Real-Time and Robust Compressive Background Subtraction for Embedded Camera Networks

Yiran Shen, *Member, IEEE*, Wen Hu, *Senior Member, IEEE*, Mingrui Yang, *Member, IEEE*, Junbin Liu, *Member, IEEE*, Bo Wei, *Member, IEEE*, Simon Lucey, *Senior Member, IEEE*, and Chun Tung Chou, *Member, IEEE*

Abstract—Real-time target tracking is an important service provided by embedded camera networks. The first step in target tracking is to extract the moving targets from the video frames, which can be realised by using background subtraction. For a background subtraction method to be useful in embedded camera networks, it must be both accurate and computationally efficient because of the resource constraints on embedded platforms. This makes many traditional background subtraction algorithms unsuitable for embedded platforms because they use complex statistical models to handle subtle illumination changes. These models make them accurate but the computational requirement of these complex models is often too high for embedded platforms. In this paper, we propose a new background subtraction method which is both accurate and computationally efficient. We propose a baseline version which uses luminance only and then extend it to use colour information. The key idea is to use random projection matrics to reduce the dimensionality of the data while retaining most of the information. By using multiple datasets, we show that the accuracy of our proposed background subtraction method is comparable to that of the traditional background subtraction methods. Moreover, to show the computational efficiency of our methods is not platform specific, we implement it on various platforms. The real implementation shows that our proposed method is consistently better and is up to six times faster, and consume significantly less resources than the conventional approaches. Finally, we demonstrated the feasibility of the proposed method by the implementation and evaluation of an end-to-end real-time embedded camera network target tracking application.

Index Terms—Object tracking, real-time performance, embedded camera networks, background subtraction, compressive sensing, Gaussian mixture models

✦

1 INTRODUCTION

EMBEDDED camera networks, which consist of video sensors distributed in a spatial environment, can be used for security surveillance, environmental monitoring and many other applications. An important service that can be provided by embedded camera networks is *real-time target tracking*. The first step in target tracking is to extract the moving targets from the video frames, which can be realised by using background subtraction. Robust background subtraction is typically the dominant consumer in resources. The aim of background subtraction is to detect whether the foreground is present in a newly acquired video frame. This is usually realised by using the knowledge of earlier video frames to learn a background model, and then applying

statistical tests to decide whether the newly acquired frame is different from the background. A challenge for background subtraction is to differentiate between the foreground and subtle changes in the background, caused by events like illumination changes or moving tree branches. Moreover, a new challenge arises when background subtraction is to be used in embedded camera networks. The background subtraction algorithm must be computationally efficient due to resource constraints of embedded platforms. In this paper, we present a new background subtraction method which is both accurate and computationally efficient.

As mentioned earlier, one challenge for background subtraction is to differentiate the foreground from subtle changes in the background. This problem can be solved by modelling the background by some complex statistical models, such as, kernel density [14], [20] or Gaussian density [15], [16], [31], [36] models. Background can also be approximated by non-statistical models, such as codebook based method [18], [19] and sample based methods such as Vibe [2]. Background subtraction can be also formulated as a sparse error recovery problem [11] or low rank matrix estimation problem [8]. Although these recent background subtraction methods are highly accurate, the use of complex models means they require high computation resources. Fig. 1 depicts the design space for background subtraction algorithms where an optimal algorithm should be accurate and have low computation cost. According to [5], the mixture of Gaussians (MoG) [31] model achieves the best trade-off between the accuracy and computation among the state of the arts and Fig. 1 depicts the

- *Y. Shen is with College of Computer Science and Engineering, Harbin Engineering University, No. 145 Nantong Street, Harbin, China, 150001. E-mail: shenyiran@hrbeu.edu.cn.*
- *M. Yang is with the Autonomous Systems Lab, CSIRO ICT Centre, QCAT Technology Ct, Pullenvale, QLD, Australia 4109.*
- *J. Liu is with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Queensland, Australia.*
- *S. Lucey is with the Robotics Institute, 5000 Forbes Ave, Pittsburgh, PA 15213.*
- *W. Hu is with the ICT Centre, CSIRO, Australia, 1 Technology Court, Pullenvale, Queensland, Australia 15213.*
- *B. Wei and C.T. Chou are with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia 2052.*
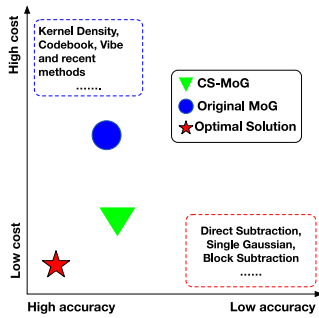
Fig. 1. Design space and related work.

"position" of MoG in the design space. Although MoG is accurate, our evaluation in [27] shows that MoG can only processes three frames per second on the embedded platform with low image resolution. In this paper, we propose to resolve the tension between accuracy and computation cost by using random projection matrix to reduce the dimensionality of the problem. The result is a background subtraction method (called CS-MoG) which has comparable accuracy compared to MoG but with significantly lowered computation time, see Fig. 1.

The use of random projection matrix to produce "compressed" projections is inspired by the recent applications of information theory of compressive sensing [7], [12]. Random projection matrices are used to improve the efficiency of the sampling and reconstruction in compressive sensing. In this paper, we use random projection matrices to reduce the dimensionality of the problem of background subtraction while perserving the accuracy. The contributions of this paper are four folds:

- We propose a novel background subtraction method that uses random projection matrices (for dimensionality reduction) and MoG (for accuracy). Since our proposed method uses "compressed" samples, we will refer to it as CS-MoG where CS stands for *Compressed Samples*. We introduce two versions of CS-MoG. The baseline version of CS-MoG uses only luminance information and will be simply called CS-MoG. We also propose an extension that uses colour space and call it *Colour Space Compressed Sampling* (CoSCS) -MoG (which uses all three colour channels).
- We show, by using multiple real world datasets, that the accuracy of CS-MoG is comparable to MoG and is significantly more accurate than a number of other background subtraction methods. Further evaluations show that CoSCS-MoG improves the accuracy of CS-MoG significantly and is equally or more accurate compared to the traditional MoG.
- In order to show that the computational efficiency of our proposed methods is not platform specific, we implement our methods and MoG on two embedded platforms: *Blackfin* (which is used in previous paper [27]) and *PandaBoard*. The results show that our approaches are up to 6 times faster than the original MoG.
- We implement and evaluate an end-to-end multiple camera object tracking application based on our proposed algorithm, which demonstrates the feasibility of our approach to operate in real-time scenario on embedded platforms.

The organisation of this paper is as follows. We first provide the related work in Section 2. In Section 3, we provide technical background on the MoG for background subtraction. We then present and evaluate our proposed method CS-MoG and its extension, CoSCS-MoG in Section 4. In Section 5, we present results on running our compressive background subtraction approaches on pandaboard. At last, Section 6 concludes the whole paper.

2 RELATED WORK

In this section, we review and discuss the recent related works. Because most of the recent applications of random projection matrix come with compressive sensing, we review the papers related to the compressive sensing and background subtraction.

2.1 Background Subtraction

MoG [31] has been one of the most popular background subtraction techniques in computer vision because of its robustness to subtle illumination changes. Its bottleneck is its computational intensity because of the need to compute and update the Gaussian mixtures. Instead of using a fixed number of Gaussian mixtures as in [31], the work in [40] adaptively determines the number of Gaussians for each pixel. This results in a more computationally efficient procedure. The comparison in [40] shows that the adaptive procedure is 2−30 percent faster compared with original MoG. However, our experiments in Section 5.1 show that our proposed approaches are up to six times faster than the original MoG on different embedded platforms. Another example of improving the efficiency of MoG is in [32]. In this work, it simplifies the learning update of the Gaussian mixtures and instead of using $\frac{\omega}{\sigma}$ (where ω and σ are respectively the weight and standard deviation of a Gaussian distribution within a Gaussian mixture) to order the Gaussians, it simply uses ω. These simplifications can decrease the computation time of MoG by 1.6 times. However, these simplifications may not be suitable for some situations because the reorder condition is only based on ω, which may increase or decrease so quickly that a slowly moving target may be mistakenly incorporated into background or removed from the current background model. Our proposed approaches perform foreground detection in two stages. The first stage includes a block-based foreground detection step and subsequently pixel-based foreground refinement step. Block-based methods are classical and known for their efficiency in background subtraction [26]. This block-based background subtraction method divides a frame into 8 × 8 blocks and then computes a feature vector with eight elements. For foreground detection, it uses a block-scale background training set for comparison using normalised vector distance as the criterion. The method can be used as an assistant to traditional pixel based background models like MoG to increase the accuracy. However, its accuracy is related to the size of the training set. Also, the algorithm has to traverse the whole dataset to find the closest fit. When the dataset is large, the algorithm will be computationally intensive. However, our proposed method does not require any training set. Some more advanced background subtraction models have been proposed recently [2], [8], [11], [18], [19]. Vibe [2] is a recently

proposed background subtraction method which is highly accurate. Its key idea is to use past samples of a pixel to represent the background instead of using a probability density model. However, it is too computationally expensive to be implemented on embedded systems.

2.2 Applications of Compressive Sensing

Compressive sensing has been an active research area recently. There is a lot of existing work in the area and we will limit this review to the work related only to computer vision and sensor networks. One of the most important breakthroughs in computer vision with compressive sensing is the invention of compressive sensing based imaging hardwares. Two examples are the single pixel camera [13] and the random convolution camera [17], [25]. These cameras exploit the theory of compressive sensing. Instead of sensing pixel-wise, they use projections as the measurements. As a result, the sensing requirements of these cameras are significantly lower. Other research work in using compressive sensing in computer vision includes the CSBS background subtraction algorithm [9], object tracking and localisation [38] and 3-D reconstruction [10], [24].

Sensor networks are resource constraint while compressive sensing is capable of significantly reducing the sensing rate and data dimension. Research is done on applying compressive sensing and a closely related idea, sparse representation in sensor networks. Compressive sensing can be used as an efficient sensing strategy [28], [29], [37]. In [37], the authors considered the problem of monitoring soil moisture with wireless sensor networks. With compressive sensing, they achieve high accuracy at no more than 10 percent of the traditional sensing rate. Also the sparse representation is attracting increasingly attentions with compressive sensing. One of the examples is [22]. In this paper, the authors deal with cross-correlation problem (which is widely used in sensor networks) efficiently via sparse representation. Another example is [34] which applies sparse representation for the activities recognition using radio frequency interference. The use of random projections to reduce the amount of computation in embedded systems has also been investigated in [30], [35], [39].

3 Technical Background

In order to make this paper self-contained, this section provides technical background on background subtraction using MoG.

In [31], the authors proposed to use a MoG to model the background in background subtraction. In this method, the history of each pixel is modelled by a MoG consisting of K (typically chosen to be 3-5) Gaussian distributions. When a new video frame is presented, each pixel is compared with the MoG model for the corresponding pixel. If the new pixel value is within 2.5 standard deviation of any one of the K Gaussian distributions making up the MoG, then the pixel is considered a background candidate. A background candidate, afterwards, should be checked if it belongs to a background distribution. The MoG for each pixel is updated for each frame. This update allows MoG to adaptively deal with noise and illumination changes which fixed threshold cannot handle.

The updating of the MoG model for a pixel is as follows. At time t, the MoG of each pixel consists of K Gaussian distributions. The kth ($1 \le i \le K$) Gaussian is assigned a weight of $\omega_{k,t}$. If the new pixel value does not match any of the K Gaussians, the least probable distribution will be replaced by a new distribution with high variance and low weight; otherwise, the weight for the kth Gaussian is updated as:

$$\omega_{k,t+1} = (1 - \alpha)\omega_{k,t} + \alpha(G_{k,t+1}), \qquad (1)$$

where α is the learning rate and $G_{k,t+1}$ is a binary variable whose value is 1 if the kth Gaussian matches the new pixel and is zero otherwise. If the new pixel value x_{t+1} at time $t + 1$ is accounted by, say the kth, Gaussian distribution, its mean $\mu_{k,t}$ and variance $\sigma^2_{k,t}$ will be updated as:

$$\mu_{k,t+1} = \gamma x_{t+1} + (1 - \gamma)\mu_{k,t} \qquad (2)$$

$$\sigma^2_{k,t+1} = \gamma(x_{t+1} - \mu_{k,t+1})^2 + (1 - \gamma)\sigma^2_{k,t}, \qquad (3)$$

where

$$\gamma = \frac{1}{\sqrt{2\pi}\sigma_{k,t+1}} \exp\left(-\frac{(x_{t+1} - \mu_{k,t+1})^2}{2\sigma^2_{k,t+1}}\right). \qquad (4)$$

The probability that one of these K Gaussian distributions is the current background model is determined by the ratio of $\omega_{k,t}/\sigma_{k,t}$ at the current time t. When a new pixel value is available at time $(t + 1)$, it will be checked if it belongs to any of the K distributions. If a new pixel does not match any of the K distributions, the least probable distribution will be replaced by a new distribution with mean equals to the new pixel value, and initial variance and weight. If the new pixel value matches any one of the K Gaussian distributions that models the pixel, the parameters of these distributions should be updated. After updating, the current K Gaussian distributions are sorted using the updated $\omega_{k,t+1}/\sigma_{k,t+1}$. According to this ratio and the prior information about the portion of the pixels accounted for the background, the number of background distributions in these K distributions is decided as,

$$N_b = \arg\min_n \left(\sum_{k=1}^{n} \omega_k > T_{h_b}\right), \qquad (5)$$

where T_{h_b} is the portion of pixels accounted for by the background. This equation means that the first N_b distributions are chosen as the current background model. Therefore, the current pixel values that are located within 2.5σ of these N_b distributions will be marked as background. With the multi-modal distributions, MoG can accommodate multiple background scenario well.

4 Compressive Background Subtraction

In this section, we will present a novel method for background subtraction and its colour space extension. Both approaches use random projection matrix for dimensionality reduction and then apply MoG to the reduced dimension data for foreground detection. The difference between the two approaches is that CS-MoG uses only the luminance (or
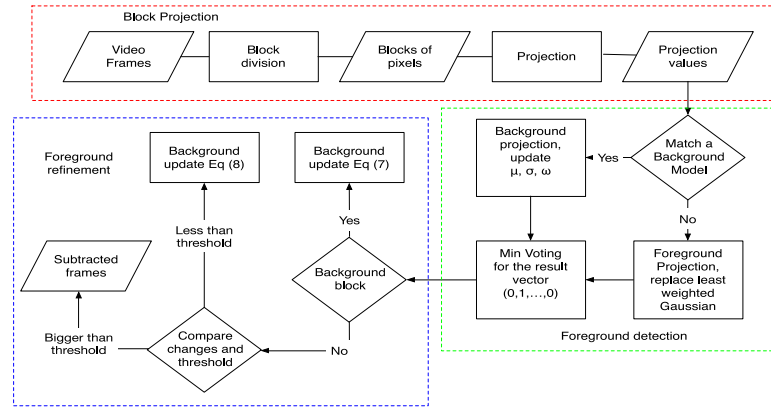
Fig. 2. Flow chart of the Algorithm. Different components are grouped by dashed frames in different color.

grey scale) while CoSCS-MoG uses full colour information. We evaluate both approaches on multiple datasets and show both of them are robust while CoSCS-MoG achieves substantially better performance than CS-MoG.

4.1 CS-MoG

The aim of this part is to describe CS-MoG, which is an accurate but yet computationally efficient background subtraction method. The MoG background subtraction method is able to deal with subtle changes and non-static/moving backgrounds (e.g., rain, moving tree branches or fluttering leaves) because it models each pixel by a mixture of 3-5 Gaussian distributions. However, this also makes MoG computationally intensive to be used on embedded platforms. In order to simultaneously realise accuracy and computational efficiency, we propose CS-MoG. CS-MoG uses random projection matrix to reduce the dimensionality of the data while retaining much of the information. We then apply MoG to the reduced dimension data for background subtraction. This section is divided into three parts. We describe the steps of CS-MoG in Section 4.1.1, justify the use of Gaussian mixture models for reduced dimension data in Sections 4.1.2 and 4.1.3 and evaluate its performance in Section 4.1.4.

Fig. 2 shows the flow chart of the CS-MoG algorithm. Each dashed line box in the flow chart corresponds to a step of the algorithm. We will now describe each step in details.

4.1.1 Steps of CS-MoG

Our method is divided into three steps. In the first step, the image is segmented into blocks of 8×8 pixels. (Note: We have experimented with different block sizes and found 8×8 blocks gave the best results. We therefore assume a block size of 8×8 throughout this paper. The default block size in JPEG is also 8×8.) Projections are then computed for each block. In the second step, each projection value is modelled as a MoG to determine if the block contains some part of the foreground. We then fuse the results from all the projection values from a block to determine if it is a background or foreground block. The pixels in a background block are all background but the pixels in a foreground block can include both background and foreground. We call the second step foreground detection. The third step, which will be referred to as foreground refinement, is to identify which pixels in a foreground block is foreground. Thus, at the end of these

three steps, each pixel in the image is labelled either as a foreground or background.

Block projections. Prior to performing background subtraction, we carry out a pre-processing step where we convert the video frames from RGB format to grey scale or intensity. This pre-processing step applies to all the background subtraction methods in Section 4.1.

After pre-processing, we start the background subtraction process. The first step of CS-MoG is to divide the image into blocks of 8×8 pixels. After that, for each block, we form a 64×1 vector of the pixel values in a block and compute random projections of the vector. The projection matrix we utilise is randomly generated at the beginning of a video sequence. Once it has been generated, the same projection matrix is used for each block for the entire video.

We consider two types of projection matrices, which we will call unbalanced matrices and balanced matrices. Each element of an unbalanced projection matrix is generated by a symmetric Bernoulli distribution of ± 1. A balanced projection matrix also consists of ± 1 at equal probability but in addition we require that each row must contain equal number of 1's and -1's. Therefore, the sum of the elements in each row of a balanced matrix is always zero. We choose Bernoulli distribution instead of Gaussian distribution because the embedded platforms we use in the experiment part are slow for floating point computation. We will refer to CS-MoG that uses a balanced matrix (resp. unbalanced matrix) as CS-MoG-Balance (CS-MoG-Unbalance). In particular, we will show experimentally and analytically that CS-MoG-Balance gives better performance.

Since the operations to be carried out on each block of 8×8 pixels are identical, we will describe the operations on one block. We first stack the pixel values of the $n = 64$ pixels in a block into a $n \times 1$ vector that we call x. We assume that a $m \times n$ projection matrix Φ (which can be balanced or unbalanced) has also been generated. Note that m, which is the number of projection vectors, is a design parameter which we will study later on in Section "Impact of Number of Projections on the ROC for CS-MoG Balance". We compute the projection:

$$y = \Phi x. \tag{6}$$

The $m \times 1$ vector y contains the m projections values for this block. Given that the vector x (which contains the pixel values in a block) is compressible (Note: We know from image

compression that the pixels in a block is compressible in DCT or wavelet basis [21].), we expect from compressive sensing that, for properly chosen value of m and projection matrix Φ, the m projection values in y contain almost all the information in x. This means that one can use the vector y, instead of x, to decide whether the block is foreground or not. Furthermore, we expect $m \ll n$ which means that we will be working with data of lower dimension. In fact, the results in Section "Impact of Number of Projections on the ROC for CS-MoG Balance" show that $m = 8$ projections per block give good accuracy for background subtraction.

Foreground detection. After computing the projections for each block, we need a method to determine whether this block contains some part of foreground according to projection values. To build a robust decision, we use MoG for the foreground detection step.

Our experimental evaluation of MoG (see Section 5) shows that, MoG can only process three frames per second on two different embedded platforms. In fact, the experiments in Section 5.1 shows that the Gaussian mixture computations take up almost all the processor resources. Therefore, computation efficiency can be improved if we can reduce the total number of Gaussian distributions that we use per frame.

Our main idea is to model *each projection value* by a mixture of K Gaussian distributions, which is similar to what MoG does for each pixel. (The choice of the parameter K will be discussed in Section 4.1.2.). Furthermore, we model each projection value independently and do not consider possible correlation between them. We note that the projection values are *not* independent of each other and we make the independence assumption because of the limited computation resources on the embedded platform. This is exactly the same consideration in the original MoG paper [31] where the different colour channels are treated as independent to avoid the costly matrix inversion. Since we aim to propose a real-time background subtraction method which is to be implemented on the embedded systems. The computation complexity is an important consideration to achieve the realtime processing. Modelling the projections as multivariate MoG by considering the dependence between the projections is not applicable on embedded systems. We show in Section 4.1.4 that the independence assumption does not have severe impact on background subtraction accuracy. Following MoG, we consider a projection value to be a *background projection value candidate* if it is within 2.5 standard deviations of one of the K Gaussian distributions that models the projection value; otherwise it is a *foreground projection value.*

It is likely that the result of applying MoG to the m projection values will result in a mixture of background and foreground projection values. We therefore need a method to fuse the results. A number of fusion strategies are possible. Let us assume that the MoG test results in f foreground projection values out of all m projection values in block. We evaluated three fusion strategies: (1) Majority voting: the block is foreground if $f > \frac{m}{2}$; (2) Max voting: the block is foreground if $f = m$; and (3) Min voting: the block is foreground if $f \neq 0$, i.e. $f \geq 1$. Our evaluations

(not shown here due to lack of space) show that min voting gives the best result. In the following, we will assume min voting is used.

Foreground refinement. The foreground detection step so far works on the resolution of a block. This resolution may be sufficient for some applications but sometimes it is desirable to work with resolution at pixel level. We show how we can do that in this foreground refinement step.

We will assume that if a block is classified as the background, then all pixels in the block are background pixels. However, we cannot do the same for a foreground block. It is possible for a foreground block to contain both foreground and background pixels. This is especially true for those foreground blocks lying at the edge of the foreground. This means that we only need to work further on the foreground blocks. Since a video frame is expected to consist mainly of background blocks, the number of foreground blocks that we need to work with is likely to be small.

In order to determine which pixels in a foreground block is in fact foreground without introducing significant computation burden, we build a simple background learning strategy for each block. Our pixel-scale background learning method can be described as follows. If a block X_{t+1}^b is marked as the background, then its pixel values are incorporated into the current background model B_t (at time t) of the block where B_t is a n-by-1 vector whose elements are learned from the corresponding historical pixel values by using a learning rate α:

$$B_{t+1} = \alpha X_{t+1}^b + (1 - \alpha)B_t, \qquad (7)$$

where B_{t+1} is the updated background 'image' block. The first image of the video frames is treated as the initial background model B_1.

If the block X_{t+1}^f is marked as a foreground, then the background model B_t of this block will be updated with a background mask M_{t+1}

$$B_{t+1} = \left(\alpha X_{t+1}^f + (1 - \alpha)B_t\right)_{M_{t+1}}. \qquad (8)$$

The background mask consists of indices of pixels which satisfy the following condition:

$$M_{t+1} = \text{Index}\left[|X_{t+1}^f - B_t| < \delta\right], \qquad (9)$$

where δ is the threshold that accommodates a certain extent of noise and illumination change. The background "image" B_t is used as the reference to obtained the indices of the background pixels. This threshold is specified by the applications. For the datasets used in this paper and the experiments, a threshold around 10 (the pixel value is between 0 to 255) is suitable. A larger threshold should be applied if the illumination change is more severe. The function of the background mask is to prevent the foreground mistakenly being incorporated into the background model so that the background model for the foreground pixels at time $(t + 1)$ will remain the same as that at time t. With these background model update processes, we are able to realise pixel-level background subtraction.

The foreground refinement in Eq. (9) models each of the pixels as a single Gaussian model to accommodate the
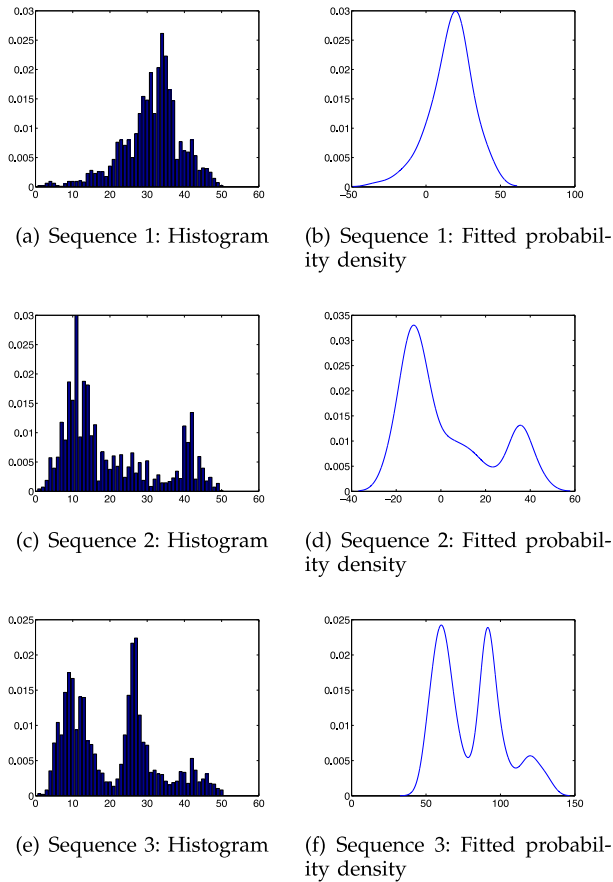
(a) Sequence 1: Histogram

(b) Sequence 1: Fitted probability density

(c) Sequence 2: Histogram

(d) Sequence 2: Fitted probability density

(e) Sequence 3: Histogram

(f) Sequence 3: Fitted probability density

Fig. 3. This figure shows distributions of three sequences of projection values. The x-axis is the projection value. The figure on the left shows the histogram, i.e., y-axis is the relative frequency of each bin. Figures on the right show the fitted probability density.

TABLE 1
Number of Gaussians Required for Approximating
Distribution of Projection

	One	Two	Three	More
Datasets 1	99.94 %	0.06%	0%	0%
Datasets 2	89.58%	10.06%	0.35%	0.01%
Datasets 3	76.28%	19.12%	3.65%	0.095%

discussion, we consider a generic 8×8 block and we assume the value of ith pixel x_i ($1 \leq i \leq 64$) is modelled by a random variable X_i. The projection value v is therefore a random variable $V = \sum_{i=1}^{64} \beta_i X_i$ where $\beta_i = \pm 1$ are the elements of a generic projection vector that we use in CS-MoG.

Let us assume that each random variable X_i is a mixture of Q Gaussians. Since the probability distribution of a sum of random variables is equal to the convolution of the probability distributions of the random variables, it can be shown that the random variable V is still a mixture of Gaussians. However, the number of Gaussians needed can be as large as Q^{64}. The use of such a large number of Gaussians is certainly not practical. We therefore investigate whether it is possible to use a small number of Gaussians to model a projection value.

For our investigation, we use three datasets, or video sequences. (The details of these datasets are described in Section "Goals, Metrics and Methodology".) The same method of investigation is applied to all three datasets. Consider a video sequence consisting of F frames, indexed by $f = 1, \ldots, F$. Each frame consists of B 8×8-block, indexed by $b = 1, \ldots, B$; it is assumed that the bth block is always located in the same location within a block. We then generate P projection vectors, which are indexed by $p = 1, \ldots, P$. By computing the projection values of the pth projection vector with the bth block in frames $f = 1, ..., F$, we obtain a sequence of F projection values. We do this for each combination of p and b, giving altogether BP sequences of projection values per dataset. Fig. 3 shows the histograms and fitted probability densities of three different sequences of projection values. Visual inspection suggests that these three sequences can be modelled by a probability distribution with one to three Gaussians.

In order to systematically determine the number of Gaussian distributions needed to approximate a sequence of projection values, where we use the kernel density estimation method in [6] to calculate the number of Gaussians required. We do this for all three video sequences and the results are shown in the Table 1. The results show that, for each dataset, over 99.9 percent of the projection value sequences can be modelled by no more than three Gaussians. We therefore choose the value of K to be 3 in our CS-MoG method. Note that this is a very encouraging result, especially if we want to implement our CS-MoG method on embedded platforms. Consider the original MoG where each pixel is modelled by three Gaussians, which means we need 64×3 Gaussians per block. For our CS-MoG, we show later that eight projections per block is sufficient; since each projection value needs three Gaussians, the number of Gaussians needed per block is 8×3, which is a reduction by a factor of 8.

lighting change. Although it is simple, our experiment results demonstrate that this simple method has good accuracy, especially because of good accuracy of our proposed foreground block detection method. We do not apply MoG for the foreground refinement as we will lose the computation efficiency obtained from the foreground block detection stage if we use MoG to maintain the background models for every pixel in both foreground and background blocks. However, the performance of foreground refinement relies on the accurate foreground block detection. Therefore, we demonstrate accuracy of both block-wise and pixel-wise results in Section 4.1.4.

4.1.2 Parameter Choice for CS-MoG

The CS-MoG method that we have described comes with a number of different parameters. Two key parameters are the number of projections per block and the number of Gaussians K to model a projection value. We will study the number of projections later on in Section 4.1.4. In this section, we look at the choice of the parameter K which is the number of Gaussian distributions being used to model a projection value. In particular, we will show that a small value of $K = 3$ is needed. This is very encouraging because a small K means less computation burden.

Given that the same set of operations are applied to all blocks, it is sufficient to consider a block and the projection value obtained by one projection. Therefore, for this

TABLE 2
This Table Shows the Number of Gaussians Required
to Approximate the Subspace Components

	One	Two	Three	More
CS	94.75 %	5.17 %	0.079%	0%
PCA	89.62%	3.92%	5.88%	0.59%
DCT	89.15%	4.36%	5.91%	0.57%

It is calculated by the kernel density estimation tool. This shows that DCT and PCA transformation produces more complex distributions than random projections.

4.1.3 Comparision with other Subspace Analysis Methods

In order to justify the use of random projections in background subtraction, we compare random projections against other subspace analysis methods.

A key idea behind CS-MoG is that, instead of testing whether a pixel is the background, it performs the statistical test on the projection values of a block of pixels. The projection in CS-MoG is carried out by a random Bernoulli matrix and given that projection is a mapping from one subspace to another, a question is whether other linear mappings may perform better. Here we choose two well known linear mappings for comparison: PCA and 2-D DCT.

Following the parameter choice discussions above, we divide each frame of the test video into 8×8 blocks and compute the grayscale intensity. CS-MoG then generates foreground detection test statistics (i.e., the projection values) by using a Bernoulli projection matrix. We now describe two alternative background subtraction algorithms, PCA-MoG and DCT-MoG, which are based on MoG, but with foreground detection test statistics generated by PCA and DCT. For PCA-MoG, we use a number of video frames which contain only the background as the training set to calculate PCA transform basis. This transform basis is then used to compute the principal values (the test statistics) for the current video frame for foreground detection. For DCT-MoG, the test statistics are the dominant 2-D DCT coefficients.

In Section 4.1.2, it is shown that, for CS-MoG, the statistical distribution of the projection values can be well approximated by Gaussian mixtures with at most three Gaussians. This significantly reduces the number of Gaussians that CS-MoG has to track and results in a much faster background subtraction algorithm. A similar evaluation is shown in Table 2. The results shown in Table 2 is the average over the three datasets in Section 4.1.2. The table shows that, DCT and PCA transformation lead to more complex distributions than that of random projections. Therefore, CS-MoG may obtain better performance because of its lower complexity. To validate our hypothesis, we test the background subtraction performance of these three algorithms. We carry out six tests. In test i ($i = 1, \ldots, 6$), we use i Gaussians to model each test statistic of the three algorithms. Therefore, $i = 3$ corresponds to CS-MoG. Fig. 4 shows how the number of false detections varies with i for the three algorithms. For all three algorithms, the figure shows that there is no significant performance improvement when $i > 3$. It also shows that CS-MoG outperforms both PCA-MoG and DCT-MoG by a good margin.
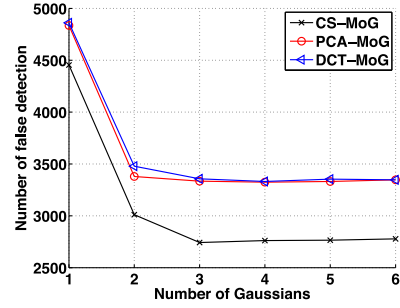

Fig. 4. This figure shows the performance of CS-MoG, PCA-MoG and DCT-MoG. The x-axis shows the number of Gaussians used to model each test value for background detection. The y-axis gives the total number of false classifications.

Why random projections works? As stated in many papers related to compressive sensing, the random projections preserve the information of the high-dimensional signals with significantly lower dimension [1]. Another theoretical insight is derived from the dimensionality reduction of random projections. It is well known that generative classifiers like MoG suffer from the *curse of dimensionality* [4] especially when it is extended into colour space. This phenomenon arises when analysing data in high-dimensional space. It is problematic for methods requiring statistical significance. Since random projection matrix reduces the dimensionality, the *curse* is diminished.

At last, as our obseravation, eight projections are not sufficient to accurately recover an image block that contains 64 pixels (or 192 values in color space). However, CS-MoG uses significantly smaller number of projections than that required by accurately recovering the original signal with ℓ_1 reconstruction. This is due to the fact that CS-MoG does not require the recovery phase. Therefore it does not need to satisfy the special requirement of ℓ_1 reconstruction on the number of projections. Instead, it only needs to satisfy the requirement of ℓ_0 reconstruction (i.e., the number of of projections $M_{\ell_0} \geq 2H$, where H is the sparsity of the image block) which is a much relaxed condition [3]. This feature significantly reduces the dimensionality of data so as to boost the processing time of the algorithm and diminish the *curse of dimensinality* .

4.1.4 Performance Evaluation

Goals, metrics and methodology. The goals of our evaluation is to demonstrate whether CS-MoG 1) achieves the best subtraction accuracy among MoG-based efficient background subtraction algorithms and 2) obtains a better capability to deal with illumination change especially with balanced matrix.

We use four datasets to evaluate the performance of various background subtraction algorithms. The first dataset (dataset 1) is a private dataset from our laboratory for monitoring a footpath. Datasets 2 and 3 are, respectively, VS-PETS'2003 and PETS'2001, from http://www.cvg.rdg.ac.uk/. Dataset 2 is on a football match while dataset 3 is from monitoring people and vehicles outdoor. Dataset 4 is Perception Sequence from monitoring shopping mall (http://perception.i2r.a-star.edu.sg/). We use 400 consecutive video frames from each dataset and the foreground is annotated by a mask. Dataset 1 monitors the outdoor footpath in our
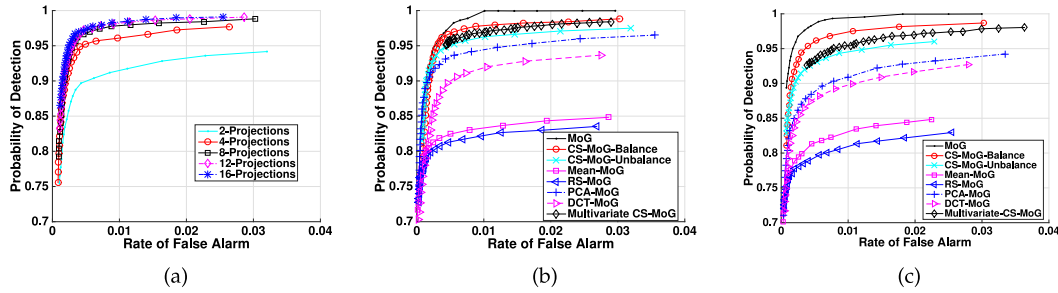
Fig. 5. ROC curves.

institute. There are illuminition changes and moving backgrounds (waving of trees by the wind). Dataset 2 monitors a football match. There are no pure background frames and foregrounds are moving quickly in the video. In dataset 3, people and vehicles are recorded in an outdoor environment. There are no pure background frames. The change of the illumination is significant and moving backgrounds exist in the video. The dataset 4 provides the evaluation for indoor environment. (We do not demonstrate the results from datasets 2 and 3 due to the space limit. But you can refer [27] for the details.)

Given that the spirit of CS-MoG is to use projections to reduce the dimensionality of input to MoG, we consider PCA-MoG, DCT-MoG as well as two other methods to realise reduction in dimension but not using projections. The first method is called Random sensing MoG (RS-MoG). RS-MoG is identical to CS-MoG except that RS-MoG does not compute m projections. Instead, RS-MoG uses m random pixels in a block to decide whether that block is foreground or not. The second method is called Mean-MoG. For Mean-MoG, we divide each block into m sub-blocks and compute the mean pixel values of each sub-blocks. These mean values are input to MoG for foreground detection. Note that CS-MoG, PCA-MoG, DCT-MoG, RS-MoG and Mean-MoG use, respectively, m projections, m dominant coefficeints, m pixel values and m mean values per block for foreground detection. The dimensionality reduction for these methods are therefore identical.

In this paper, we regard a block/pixel in the foreground (resp. background) as a positive (negative) event. A false positive means a genuine background block/pixel is incorrectly detected as a foreground. We express the performance of various methods by using the receiver operating characteristic (ROC) curve. The vertical axis of the ROC curve is the probability of detection (P_D) which is the total number of true positives divided by the number of foreground blocks/pixels (positive events) in ground truth. The horizontal axis of the ROC curve is the rate of false alarm (F_A) which is the number of false alarms (or false positives) divided by the number of background blocks/pixels (or negative events) in ground truth. The results involving random projections are obtained from 30 independent trials with different projection matrices.

Impact of number of projections on the ROC for CS-MoG balance. As the major contribution of CS-MoG is the foreground block detection based on random projections. We evaluate the impact of the number of projections on the ROC for CS-MoG-Balance on foreground block detection. We apply CS-MoG-Balance to 400 consecutive video frames

in each dataset. For each dataset, we use the following number of projections: 2, 4, 8, 12 and 16. The results for the dataset 1 are shown in Fig. 5a. We can make two observations from this figure. Firstly, the performance of foreground blocks detections improves with an increasing number of projections. Secondly, the performance improvement diminishes when eight or more projections are used. (The results of the other two datasets demonstrate the same behaviour as shown in [27].) These observations can be explained by the fact the amount of information increases with the number of projections. However, most of the information in a block can be captured by eight projections. Given these observations, we will use $m = 8$ projections for CS-MoG for the rest of this performance evaluation.

Performance of CS-MoG, MoG and other MoG based algorithms. In this part, we compare the performance of eight background subtraction methods. They include pixel based MoG [31], CS-MoG-Balance and CS-MoG-Unbalance, Multivariate-CS-MoG (each vector of projections is modelled as a mixture of multivariate Gaussian distributions, see on-line supplementary materials for detailed description), PCA-MoG, DCT-MoG as well as RS-MoG and Mean-MoG introduced in Section "Goals, Metrics and Methodology".

We apply these eight methods to 400 consecutive video frames from each dataset. The value of m is chosen to be 8 for CS-MoG-Balance, CS-MoG-Unbalance, PCA-MoG, DCT-MoG, RS-MoG and Mean-MoG. We first evaluate the accuracy of the foreground block detections. The RoC curves of the block wise results for these methods are plotted in Fig. 5b for the dataset 1.

We can see from the figure that out of all the methods that use dimensionality reduction, CS-MoG-Balance gives the best performance. It may not be surprising that the simple methods such as RS-MoG and Mean-MoG do not perform that well. (Again, the results of datasets 2 and 3 also indicate the same conclusion as in [27].) The observation that CS-MoG-Balance performs better than CS-MoG-Unbalance deserves further investigation. This is the topic of Section "Balanced versus Unbalanced Projection Matrices". Another observation is that the accuracy of Multivariate-CS-MoG is slightly worse than CS-MoG-Balance although it consumes significantly more computation resources because of the need to matrix inverse. The reason may be due to the fact Multivariate CS-MoG needs to learn a larger number of paramerters for the covariance matrices.

We see from the figure that the performance of MoG and CS-MoG-Balance are comparable. It is probably not surprising that MoG has a better performance most of the time because it maintains complete information on each pixel.

However, the better performance of MoG comes at the expense of a high computation cost. We will show in Section 5.1, by implementing both MoG and CS-MoG on an embedded platform, the computation time for MoG is 5-6 times slower than that of CS-MoG and real-time background subtraction with MoG is not feasible. Therefore, when we take into account both performance and resource constraints on embedded platforms, CS-MoG-Balance is a better choice compared with MoG. You can find more evaluation results by referring [27].

We then evaluate the pixel-wise accuracy of these methods. All the MoG-based methods, except the original MoG, use the same foreground refinement method introduced in Section "Foreground Refinement". The results are shown in Fig. 5c. The CS-MoG-Balance again achieves the second best place among the methods and is close to original MoG. Comparing the block-wise and pixel-wise results shown in Figs. 5b and 5c, the performance gap of pixel-wise accuracy between CS-MoG-balance and original MoG is almost the same to that of block-wise accuracy. Therefore, although foreground refinement model used is simple, it is sufficient for achieving high overall accuracy.

We have also compared CS-MoG against a recent proposed compressive sensing based background subtraction method [9]. The results show that CS-MoG has a much better accuracy, see on-line supplementary materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2015.2418775 available online, for details.

Balanced versus unbalanced projection matrices. The evaluation in Section "Performance of CS-MoG, MoG and other MoG based algorithms" show that CS-MoG-Balance has a better performance compared with CS-MoG-Unbalanced. Closer investigation shows that the performance difference between these two methods is due to the way they handle illumination change. We claim that balance projection matrices have a better capability to address illumination change effects than unbalanced projection matrices. The reason simply comes from its "balance" feature. In order to explain this, we investigate the effect of projection matrices using two different illumination change models.

The first and simpler illumination change model is taken from [23]. The ith pixel value measured by the camera depends on the illumination reaching the surface (l_i) and its albedo feature a_i. Under the Lambertian assumption, the pixel value is: $x_i = l_i a_i$. The albedo feature a_i is determined by the feature of the surface. Thus, a change in the source of illumination will affect the pixel value x_i through l_i. If the source of illumination is large and far away, the illumination reaching a small surface area can be assumed to be constant over the limited size of the surface. Given that our proposed CS-MoG considers a small block of pixels at a time, we can therefore assume that l_i is constant over a block. Consequently, the illumination change measured by the camera within a block is only determined by the albedo feature a_i.

Let us consider the case that the block experiences a constant shift in illumination level Δ, i.e., $l_i = \Delta$ for all pixels in a block. Similar to Section 4.1.2, we use β_i (which equals to ± 1) to denote the elements of the projection vector. The change in projection value δv is given by: $\delta v = (\sum_i \beta_i a_i)\Delta$.

Consider the case that the surface feature is the same within the block, then a_i takes the same value for all pixels in a block. If the matrix is balanced, the change in projection value δv is zero because $\sum_i \beta_i$ is always zero for a balanced projection matrix. Therefore a constant change in illumination will not change the projection value. However, if the surface feature is not even, which happens when the block is at the edge of the foreground, the illumination change will not be cancelled out. Although the illumination change is not always exactly constant on the whole block, especially near the edge of the foreground, a balanced projection matrix still has a better ability to deal with the illumination change compared with a unbalanced one.

More precise results can be obtained by assuming a statistical model for illumination change. By assuming that the illumination change is Gaussian distributed, we show analytically in the supplementary materials, available online, that a balanced projection matrix maximises the probability of correct detection of background.

4.2 CS-MoG in Colour Space

Although CS-MoG has achieved comparably good accuracy against the original MoG, there is still a margin between the ROC curves of CS-MoG and original MoG shown in Fig. 5c. To further improve the performance of compressive sensing based background subtraction method, we propose another novel background subtraction method to make use of colour information. We call this new approach as Colour Space Compressed Samples and will refer to it as CoSCS-MoG. Then, we compare the performance of CoSCS-MoG, against the original MoG with illuminance information and colour information respectively and CS-MoG. Our evaluation shows that the CoSCS-MoG performs significantly better than CS-MoG, and equally well or better than the original MoG.

4.2.1 CoSCS-MoG

The CS-MoG algorithm in Section 4.1 uses only the luminance channel for background subtraction. A straightforward generalisation is to treat each colour channel independently and then fuse the results. Unfortunately, while requiring three times more computation cost, this method does not improve the accuracy at all. Another choice is to model the multiple colour channels with three-dimensional Gaussian distribution. However, the high dimensional Gaussian model is computationally more prohibitive and will incur the phenomena of *curse of dimensionality* [4]. Therefore, we develop CoSCS-MoG to treat the three colour channels in an integrated manner and do not introduce high dimensional Gaussian models.

Choice of colour space. CoSCS-MoG works on YC_bC_r which is widely used in digital image processing. Y is the luminance component which is the same as grayscale intensity. C_b and C_r are, respectively, the blue-difference and red-difference chrominance components. Note that the trivial choice of the RGB colour space results in a worse accuracy.

CoSCS on blocks of pixels. After acquiring a new colour video frame in YC_bC_r space, the first step of CoSCS is to divide the frame into blocks of 8×8 pixels. For each of the colour channel of each block, we form a 64×1 vector of the pixel values of the channel. We then stack the three vectors
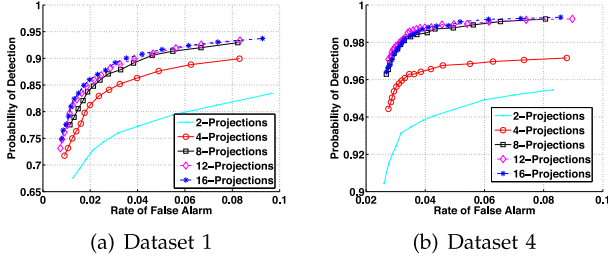
Fig. 6. ROC curves of different number of projections.

Fig. 7. ROC curves of different methods.

from the three colour channels to form a 192×1 vector x. Therefore, each block is now associated with a 192×1 vector. (In order to simplify the description of CoSCS later on, we will assume from now on that the luminance values are placed in the first 64 elements of x.) We then follow CS-MoG and compute the projection of x for each block. The same projection matrix is used for all the blocks for the whole video, so that it only has to be generated once.

Mathematically, the computation of the projection values at each block can be represented as follows. Let $n = 192$. We denote the projection matrix by Φ which is an $m \times n$ matrix. The projection values are in the vector $y = \Phi x$. Our evaluation in Section 4.2.2 shows that there is negligible improvement in accuracy in using $m \geq 8$, so we will assume $m = 8$ from now on. We would like to remark on the fact that only $m = 8$ projections are required for CoSCS-MoG is significant. Since CS-MoG uses only eight projections, we will see in Section 5 that the computational requirements for CoSCS-MoG and CS-MoG are similar. Taking into account that CoSCS-MoG has better background subtraction performance compared to CS-MoG (Section 4.2.2), this means the use of colours can significantly improve the performance but with little overhead in computation time.

We show in Section 4.1.4 that balanced projection matrix performs better in CS-MoG, which is based on luminance channel alone. For CoSCS-MoG, we impose that, for each row of the projection matrix Φ, the vector formed by the first 64 elements—which projects the luminance channel—is balanced. The other elements in the matrix Φ, in columns 65 to 128, are drawn from symmetric Bernoulli distribution.

MoG in random projections subspace. After computing the projections for each block we need a method to determine whether the block contains some part of the foreground according to the projection values in y. The main idea of this part is the same as CS-MoG. We model each projection value independently as a MoG with three Gaussians and do not consider possible correlation between the projection values. Again, if a projection value lies within 2.5 standard deviation of one of the background distributions, then it is considered to be a background and otherwise it is a foreground. Because each vector y contains eight different projections values, it is likely that we get a mixture of background and foreground decisions from a vector y. We therefore need a method to fuse the results and we use the same voting method as CS-MoG. At this point, we have the background subtraction results at the resolution of a block. We then follow CS-MoG to obtain the pixel level background subtraction results. The pixel-level method is identical to CS-MoG and uses only the luminance component for pixel level foreground detection. Note that our experience
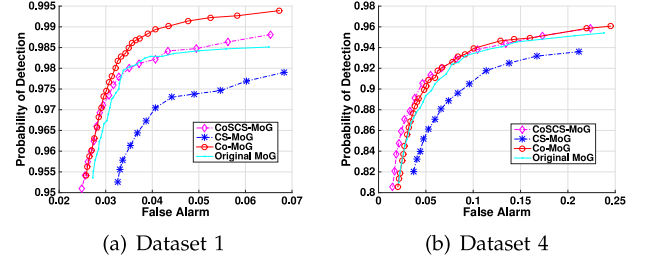
shows that the resulting background subtraction method has good accuracy.

4.2.2 Performance Evaluation

In this section, we again regard a pixel in background (resp. foreground) as a negative (positive) event. The use of ROC to measure the performance of background subtraction is the same as before.

Impact of number of projections on the ROC curves. To determine the number of projections required for a good performance, we also evaluate CoSCS-MoG on two different datasets whose exact ground truth is available: one is our private datasets (dataset 1) and one from Perception Sequence (dataset 2). The results shown in Fig. 6 are similar to that of CS-MoG: there is no substantial improvement on the accuracy when the number of projections is above 8. Therefore, we also choose 8 as the default number of projections for CoSCS-MoG.

Performance comparison. We compare four methods: the conventional MoG with luminance information only (original MoG), the conventional MoG with colour information (Co-MoG), CS-MoG, and CoSCS-MoG. In particular, we want to see: 1) whether CoSCS-MoG performs better than CS-MoG; 2) whether the CoSCS-MoG can achieve almost the same accuracy as MoG. We use datasets 1 and 4 for evaluation. All of the datasets contain multi-modal background, lighting changes and are complex enough for the evaluation of the background subtraction methods.

The ROC curves of different background subtraction methods are shown in Fig. 7. We apply the four methods to 400 consecutive video frames from the two datasets. We see from Fig. 7 that CoSCS-MoG outperforms CS-MoG significantly and it achieves almost the same background subtraction accuracy as the original MoG. Another observation is that Co-MoG cannot guarantee significantly better accuracy than other methods although it consumes almost three times computation as the original MoG. From Fig. 7 we can see that Co-MoG only provides sightly better subtraction accuracy (about 0.5 percent in dataset 1) or almost the same accuracy as CoSCS-MoG (in dataset 4).

5 EXPERIMENTS ON PLATFORMS

Two sets of experiments were conducted to evaluate the performance of the proposed approaches. The first set aimed at benchmarking the performance of the proposed algorithm against the original MoG in terms of computation time on various platforms (Blackfin DSP-camera node and PandaBoard). The second set of experiments demonstrate the feasibility of new algorithms on embedded platforms with an end-to-end distributed multi-camera object tracking application.

TABLE 3
Computation Time of Different Methods

	MoG	CS-MoG	CoSCS-MoG
Initialisation (ms)	1.80	1.80	1.80
CS (ms)	-	2.8	8.9
BS (ms)	337.2	52.2	54.6
Total (ms)	339	56.8	65.3
Energy Consumption (mJ)	649.95	116.44	125.96

5.1 Computation Evaluation on PandaBoard

We have demonstrated in our earlier work [27] that, on the Blackfin DSP camera nodes, MoG could only process less than three video frames a second and CS-MoG could process 15 frames a second assuming a frame size of 320×240. The long computation time of MoG is due to the need to update the MoG probability distribution parameters. Since CS-MoG uses eight times fewer MoG probability distributions compared to MoG, the computation time for CS-MoG is much faster. To demonstrate our algorithm is not platform specific, the other implementation was based on one of the most widely used embedded platform: PandaBoard (PandaBoard ES Rev B1) connected with a USB video camera (Logitech HD Pro Webcam C920). PandaBoard can accommodate various embedded operating systems such as Linux Minimal, Android and Ubuntu. In this implementation, the operating system is Ubuntu 12.04. The programming language can be *C* or *C++*. PandaBoard has its built-in wifi component. Therefore, it is convenient to exchange information between PandaBoards and base station. The computation time of each method is measured by the CPU time of the platform and the energy consumption is obtained by measuring the current, voltage and the processing time. Table 3 shows the computation and energy consumptions of original MoG and CS-MoG. We do not include the memory usage because PandaBoard has 2 Gigabyte RAM which is not a problem in our application. These run-time and energy consumption results were computed as the mean of 100 consecutive images with the image resolution at 320×240. The results show that, MoG can only process about three frames per second which is far from of real-time. However, CS-MoG accelerates the MoG based background subtraction up to six times. Therefore, our proposed approach demonstrates consistently good performance for accelerating background subtraction.

Note that in Table 3, CS refers to the time for computing projections, which is negligible; BS refers to the time to perform background subtraction and is dominated by the time required to update the MoG parameters. Since CS-MoG has eight times fewer parameters compared with MoG, the computation time for BS is significantly reduced.

5.2 Computation Evaluation of CoSCS-MoG

According to the computation evaluation on Pandaboard (Blackfin), CS-MoG require 57 ms (56.8 ms) to process a frame. Out of the 57 ms (56.8 ms), CS-MoG spends 52.75 ms (52.2 ms) and 3 ms (2.8 ms), respectively, on background subtraction processing and computing the projections (i.e., the compressed sampling component). Since CS-MoG and CoSCS-MoG use exactly the same number of projections per block and the same number of Gaussians per projection value, CoSCS-MoG is expected to use the same amount of time for MoG processing as CS-MoG. Since the projection computation in CoSCS-MoG is based on three colour channels, compared to one channel for CS-MoG, we expect CoSCS-MoG may take up to 9 ms (8.4 ms) to compute the projections. Therefore, CoSCS-MoG is expected to need 63 ms to process a frame.

To validate our conjecture, we also implemented CoSCS-MoG on PandaBoard. The results in the third column of Table 3 show that the total run time of CoSCS-MoG is 65.3 ms (it is 5.2 times faster than original MoG) to process a frame. Computing the projections consumes about 8.9 ms which is about 3.17 times compared with that in CS-MoG.

5.3 Compressive Background Subtraction for Real-Time Distributed Object Tracking

To demonstrate the feasibility of our proposed approach for embedded computer vision applications, we further implemented an end-to-end distributed multi-camera tracking application.

In the experiments, three wireless cameras were set-up in an approximately $4\,\text{m} \times 4\,\text{m}$ area with overlapping coverage of the ground. The cameras communicated with a server using the BLIP and IPv6 network. We used a toy train as the target in the experiments in order to collect high-precision (in cm) ground truth information.

We further deployed a number of tags on the ground which were needed to compute the ground plane homographies [33] of each camera. A homography is a projective transformation that maps the coordinates from one plane to another, which in this case is the camera's image plane and the ground. With the computed homographies, we were able to obtain the calculated locations of the target and the ground truth in the ground coordinates with high-precision.

The target (train) moved along a track within the area of interest. All cameras continuously processed incoming frames to firstly segmented out the moving foreground, which was then passed into a connected component analysis that outputed the centroid of the moving object. The centroids are taken as the objects' locations in the image coordinates. To conserve resources (in terms of radio bandwidth and energy consumption), packets that contained these

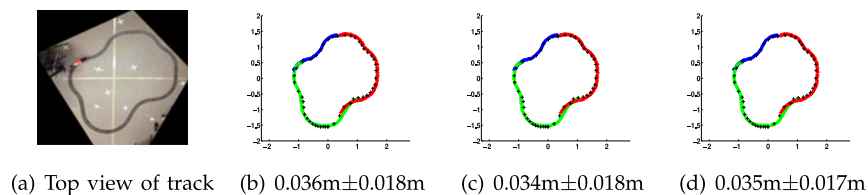(a) Top view of track   (b) 0.036m±0.018m   (c) 0.034m±0.018m   (d) 0.035m±0.017m

Fig. 8. Target tracking experiment set-up and results. The black cross are the ground truth and the other different colour cross represent the results from different camera nodes.

object's locations (in the image coordinates) were only transmitted to the server from a camera when the object was in the camera's field of view. When the server obtained a location message along with a camera ID (e.g., IPv6 address), it would calculate the locations in the ground coordinates using the corresponding homography of the camera.

We ran the tracking experiment on the track of the train and the target (train) made three laps on the track (see Fig. 8a). The right columns of Fig. 8 shows the tracking results of each lap. The black crosses in Fig. 8 are the ground truth in the ground coordinates and the results from three camera nodes are shown as crosses of different colours (red, green, blue). Furthermore, we calculated the mean and standard deviation of the distances from the estimated target locations to the ground truth and the results are shown in the caption of the figures. Overall, we achieved high precision with less than 1 percent (less than 4 cm) target localisation errors relative to the size of the area of interest ($4\,\mathrm{m} \times 4\,\mathrm{m}$).

5.4 Tracking Performance with Multiple Objects Moving at Different Speeds

The experiment above considered only one single object moving at a slow speed of 0.1 m/s. In order to investigate the tracking performance of our background subtraction algorithm when there are multiple objects moving at different speeds, we have conducted another experiment with two people walking at different speeds through an area of surveillance. Our tracking algorithm can track the two people within 27 cm accuracy. Due to space limitation, the results are described in Section 3 of the on-line supplemental materials, available online.

6 CONCLUSION

In this paper, we address the challenge of performing background subtraction, both accurately and efficiently, on embedded camera networks. Traditional background subtraction algorithms, though accurate, are not computationally efficient because complex statistical models are needed to capture subtle illumination changes. To address this computation bottleneck, we use random projections to reduce the dimensionality of the data while retaining the information content. This results in a computationally efficient and yet accurate background subtraction algorithm. Our experiments show that the accuracy of this algorithms is comparable to that of traditional algorithms. Moreover they are up to six times more efficient on embedded platforms. Furthermore, we show that our proposed approach can accurately track a moving object in real-time on an embedded camera network. Y. Shen is the corresponding author.

REFERENCES

[1] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, Apr. 2010.
[2] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.
[3] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, Distributed compressed sensing, *arXiv preprint arXiv:0901.3403*, 2009.
[4] R. Bellman. *Dynamic Programming*, Princeton University Press, 1957.
[5] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. "Review and evaluation of Commonly-implemented background subtraction algorithms," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[6] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *Ann. Statist.*, vol. 38, pp. 2916–2957, 2010.
[7] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, 52, pp. 489–509, Feb. 2006.
[8] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 11, 2011.
[9] V. Cevher, A. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *Proc. 10th Eur. Conf. Comput. Vis*, Berlin, Heidelberg, 2008, pp. 155–168.
[10] M. Cossalter, M. Tagliasacchi, and G. Valenzise, "Privacy-enabled object tracking in video sequences using compressive sensing," in *Proc. 6th IEEE Conf. Adv. Video Signal Based Surveillance*, Sep. 2009, pp. 436–441.
[11] M. Dikmen and T. S. Huang, "Robust estimation of foreground in surveillance videos by sparse error estimation," in *Proc. 19th Int. Conf. Pattern Recognit.*, 2008, pp. 1–4.
[12] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[13] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 83–91, Mar. 2008.
[14] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proc. 6th Eur. Conf. Comput. Vis.*, London, UK, 2000, pp. 751–767.
[15] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proc. 13th Conf. Uncertain. Artif. Intell.*, San Francisco, CA, USA, 1997, pp. 175–181.
[16] E. Hayman and J.-O. Eklundh, "Statistical background subtraction for a mobile observer," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Oct. 2003, pp. 67–74.
[17] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici, "CMOS compressed imaging by random convolution," in *Proc. 34th IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 1113–1116.
[18] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Proc. Int. Conf. Image Process.*, 2004, vol. 5, pp. 3061–3064.
[19] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground–background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.
[20] E. G. Learned-Miller, M. Narayana, and A. Hanson, "Background modeling using adaptive pixelwise kernel variances in a hybrid feature space," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2012, pp. 2104–2111.
[21] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.
[22] P. Misra, W. Hu, M. Yang, and S. Jha, "Efficient cross-correlation via sparse representation in sensor networks," in *Proc. 11th Int. Conf. Inf. Proc. Sens. Netw.*, 2012, pp. 13–24.
[23] J. Pilet, C. Strecha, and P. Fua, "Making background subtraction robust to sudden illumination changes," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 567–580.
[24] D. Reddy, A. Sankaranarayanan, V. Cevher, and R. Chellappa, "Compressed sensing for Multi-view tracking and 3-d voxel reconstruction," in *Proc. 15th IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 221–224.
[25] J. Romberg, "Compressive sensing by random convolution," *SIAM J. Imaging Sci.*, vol. 2, pp. 1098–1128, 2009.
[26] D. Russell and S. Gong, "A highly efficient block-based dynamic background model," in *Proc. 3th IEEE Conf. Adv. Video Signal Based Surveillance*, Sep. 2005, pp. 417–422.
[27] Y. Shen, W. Hu, J. Liu, M. Yang, B. Wei, and C. T. Chou, "Efficient background subtraction for real-time tracking in embedded camera networks," in *Proc. 10th ACM Conf. Embedded Netw. Sens. Syst.*, 2012, pp. 295–308.
[28] Y. Shen, W. Hu, R. Rana, and C. T. Chou, "Non-uniform compressive sensing in wireless sensor networks: Feasibility and application," in *Proc. 7th Int. Conf. Intell. Sens., Sens. Netw. Inf. Proc.*, 2011, pp. 271–276.

[29] Y. Shen, W. Hu, R. Rana, and C. T. Chou, "Nonuniform compressive sensing for heterogeneous wireless sensor networks," *IEEE Sens. J.*, vol. 13, no. 6, pp. 2120–2128, Jun. 2013.

[30] Y. Shen, W. Hu, M. Yang, B. Wei, S. Lucey, and C. T. Chou, "Face recognition on smartphones via optimised sparse representation classification," in *Proc. 13th Int. Symp. Inf. Proc. Sens. Netw.*, 2014, pp. 237–248.

[31] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. 12th IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 1999, pp. 246–252.

[32] Z. Tang and Z. Miao, "Fast background subtraction and shadow elimination using improved gaussian mixture model," in *Proc. IEEE Int. Workshop Haptic, Audio Visual Environ. Games*, Oct. 2007, pp. 38–41.

[33] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Proc. Int. Workshop Vis. Algorithms: Theory Practice*, London, UK, 2000, pp. 278–294.

[34] B. Wei, W. Hu, M. Yang, B. Wei, and C. T. Chou, "Radio-based device-free activity recognition with radio frequency interference," in *Proc. ACM/IEEE Conf. Inf. Proc. Sens. Netw.*, 2015.

[35] B. Wei, M. Yang, Y. Shen, R. Rana, C. T. Chou, and W. Hu, "Real-time classification via sparse representation in acoustic sensor networks," in *Proc. 11th ACM Conf. Embedded Netw. Sens. Syst.*, 2013, pp. 21.

[36] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

[37] X. Wu and M. Liu, "In-situ soil moisture sensing: measurement scheduling and estimation using compressive sensing," in *Proc. 11th Int. Conf. Inf. Proc. Sens. Netw.*, New York, NY, USA, 2012, pp. 1–12.

[38] C. Zhang, F. Li, J. Luo, and Y. He, "Ilocscan: Harnessing multipath for simultaneous indoor source localization and space scanning," in *Proc. 12th ACM Conf. Embedded Netw. Sens. Syst.*, 2014, pp. 91–104.

[39] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 864–877.

[40] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit.*, Aug. 2004, vol. 2, pp. 28–31.

Yiran Shen received the PhD degree in computer science and engineering from the University of New South Wales. He is an associate professor in the College of Computer Science and Technology, Harbin Engineering University (HEU). He was SMART Scholar at Singapore-MIT Alliance for Research and Technology before he joined HEU. He publishes regularly at top-tier conferences and journals. His current research interests are wearable/ mobile computing, wireless sensor networks and applications of compressive sensing. He is a member of the IEEE.

Wen Hu is a senior lecturer at the School of Computer Science and Engineering, the University of New South Wales (UNSW). Much of his research career has focused on the novel applications, low-power communications, security and compressive sensing in sensor network systems, and Internet of Things (IoT). He published regularly in the top rated sensor network and mobile computing venues such as ACM/IEEE IPSN, ACM SenSys, ACM and IEEE Transactions, Proceedings of the IEEE, and *Ad-hoc Networks*. He serves on the organising and program committees of networking conferences including ACM/IEEE IPSN, ACM SenSys, ACM MobiSys, and IEEE ICDCS. He is a senior member of the IEEE.
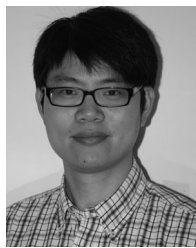
Mingrui Yang (M'11) received the PhD degree in mathematics from the University of South Carolina in 2011. He is a postdoctoral research fellow in the Autonomous System Lab of the Digital Productivity Flagship in CSIRO, Australia. He is interested in interdisciplinary researches in compressive sensing, sparse approximation, signal/image processing, greedy algorithms and nonlinear approximation, and their applications. He has published in high-quality journals and conferences in mathematics and computer science, such as *Advances in Computational Mathematics*, ACM/IEEE IPSN, and ACM SenSys. He is a member of the IEEE.
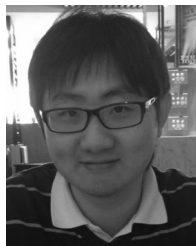
Junbin Liu received the BEng degree in telecommunications with first class honors and the PhD degree in the School of Electrical Engineering & Computer Science of QUT in 2008 and 2013, receptively. He is a researcher from the Image and Video Technology Lab at Queensland University of Technology (QUT), Australia. His research interests include distributed vision systems, camera placement, and face alignment. He is a member of the IEEE.

Bo Wei is received the bachelor of engineering and the master of engineering degrees in 2009 and 2011, repectively, from the College of Information Science and Engineering, Northeastern University, Shenyang, China. He is working towards the PhD degree at the University of New South Wales, Australia. He was also a visiting student in Commonwealth Scientific and Industrial Research Organisation (CSIRO) Australia and the Swedish Institute of Computer Science (SICS). He is a member of the IEEE.

Simon Lucey received the PhD degree from the Queensland University of Technology, Brisbane, Australia, in 2003. He is an associate research professor in the Robotics Institute at Carnegie Mellon University. He also holds an adjunct professorial position at the Queensland University of Technology. His research interests include computer vision and machine learning, and their application to human behavior. He is a senior member of the IEEE.

Chun Tung Chou received the BA degree in engineering science from the University of Oxford, United Kingdom, and the PhD degree in control engineering from the University of Cambridge, United Kingom. He is an associate professor in the School of Computer Science and Engineering, The University of New South Wales, Australia. His current research interests are wireless sensor networks, compressive sensing, and nano-communication. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.