = \{10, 20, 30, 40\}$. From Fig. 3(a), we can learn that if the number of samples is too small, the maximum probability of detection is limited, like $\{N_S = 20, PD = 0.935\}$. And when $N_S = 40, N_{\min} = 6$, the proposed method acquires an optimal parameter combination where the probability of detection is close to 95% accuracy with 4% false alarm. Note that for different background sample numbers, the best performance can be achieved with $N_{\min}$ ranging from 5 to 10. Meanwhile, it can be observed larger sample number produce higher detection accuracy. Thus we would go further to study the effect of sample number N_S on the ROC curve.

Once $N_{\min}$ has been chosen, next, we study the influence of the sample number N_S on the average performance of several videos. We draw the ROC curve with $N_{\min} = \{5, 6, 8, 10\}$ on N_S varying from 10 to 100 with step of 10. From results shown in Fig. 3(b), it can be seen that the best performance is achieved when $N_S = 40$, $N_{\min} = 6$ or 8. Also we can know that $N_{\min} = 6$ or 8 is suitable for different number of samples, while the PD increases with the number of background samples. Making the tradeoff between computational efficiency and model accuracy, we vary N_S between $\{30, 40, 50\}$ and $N_{\min}$ between $\{6, 8, 10\}$ on different videos.

Besides N_S and $N_{\min}$, the length m of the projected vector also impacts on the evaluation of the BCSBS method. We apply the same datasets to draw the ROC curve with projection number m varying from 2,4,8,16. Results shown in Fig. 3(c) indicate that when $m = 8$, the CS theory works well to capture most information of a block. Given this observation, m is fixed as 8 for evaluating the performance of the proposed BCSBS algorithm.

4.2 Evaluation Comparing with Other Techniques

After setting the parameter of the proposed algorithm, we evaluate these background subtraction methods on five typical videos mentioned above. Video of "office" is the baseline with small illumination changes. "canoe" exhibits dynamic background motion, and "traffic" was recorded with heavy camera jitter. "sofa" video contains background objects moving away, abandoned objects and objects stopping for a short while then moving away. "highway" is challenged by the swaying leaves and illumination changes.

Here, these background subtraction algorithms work on grayscale images. GMM-based methods [2][3] is tested using their implementation in the Open Source Computer Vision Library (OPENCV). ViBE is tested using the C++ implementation available on the author's paper. We implement the codebook algorithm [4] and the LW algorithm [6] ourselves by Matlab2010. The proposed algorithm is implemented by C++ and the parameter has been discussed in Section 4.1. Parameters for GMM1 [2], GMM2 [3], CodeBook [4], LW [6] and ViBE [9] are respectively chosen according to the default values of [2][3][4][6][9] and are properly adjusted in line with the testing videos.

Table 1 shows the PCC scores for different algorithms on the five datasets, and we can see that the proposed BCSBS method outperforms than others on average.

Table 1: PCC for different methods on different data

Cases	office	canoe	traffic	sofa	highway
CB	0.9073	0.8785	0.8633	0.9527	0.9561
ViBE	0.9547	0.9353	0.9090	0.9643	0.9694
GMM1	0.9601	0.9683	0.9275	0.9662	0.9525
GMM2	0.9650	0.9709	0.9507	0.9673	0.9595
LW	0.9108	0.9320	0.8995	0.9442	0.9213
BCSBS	0.9695	0.9843	0.9499	0.9726	0.9697

Moreover, Fig. 4 displays the background subtraction results of one typical frame in "office" video where a person has remained still for more than 150 frames. Compared to GMM1, CB and LW, the proposed method contains fewer false alarm pixels. In terms of GMM2 and ViBE, the whole foreground object can be extracted accurately by the proposed BCSBS method. Especially, although we use the similar update strategy in the block level with ViBE, the proposed algorithm achieves a higher PCC score than ViBE. Fig. 5 shows outputs of another challenging video "canoe" with swaying trees and rippling water. Note that the BCSBS method can eliminate more disturbances as well as acquire a higher PCC score.

We also compare the memory requirement per frame of these six methods for a 320×240 frame and results are shown by a bar grough in Fig. 6.

GMM-based algorithms require 23 to 32 bytes per pixel if one color channel and three Gaussian distributions are used. If three color channels are used, the memory required per pixel will be up to 96 bytes. Codewords-based algorithms require about 91 bytes per pixel for one color channel. And LW consumes about 7.25 bytes per pixel which is competitive to our method.

The BCSBS method needs about 5 to 8 bytes per pixel according to the complexity of different scenarios. In the block level, we carry out background subtraction by comparing a new projected vector to N_S ($N_S = 40$ on average) background projection samples to find $N_{\min}$ matches ($N_{\min} = 6$) at least. It indicates that every block requires $40 \times 8 = 320$ bytes to store samples and 8 integers for the measurement matrix $\boldsymbol{R}$. Since the measurement matrix $\boldsymbol{R}$ is regarded as a global parameter, the memory requirement of $\boldsymbol{R}$ can be omitted compared to the total memory requirement of the proposed algorithm. In another word, the memory requirement is about 5 bytes per pixel yet ViBE requires almost 40 bytes per pixel for storing background samples. Generally, foreground objects occupy a small percentage of the whole image and here we consider this percent as 50%. Thus, for the pixel-level subtractor, about 1.5 bytes per pixel is required (one byte for the foreground mask, one byte for the background model and one byte for the continuity index). In summary, the proposed method requires about 6.5 bytes per pixel.

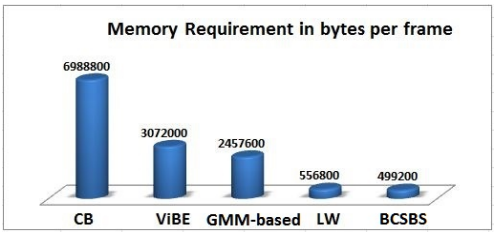


Fig. 6: Memory requirement per frame of different background subtraction algorithms using one color channel.

5 Conclusion

In this paper, we propose a light-weight yet robust background subtraction method based on block compressed sensing. The processing of each block is parallel, thus it enables high-speed parallel implementation. Since we classify blocks in the compressive sensing domain without reconstruction, the proposed method reduces memory and improves efficiency. Furthermore, the designed integral filter can deal with considerable perturbations in natural scenes, meanwhile can improve the accuracy of the methods based on the block level. In addition, a sudden illumination changes detection module is embedded into this framework easily. Numerous experiments on various challenging videos indicate that the proposed algorithm outperforms other existing background subtraction algorithms. In our future work, the proposed BCSBS algorithm can be extended to support pan-tilt-zoom cameras and smart camera networks.

References

[1] W. Hu, et al., A survey on visual surveillance of object motion and behaviors, *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3): 334–352, 2004.

[2] P. KaewTraKulPong and R. Bowden, An improved adaptive background mixture model for real-time tracking with shadow detection, in *Proceedings of European Workshop on Advanced Video-based Surveillance Systems*, 2001: 135–144.

[3] Z. Zivkovic, Improved adaptive Gaussian mixture model for background subtraction, in *Proceedings of International Conference on Pattern Recognition (ICPR'04)*, 2004: 28–31.

[4] K. Kim, et al., Background modeling and subtraction by codebook construction, in *Proceedings of International Conference on Image Processing (ICIP'04)*, 2004: 3061–3064.

[5] K. Kim, et al., Real-time foreground-background segmentation using codebook model, *Real-Time Imaging*, 11(3): 172–185, 2005.

[6] M. Casares and S. Velipasalar, Light-weight salient foreground detection with adaptive memory requirement, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'09)*, 2009: 1245–1248.

[7] M. Casares, S. Velipasalar and A. Pinto. Light-weight salient foreground detection for embedded smart cameras. *IEEE Trans. on Computer Vision and Image Understanding*, 114(11): 1223–1237, 2010.

[8] A. Manzanera, $\Sigma-\Delta$ background subtraction and the Zipf law, in *Proceedings of Pattern Recognition, Image Analysis and Applications*, 2007: 42–51.

[9] O. Barnich and M. Van Droogenbroeck, ViBe: A universal background subtraction algorithm for video sequences, *IEEE Trans. on Image Processing*, 20(6): 1709–1724, 2011.

[10] D.L. Donoho, Compressed sensing, *IEEE Trans. on Information Theory*, 52(4): 1289–1306, 2006.

[11] L. Gan, Block compressed sensing of natural images, in *Proceedings of Conference on International Conference on Digital Signal Processing (ICDSC'07)*, 2007: 403–406,

[12] F. Sebert, Y.M. Zou and L. Ying, Toeplitz block matrices in compressed sensing and their applications in imaging, *International Conference on Information Technology and Applications in Biomedicine (ITAB'08)*, 2008: 47–50.

[13] H. Rauhut, Circulant and Toeplitz matrices in compressed sensing, in *Proceedings of Signal Processing with Adaptive Sparse Structured Representations (SPARS'09)*, 2009.

[14] E. Rosten and T. Drummond, Machine learning for high-speed corner detection, in *Proceedings of the 9th European Conference on Computer Vision (ECCV'06)*, 2006: 430–443.

[15] Y. Benezeth, et al, Review and evaluation of commonly-implemented background subtraction algorithms, in *Proceedings of IEEE International Conference on Pattern Recognition (ICPR'08)*, 2008: 1–4.