Background Subtraction using Spatio-Temporal Group Sparsity Recovery

Xin Liu, Jiawen Yao, Xiaopeng Hong, Xiaohua Huang, Ziheng Zhou, Chun Qi, and Guoying Zhao*

Abstract—Background subtraction is the key step for a wide spectrum of video applications such as object tracking and human behavior analysis. Compressive sensing based methods, which make little specific assumptions about the background, have recently attracted wide attention in background subtraction. Within the framework of compressive sensing, background subtraction is solved as a decomposition and optimization problem, where the foreground is typically modeled as pixel-wise sparse outliers. However, in real videos, foreground pixels are often not randomly distributed, but instead, group clustered. Moreover, due to costly computational expenses, most compressive sensing based methods are unable to process frames online. In this paper, we take into account the group properties of foreground signals in both spatial and temporal domains, and propose a greedy pursuit based method called Spatio-Temporal Group Sparsity recovery, which prunes data residues in an iterative process, according to both sparsity and group clustering priors, rather than merely sparsity. Furthermore, a random strategy for background dictionary learning is used to handle complex background variations, while foreground-free training is not required. Finally, we propose a two-pass framework to achieve the online processing. The proposed method is validated on multiple challenging video sequences. Experiments demonstrate that our approach effectively works on a wide range of complex scenarios, and achieves a state-of-the-art performance with far fewer computations.

Index Terms—Background subtraction, foreground detection, group sparsity, sparse signal recovery, orthogonal matching pursuit, spatio-temporal

I. INTRODUCTION

TASk of foreground object detection originates from numerous applications in computer vision, such as human machine interactions, intelligent video surveillance [1], and human behavior analysis. Background subtraction is a commonly used approach for extracting foreground in videos from static cameras. The performance of background subtraction mainly depends on the algorithm used for modeling background.

In the past few decades, a large number of models have been proposed. One of the most famous pixel-based methods is the Gaussian Mixture Models [2] [3], which use a mixture of Gaussian probability density functions to model color intensity variations of individual pixels. In [4], Li et al. utilized spatial and temporal features to model dynamic backgrounds, and used a Bayesian rule to estimate the probability distributions. The code-book methods of [5] [6] record the background states of each pixel by a certain number of codewords, and the foreground is detected by a distance measurement. The MAP-MRF algorithm in [7] used a maximum a posteriori Markov random fields (MRF) decision framework to determine the pixels as either background or foreground. The method of self-organizing artificial neural network in [8] [9] presented another alternative for solving the background subtraction problem. In ViBe [10] and PBAS [11], background modeling is based on the selection and updating of pixel samples. A more detailed discussion of these conventional techniques can be found in recent surveys [12] [13] [14] [15].

Despite having made significant progress in foreground detection, it is still challenging. Most of these methods fail to work well in dynamic complex situations. The main reasons are that these methods often make overly restrictive assumptions about the background [16] [17], and that they are rarely aware of the correlation of background across different frames. The background is a complex environment that usually includes distracting motion. An adaptive background model should also detect shadows cast by moving objects, and handle various changes such as where new objects are introduced into the background or old ones are removed from it. Furthermore, an ideal background modeling should be able to tolerate sudden background variations like turning on/off of lights, without losing the sensitivity to detect real foreground objects. In light of these complex factors, it is very difficult to achieve good background modeling by making specific assumptions.

Recently, compressive sensing [18] [19] based methods have shown promising performances in many vision tasks. Foreground detection is formulated as a classic problem of learning a low dimensional linear model from high dimensional data. According to different assumptions on background, compressive sensing based methods can be grouped into two major categories, namely the Robust Principle Component Analysis

Copyright (c) 2017 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.
(RPCA) and sparse signal recovery. The only assumption made about the background by RPCA [20] is that any variation in background appearance is highly constrained and can be captured by the low-rank condition of a suitably formulated matrix [16]. Sparse signal recovery is another popular model, which only assumes that a new coming frame can be modeled as a sparse linear combination of a few preceding frames plus a sparse outlier term [21]. Compressive sensing based methods can handle most challenges mentioned above by making very little specific assumptions about the background [16] [17]. However, there are still three remaining issues:

- The constraint on the foreground tends to treat each entry independently, without considering the group properties of the foreground pixels. Few works have explored the group properties in both spatial and temporal domains.
- Methods based on the sparse signal recovery assume that background should be modeled as a sparse linear combination of the preceding frames, but the observations such as in video surveillance are often corrupted by the foreground and noise.
- The computation of existing methods is very expensive. Although some on-line methods have been proposed, only a few of them can run as a real-time processing system.

To solve the first problem, previous methods [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] consider possible relationships among subsets of the non-zero entries (foreground pixels) on the spatial domain. In fact, the group information does exist in real video scenarios, Fig. 1 shows an example, and we can see that non-zero coefficients are not randomly distributed but clustered spatially in the foreground image. However, few works have considered the group clustering priors in the temporal domain. Moreover, it is not difficult to find the temporal continuity of non-zero coefficients from the two consecutive frames in Fig. 1.

With the second issue, conventional sparse signal recovery methods extract frames directly from video sequence [21] [35], as the atoms of background dictionary, which often contain both background and unwanted foreground. Thus, it leads to poor background recovery and inaccurate foreground segmentation. To deal with this issue, a robust background dictionary learning mechanism is needed, which can automatically prune unwanted foreground pixels out, and collect a few clean background frames as the atoms. Moreover, an ideal dictionary should represent events in the background as much as possible.

In the third issue, RPCA based algorithms are batch models [20] [22] [23] [24] [25] [27] [30] [32] [33], emphasize the low-rank property of the input matrix stacked by a large number of frames [20], in that the matrix decomposition can be started only if all of the predefined number of successive frames is observed. Since the computational cost of most RPCA algorithms is dominated by the evaluation of the low-rank of observation matrix, typically, operated by singular value decomposition (SVD) with high time complexity $\mathcal{O}(mnr)$ [36], where $m$ is the size of a frame, $n$ is the number of frames, and $r$ is the rank of the matrix. Also, it should be noted that the tremendous memory requirements due to the typical size of input matrix, such as the sequence “Fall” in dataset CDnet [37], its resolution is $720 \times 480$, the size of a input matrix with $400$ frames is $1036800\times 400$, equivalent to $2.17$ Gb in single precision floating-point representation. Although some online and incremental RPCA methods have been proposed, they still need a batch initialization [28] [38] [39] [40] [41] [42] [43] to guarantee the accuracy of the background estimation, otherwise they need other procedure to refine the foreground, such as MRF [44] [45] and median filter post-processing [46] [42] [43], some incremental methods also utilized pre-processing step to improve the speed and the quality of results, for instance, using superpixel to obtain homogeneous regions (structural and group information) of each frame [28] [47], and saliency map to detect salient foreground object rather than background [34]. On the other hand, the sparse signal recovery based methods process the frames sequentially, but the computations are still expensive, since most of the existing methods assume that the sparsity $K$ was known before recovery, while $K$ may not be available in many practical applications, such as foreground detection. Therefore, a slow step-wise approximation scheme is always needed to insure the recovery performance.

In this paper, we propose a novel method for background subtraction, which falls into the category of sparse signal recovery. We formulate the problem in a unified framework named Spatio-Temporal Group Sparsity recovery. The main contributions are summarized as follows:

1) We propose a new formulation of foreground detection via a fast greedy pursuit algorithm. It explicitly considers group properties of sparse outliers (foreground) on both spatial and temporal domains for better sparse recovery, instead of merely considering the spatial, as conventional methods do.
2) We formulate the background modeling as a dictionary learning problem, so that a training sequence without any foreground is not required. Furthermore, a random update policy is employed to extend the time windows covered by the background dictionary, therefore it can deal with a wide range of events in the background scene, such as dynamic background motions and illumination variations. This background model can rapidly respond to sudden background changes.
3) We propose a two-pass framework to achieve the online processing. The experimental results show that the proposed framework can significantly improve running speed.

The remainder of this paper is organized as follows. Section II reviews prior work and related methods. In Section III, the spatio-temporal group sparsity recovery for background
subtraction is presented. In Section IV, the two-pass framework for online processing is described. While experiments and discussions are presented in Section V and conclusions are drawn in Section VI.

II. RELATED METHODS

A. Robust Principle Component Analysis

In RPCA [20], Wright et al. considered foreground detection from a viewpoint of matrix decomposition and optimization problems, which can be expressed as follows:

\[
\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad D = L + S
\]

where \( D \in \mathbb{R}^{m \times n} \) is the observed videos (\( n \) frames), and \( m \) is the size of a frame. \( L \) and \( S \) denote the background and foreground signals respectively. It is assumed that the background images are linearly correlated with each other, forming a low-rank matrix \( L \) (\( \|L\|_* \) means the nuclear norm of matrix \( L \)). And the \( \ell_1 \)-norm is employed to constrain the foreground, since these regions should be a sparse matrix with a small fraction of non-zero entries.

However, as the geometry of \( \ell_1 \)-norm is diamond shaped and this regularization treats each entry (pixel) independently, it does not take into account any possible relations among subsets of entries [25] [26], while in real videos, foreground pixels usually have the group properties of spatial connection and temporal contiguity (see Fig. 1). To solve this issue, in [23] [24], the low-rank and block-sparse matrix decomposition is proposed, which adopts the \( \ell_{2,1} \)-norm to detect outliers (foreground) with column-wise sparsity. However, the block-sparsity property still has no structured information to model sparse outliers [22]. In [48], the local sparseness constraint is exploited and total variation (TV) penalty was employed to better deal with corrupted data. In DECOLOR [25], a spatial constraint that favors contiguous regions was incorporated using the Markov Random Field (MRF), but foreground regions tend to be over-smoothed due to the smoothness constraint. In addition, the structural and group information about non-zero patterns of variables has been explored in the Bayesian robust matrix factorization (BRMF) [27], block sparse [16], TV regularizer with RPCA (TV-RPCA) [29], low-rank and structured sparse decomposition (LSD) [22], and generalized fused Lasso (GFL) [30]. However, those approaches like RPCA (low-rank based) is a batch method stacking a number of training frames in the input observation matrix [49]. This limitation restricts them to be used for time-critical purposes such as unusual event detection. In [50], Oh et al. used partial sum minimization of singular values rather than nuclear norm for the rank minimization method in RPCA, which achieved a better control of the target rank of the low-rank solution, even when the number of observations is limited. However, it is still a batch model method.

To solve this issue, many efforts have been devoted to developing online or incremental model of RPCA. Javed et al. [44] [45] [46] proposed an online RPCA (OR-PCA) [51] using multiple features together with MRF constraints. OR-PCA processes frames in an online manner. Moreover, it can handle the challenging global illumination changes in background. However, the method using the MRF may tend to be over-smoothed, share the same limitation as DECOLOR. In [47], Javed et al. proposed an online method by using the superpixel as inputs, named superpixel-based online matrix decomposition with structured foreground constraints (SOD-SC), which obtained better F-measure performance than RPCA with a high frame rate, but may difficult to deal with the heavy camera jitter case due to the smoothness property of the superpixel.

In [38], Qiu and Vaswani proposed singular value decomposition (SVD) based method named recursive projected compressive sensing (ReProCS), which can process observations (frames) one by one, but the precondition is that the background matrix has been accurately estimated [38]. Thus, ReProCS employed a batch method in its SVD-based initialization step [38] [36]. Guo et al. [39] (Prac-ReProCS) extended ReProCS by using the incremental SVD procedures to adapt to slow variations of background, but Prac-ReProCS also required a batch initialization [39] [36]. In [40], He et al. proposed an online method for low-rank subspace tracking, named Grassmannian robust adaptive subspace tracking algorithm (GRASTA). Nevertheless, GRASTA also used the \( \ell_1 \)-norm as a convex relaxation of the ideal sparsifying function [49]. Another limitation is that GRASTA requires a batch initialization stage to guarantee the quality of background estimation. In [28], Xu et al. proposed a similar two-stage solution to online processing, called Grassmannian online subspace updates with structured sparsity (GOSUS), it imposes a connectivity constraint by grouping pixels with a superpixel method and encouraging sparsity of the groups. However, due to the high computational cost of the ADMM, the algorithm could only achieve 1 ∼ 2 frames per second (FPS) given an image size of \( 160 \times 120 \) [28], which certainly cannot satisfy the demand of real-time processing. In fact, both GRASTA and GOSUS have another implementation by initializing the background in a non-batch model, but the detection results are not as good as when the default batch procedure is used [36]. In [41], Hu et al. proposed an online method including a regression based low rank background model and a foreground model promoting the foreground contiguity, but it also had a batch initialization step. In pROST [42] [43], an \( \ell_p \)-norm (\( p < 1 \)) of the singular values is used to estimate the low rank sub-space representation of the background [36], rather than \( \ell_1 \) in GRASTA. The method has been implemented on GPU, which can achieve real-time performance at a resolution of
160 × 120. However, the occurrence of camouflaging remains an unresolved problem since the pROST has no means of exploiting the spatial correlation of foreground pixels [42]. In [36] [52], Rodriguez and Wohlberg proposed a fully incremental principal component pursuit (incPCP) method with an extremely low memory footprint, and a computational complexity that allowed real-time processing [36], but the group sparsity property of the foreground was still ignored. As reported in [36], in gray-scale, the performance of incPCP is inferior to traditional PCP [20] on data-set I2R.

In the most recent works [32] [33] [34], Pang \textit{et al.} [34] proposed an incremental low-rank based method which is incorporated with saliency map (SM-RPCA). The guidance of satisfying saliency map can effectively make estimated foreground to have high-level semantic objects and less false alarms, while bad saliency map has a negative influence (e.g., intermittent object motion) on the result. Also, in [32], the salient map was utilized to estimate the support of the foreground, and a tensor based low-rank and saliently fused sparse decomposition model (TLSFSD) is proposed to detect foreground [32]. In [33], Erfanian Ebadi \textit{et al.} proposed a dynamic superpixel structured-sparse (DSPSS) algorithm. Different to the pre-defined groups using in [22], DSPSS dynamically estimates the natural structure of objects in the scene via a superpixel generation step [33], and then a tree structured-sparsity inducing norms is utilized to impose spatial coherence on these regions. Although DSPSS is a batch model, it achieved top performance in comparison with the state-of-the-art alternatives on three datasets. A more detailed discussion of RPCA based techniques can be found in recent survey [17].

B. Sparse Signal Recovery

The same problem also exists in the sparse signal recovery\textsuperscript{1} based methods. Most of previous models did not consider any prior knowledge of the spatial distribution of outliers (foreground). To solve this problem, ProxFlow [31] used a structured regularizer to encode the prior that non-zero entries of sparse signals should be in a group structure. In [21], Huang \textit{et al.} proposed a method by making use of group clustering priors, called dynamic group sparsity (DGS). The fused sparsity was proposed to deal with the group property of foreground in [26]. Also in [28], the group lasso was used to model the foreground. However, to the best of our knowledge, existing methods only focus on spatial group sparsity, and few works have exploited both spatial and temporal group clustering priors for better sparse recovery.

As opposed to the RPCA, methods using sparse signal recovery process observations sequentially, which is an instinctive online processing modeling. Moreover, for RPCA-related methods, final foreground mask is dependent by a thresholding operation to remove background noises. In sparse signal recovery, foreground pixels are non-zero coefficients with \( K \) sparsity and we can pick \( K \) sparsity non-zero coefficients to extract foreground. This will not need a thresholding operation and make our method robust to background noises.

To the best of our knowledge, the performances of exist online RPCA methods without batch initialization and any pre or post processing are still inferior to traditional batch-based RPCA (please see the experiments in Section V). However, it is noted that sparse signal recovery based methods require a training sequence that not contain any foreground objects (foreground-free). While in many practical problems, a well-defined training sequence may not be easily obtained. Fig. 2 shows such a sequence from an indoor surveillance video, where crowds of people are always in the scene. “Ghosts” appear in the background image (see the red rectangle) produced by the ProxFlow [31] and DGS [21], and result in incomplete foreground segmentation.

In fact, in most of the current sparsity signal recovery algorithms [21] [53] [54], the sparsity degree must be given in advance. Otherwise the method has to initialize the lower bound of the sparsity range to zero, and search for the sparsity degree by incrementing the step size until certain halting conditions are satisfied. It is very difficult to maintain the balance between the computational cost and the recovery performance [16]. Therefore, a very small step size is commonly adopted to confirm recovery performance, which results in a very low processing speed.

III. Spatio-Temporal Group Sparsity Recovery

Based on the assumption of sparse signal recovery [35] [21], at time \( t \), a new coming video frame\textsuperscript{2} \( y_t \in \mathbb{R}^m \) can be modeled as a sparse linear combination of \( n \) preceding image frames \( D = [y_{t-n}, \ldots, y_{t-1}] \in \mathbb{R}^{m \times n} \) plus a sparse outliers term \( e \in \mathbb{R}^m \):

\[
y_t = Dx + e + \epsilon \tag{2}
\]

where \( x \in \mathbb{R}^n \) is the coefficient vector, and \( \epsilon \) is the additive noise. The term \( Dx \) accounts for background parts, and \( x \) should be a \( k_x \) sparse vector and \( k_e \ll n \). Also, the sparse outlier \( e \) corresponds to the foreground in \( y_t \). In this paper, we utilize an identity matrix \( I \in \mathbb{R}^{n \times m} \) as the complete dictionary for the foreground, then Eq. (2) can be rewritten as:

\[
y_t = \Phi z + \epsilon \quad (\Phi = [D, I], \quad z = \begin{bmatrix} x \\ e \end{bmatrix}) \tag{3}
\]

where \( \Phi \in \mathbb{R}^{n \times (n+m)} \) and \( z \in \mathbb{R}^{n+m} \), we denote columns of \( \Phi \) by \( \varphi_1, \ldots, \varphi_{n+m} \). The essential purpose of the method is to estimate the coefficient \( z \) from a training data-set by minimizing a given objective function. Therefore, background subtraction is formulated as the following sparse signal recovery problem:

\[
\min_z \|y_t - \Phi z\|^2 + \lambda \|z\|_0 \tag{4}
\]

where \( z \) should be \( (k_x + k_e) \) sparse vector and \( (k_x + k_e) \ll (n + m) \). It is known that Eq. (4) is an NP-hard problem due to the non-convexity of \( \ell_0 \)-norm. To find the optimal solution, many methods are introduced. One class of algorithms tries to seek for the sparsest solution by

\textsuperscript{1}The procedure of sparse signal recovery for background subtraction is detailed in Section III.

\textsuperscript{2}\( y_t \) is reshaped from 2D image frame, and \( m = w \times h \), where \( w \) and \( h \) are the size of frame.
performing the basis pursuit (BP) [55] based $\ell_1$-Minimization instead of $\ell_0$ [19].

$$\min_z \|y_t - \Phi z\|^2 + \lambda \|z\|_1$$  \hspace{1cm} (5)

BP provides strong guarantees and stability, but because of Linear Programming, this method does not yet have strong polynomially bounded runtimes [54]. Another well-known algorithm is the orthogonal matching pursuit (OMP) [56], which uses the observation vector to iteratively calculate the support of the signal. During each iteration, the OMP selects the largest component of the observation vector to be in the support:

$$v^c = \arg\max_{l=1, \ldots, m+n} \left| \langle y_{r}^{c-1}, \varphi_l \rangle \right|$$  \hspace{1cm} (6)

$$T^c = T^{c-1} \cup v^c$$  \hspace{1cm} (7)

where $c$ is the iteration counter, $y_r$ is the residual of the signal, $\langle \cdot, \cdot \rangle$ denotes the inner product, and the support set $T$ is defined as the set of indices corresponding to the non-zero elements of the signal, which can be used to reconstruct signal. It also means that OMP chooses the column that most strongly correlates with the remaining part of $y$. Then OMP subtracts its contribution and iterates on the residual.

Although the OMP is a very fast greedy pursuit algorithm, it lacks provable recovery guarantees and requires more measurements for perfect recovery [54] [53] [21]. In order to close this gap, the backtracking steps were incorporated in the subspace pursuit (SP) [53] and compressive sampling matching pursuit (CoSaMP) [54], which select many columns (through maintaining a list of indices) of the observation vector at each iteration.

$$T_{\Delta} = \text{supp}(\Phi^* y_{r}^{c-1}, K)$$  \hspace{1cm} (8)

$$T^c = T^{c-1} \cup T_{\Delta}$$  \hspace{1cm} (9)

where $\text{supp}(X, K)$ returns the support of the largest $K$ magnitude elements in $X$, $K$ is the sparsity degree, and $*$ stands for the matrix transposition. From Eq. (6) to (7), we can find in OMP that once an index is added into support set, it remains in this set throughout the remainder of the process. On the other hand, in the SP and CoSaMP algorithms, a list of indices with size $K$ is maintained and refined during each iteration. An index, which is considered reliable in some iteration but shown to be wrong at a later iteration, can be added into or removed from the estimated support set freely [53]. SP and CoSaMP provide strong guarantees as the $\ell_1$ approach, and the computation complexity is comparable to that of the greedy pursuit algorithms [21]. However, all of above methods treat each non-zero coefficient independently, and the confidence score in Eq. (6) and (8) is only calculated by inner product (projection).

$$p(l) = \left| \langle y_{r}^{c-1}, \varphi_l \rangle \right|$$  \hspace{1cm} (10)

Neither of them take into account any possible relations among the subsets of the entries in the sparse matrix.

In contrast, we propose a new greedy sparse recovery algorithm, which is motivated by the observation that in real videos, the sparse outliers treated as foreground are not randomly distributed, but often have group properties of spatial coherence and temporal contiguity. Based on the observation, we assume that group signals live in the union of subspaces, which implies that, if a point lives in the union of subspaces (see the red point in Fig. 1), its neighboring (spatial and temporal) points would also live in this union of subspaces with high probability, and vice versa. Inspired by [21] [54] [53], we propose a Spatio-Temporal Group Sparsity (STGS) recovery algorithm that includes five main steps in each iteration (see Algorithm 1).

1) **Find new support elements.** The objective of signal reconstruction is to identify the locations of the largest components in the target signal. At each iteration $c$ of STGS, the current approximation induces a residual $y_{r}^{c-1}$, the part of the target signal that has not been approximated. STGS first computes the subspace projection coefficients by the inner products of the residual signal with the columns of $\Phi$.

$$\mathcal{P} = \Phi^* y_{r}^{c-1}$$  \hspace{1cm} (11)

Then, STGS tries to select a set of candidate non-zeros ($K$ columns of $\Phi$) that are most correlated with the residual. This idea is similar to the SP and CoSaMP algorithms. However, in the task of background subtraction, the relationship of these non-zeros (foreground) is ignored by the SP and CoSaMP. In fact, these non-zero coefficients (foreground pixels) are clustered into $G$ ($G < K$) groups. Therefore, instead of only using sparsity (inner product itself), STGS calculates the confidence score by combining each entry with its neighbors in both spatial and temporal domains (see Eq. (14) and its explanation in Step 3). More specifically, if a pixel is a foreground, its neighboring (spatial and temporal) pixels would also belong to foreground, and vice versa. Therefore, STGS obtains the new support sets $T_{\Delta}$ that reflects the current residual in terms of the order statistics of the confidence score.

2) **Update the support sets.** STGS maintains a list $T^{c-1}$ with $K$ columns of $\Phi$ (a temporal solution with $K$ non-zero
entries), and adds an additional set \( T_\Delta \) of \( K \) columns (candidate non-zeros) obtained from the last step. Therefore, the size of merged support sets \( \tilde{T}^c \) should be \( K \leq \text{size}(T^c) \leq 2K \).

3) **Compute the representation.** STGS calculates the non-zero signal coefficients (representation) by applying the pseudo-inversion process.

\[
z_v = \Phi_{\tilde{T}_v}^\dagger y \tag{12}
\]

where the matrix \( \Phi_{\tilde{T}_v} \) consists of the columns of \( \Phi \) with indices \( T^c \), and \( \Phi_{\tilde{T}_v}^\dagger \) denotes the pseudo-inverse of the matrix \( \Phi_{\tilde{T}_v} \).

\[
\Phi_{\tilde{T}_v}^\dagger = (\Phi_{\tilde{T}_v}^* \Phi_{\tilde{T}_v})^{-1} \Phi_{\tilde{T}_v}^* \tag{13}
\]

Similar to Step 1), STGS calculates the confidence score of each entry, by not only using the coefficient itself, but also combining its neighbors in both spatial and temporal domains:

\[
p(i, j) = p^2(i, j) + \sum_{a=1}^{i+\tau} \sum_{b=1}^{j+\tau} \omega^2(a, b)(1+\sigma)p^2(a, b) \tag{14}
\]

Please note that \( z_v \) (and \( \mathcal{P} \)) is a 1-D vector, while we use \( p(i, j) \) rather than \( p(l) \) as in Eq. (10) to represent entries of \( z_v \). The reason is that we want to preserve the 2-D structure and neighboring information of pixels\(^3\). In Eq. (14), \( \tau \) controls the size of spatial neighbor. According to the experimental results, an 8-connected neighborhood is proved to be satisfactory for our method (see experiments in Section V), then we set \( \tau = 1 \). The parameter \( \sigma \) is used to encourage temporal continuity, which is very important for dealing with noise and dynamic background variations. Because noise or background motion is usually smaller, shorter and more irregular than real foreground object motion, and foreground object commonly has the property of temporal consistency. From experimental results (see experiments in Section V), if pixel \((a, b)\) in the last frame is foreground, then we set \( \sigma = 0.2 \), otherwise \( \sigma = 0 \).

\(^3\)In Eq. (3) \( z = [x, e]^T \) is a 1-D vector. The \( e \) part (of length \( m \)) can be regarded as (or reshape to) a \( w \times h \) matrix \((w \text{ and } h \text{ are the frame size, } m = w \times h)\), so entries of \( e \) can be regarded as a pixel with 2-D coordinates in an image.

It is noticed that the weight of neighbors \( \omega \), which is used to control the balance between the sparsity prior and the group clustering prior. The setting strategy of \( \omega \) will be subsequently reported.

4) **Prune small entries in the representation.** In this step, STGS produces a new approximation (support sets \( T^c \) with \( K \) columns of \( \Phi \)) by retaining only the \( K \) largest entries. By enforcing the spatio-temporal group sparsity constraint mentioned in the last step, the signal pruning process can be accelerated, since the sparse signals have been significantly reduced to a narrower union of subspaces [21]. Moreover, false alarms caused by noise can be reduced since STGS only permits certain combinations of its support set rather than any random entries. Results from the first experiment (Fig. 4) have shown better foreground recovery when compared to related methods.

It is also noted that STGS extends the backtracking strategies from SP and CoSaMP. However, in traditional greedy pursuit algorithms, such as the famous OMP, the method selects only one (or several) column(s) during each iteration that most strongly relates to the residual, and once this index is added into the support set, it cannot be removed throughout the remainder of the process. As a result, these strict inclusion rules are needed to ensure that a significant fraction of the newly added index (column) belongs to the correct support set [53]. On the contrary, STGS maintains a temporal list with \( K \) non-zero entries, and in each iteration it adds an additional set of \( K \) candidate non-zeros that are most correlated with the residual, and then STGS refines this list back to \( K \) elements. This recursive refinements of the estimate of the support set will lead to subspaces with strictly decreasing distance from the measurement vector [53].

5) **Update the residual.** In this step, STGS first calculates a new projection \( y_p \) of \( y \) onto \( \text{span}(\Phi_{\tilde{T}_v}) \)

\[
y_p = \text{proj}(y, \Phi_{\tilde{T}_v}) := \Phi_{\tilde{T}_v}^\dagger y \tag{15}
\]

The space spanned by the columns of \( \Phi_{\tilde{T}_v} \) (support set) is
Algorithm 2: Two-pass framework for background subtraction.

**Input:** Time \( t \), video frame \( y_t \).

**Output:** \( \hat{e}_t \): sparse approximation of foreground \( e_t \).

**First-pass:**

- Foreground detection [10]
  - Results: rough estimation of foreground, including foreground number \( k_e \) and weight \( \hat{w}_t \) by Eq. (20), and background image \( b_{f_1} \).

**Second-pass:**

- **Background part:**
  - Init: Using \( b_{f_1} \), and Eq. (19) to update background dictionary \( D_t, C = 0 \) (iterations of outer loop), support set \( T^e = \emptyset \), set \( \Phi = D_t, y = y_t \), \( \omega = 0 \), and \( K = 1 \) for input of STGS
  - repeat
    - Perform STGS recovery with sparsity \( K \) to obtain \( z_{T^e} \)
    - update \( T^e, z_{T^e} = \Phi^T_{T^e} y, \) and \( K = K + \Delta k \)
  - until halting criterion \( \| z_{T^e} - z_{T^e-1} \|_2 \leq \varepsilon \) is true
  - Results: Sparse coefficient of background \( x = z_{T^e} = \Phi^T_{T^e} y \), foreground candidate (residual) \( y_t - D_t x \).

- **Foreground part:**
  - Init: \( C = 0 \), support set \( T^c = \emptyset \), \( \Phi = I \) (identity matrix), \( y = y_t - D_t x, \omega = \hat{w}_t \), and \( K = k_e \) for input of STGS
  - Perform STGS with sparsity \( K + \Delta k \) and \( K \) to obtain \( z_{T^c} \) and \( z_{T^r} \) respectively, and now \( C = 2 \)
  - if \( \| z_{T^c} - z_{T^r} \|_2 \leq \varepsilon \) is satisfied
    - repeat
      - \( C = C + 1 \)
      - \( K = K - \Delta k \)
      - Perform STGS with sparsity \( K \) to obtain \( z_{T^c} \)
      - update \( T^c, z_{T^c} = \Phi^T_{T^c} y \)
    - until halting criterion \( \| z_{T^c} - z_{T^c-1} \|_2 \geq \varepsilon \) is true
    - \( T^c = T^{c-1} \) (true)
  - else
    - repeat
      - \( C = C + 1 \)
      - \( K = K + \Delta k \)
      - Perform STGS with sparsity \( K \) to obtain \( z_{T^c} \)
      - update \( T^c, z_{T^c} = \Phi^T_{T^c} y \)
    - until halting criterion \( \| z_{T^c} - z_{T^c-1} \|_2 \leq \varepsilon \) is true

- Form final solution: \( \hat{e}_t = \Phi_{T^r} y \) (true)

Denoted by \( \text{span}(\Phi_{T^r}) \) [53].

\[
y_p = \text{resid}(y, \Phi_{T^r}) := y - y_p \tag{16}
\]

These five steps are repeated until the halting criterion is triggered, and STGS uses the final support set \( T^e \) to get the sparse non-zero entries.

\[
\hat{z}_{T^e} = \Phi^T_{T^e} y \tag{17}
\]

The halting criterion recommended in [53] is adopted. The whole algorithm is summarized in Algorithm 1.

Then, STGS computes the residue vector, the part of the target signal that has not been approximated.

The closest work to ours includes SP, CoSaMP, and DGS. Nonetheless, the group sparsity property of the foreground signals has been ignored by SP and CoSaMP. Also in DGS, the temporal information is missing, and it is difficult to handle complex background variations without temporal information (see experimental results in Section V). In addition, there are two other issues with these methods that need to be addressed. Firstly, they require the sparsity degree is given in advance. However, knowing of the sparsity before even recover foreground makes it a typical chicken-egg problem. To ensure the quality of recovery, commonly, a multiple iterative approximation procedure is utilized, which results in a very low processing speed. Secondly, they assume that the background should be modeled as a sparse linear combination of atoms from dictionary \( D \) in Eq. (2), and that they obtain the atoms directly from \( n \) preceding image frames, which contain both background and unwanted foreground pixels. There is no mechanism of background dictionary learning in these methods, which leads to inaccurate background estimation and poor foreground detection performance.

IV. TWO-PASS FRAMEWORK FOR ON-LINE FOREGROUND DETECTION

Inspired by [38] [40] [28], we propose a two-pass process to solve the aforementioned problems. The framework is shown in Fig. 3. In the first-pass we employ a fast background modeling, which can rapidly identify the likely regions of foreground. This leads to several advantages:

- The obtained number of foreground pixels could be used to estimate the bound or narrow the range of the sparsity degree. Thus, the number of iterations will be significantly reduced.
- The location of foreground pixels could be used as the prior knowledge of group structure.
- The obtained background image could be used as a candidate for atoms in background dictionary.

Thus, in the first-pass, we employ a pixel intensity sampling based method, which records a history of recently observed pixel values, and uses a random aggregation mechanism to update the background samples. The spirit is the same with ViBe [10]. The high frame-rate of processing, about 200 FPS on images of size \( 640 \times 480 \) as reported in [10] is the main reason why we utilize this method. Although the foreground detection performance of ViBe is not top level according to the evaluation results in [37]. In fact, we tested several different methods on the first-pass, which are subsequently reported. The experimental results show that regardless of the detection precision of the first-pass, the influence on the final result is constrained. It means that a rough estimation of foreground in first-pass is sufficient for our system.

In the second-pass, we use the background image \( b_{f_1} \) from the first-pass (to update background dictionary \( D \)) rather than the original video frame as used by traditional methods. We then discuss the learning strategy of background dictionary. It is well known that many previous methods [21] [31] utilized the first-in and first-out policies to update their dictionaries. While a good dictionary should contain samples (atoms) from the recent past background images, older samples should not necessarily be discarded. Motivated by [10], instead of removing the oldest atom from the dictionary, we randomly choose the atom to be discarded according to a uniform probability density function (see Eq. (19)). The reason for doing this is to enable the handling of a wide range of events in the background scene. According to the random
updating mechanism, the probability of an atom being present in the dictionary at time $t$, and preserved after the next update ($t + 1$) is $(N - 1)/N$, where $N$ is the number of atoms in the dictionary. Thus, the probability of this atom still being alive in the dictionary after $dt$ time, is equal to

$$Pr(t, t + dt) = e^{-\ln(\frac{N}{N-1})dt}$$  

As presented in [10], this expression shows that the probability of an atom being present for the interval $(t, t+dt)$ is independent of $t$, assuming that it was included in the dictionary prior to time $t$. More specifically, an atom is not to be discarded after a fixed number of frames, but to be extended for the expected lifespan to keep the diversity of the background state. In many practical video scenes, the old history is meaningful, especially for the multi-mode dynamic background (experiments are reported in Section V).

On the other hand, this random update scheme extends the time windows covered by the background dictionary, without increasing the atom numbers, which is critical for the control of computational burden and memory requirement. Moreover, background modeling often encounters challenges from sudden background changes, like variations in illumination and other scene parameters that alter the appearance of the background. In order to rapidly respond to the sudden changes, we adopt $Th$ as the threshold. Once the foreground ratio ($F_{num}/m$, $F_{num}$ is the number of the foreground, and $m$ denotes the size of the frame) is greater than $Th$, then the origin frame $y_t$ is used to update the background dictionary rather than background image $bf_t$ produced during the first-pass. Thus, at time $t$, the update of the background dictionary is denoted as:

$$D_t = D_{t-1} \{ \forall b_i \mid b_i = b_t, \text{ with probability } 1/N \}$$

$$\text{(if } F_{num}/m > Th, b_t = y_t \text{; else } b_t = bf_t)$$  

As previously discussed, we use a parameter $\omega$ to control the balance between the sparsity prior and the group clustering prior for the outlier part $e$ in Eq. (2). It also means that if the degree of the group clustering is higher in the sparse signal, $\omega$ should be assigned a greater value, and vice versa. Based on that observation, we propose an adaptive setting of $\omega$ for image regions with distinct properties for each frame. In our method, the frame is divided into small blocks with the same size $B_s$, and in each block we count the number of foreground $BF_i$ from first-pass. Then, at time $t$, for the pixels in block $i$:

$$\omega_{B_i,t} = BF_{i,t}/B_s$$  

Up to now, we have obtained all of the inputs needed in the second-pass, including the weight $\hat{\omega}_f$ (for the $x$-related (background) part), $\omega$ can be set as zero), background dictionary $D$, and the estimation of foreground pixel number $\hat{k}_e$ from first-pass. As mentioned in Sections II and III, the conventional sparse signal recovery based methods assume that sparsity $K$ has been given in advance. However, knowing about the sparsity (foreground pixels number) even before segmenting the video frame seems to make the problem as one of many chicken-egg problems in computer vision. Commonly, these methods employ an iterative process to approximate the sparsity. More specifically, there are two loops in approximation. During each stage (inner loop), sparse data are iteratively optimized with the fixed sparsity number until the halting condition within the stage is satisfied. Then the next stage (outer loop) is switched to after adding step size $\Delta k$ into the current sparsity number [21] [54]. The range of the sparsity should start from zero. Thus, the gap between iteration starting value and ending one is $K$. Thanks to the estimation of the foreground pixel number $\hat{k}_e$ from the first-pass, on the foreground part, we can initialize sparsity number (outer loop) as $\hat{k}_e$. The gap between sparsity $K$ and $\hat{k}_e$ is

$$d = |K - \hat{k}_e|$$  

where $d$ is much less than $K$. From above, it can be concluded that the times of iteration could be significantly reduced by the proposed two-pass framework.

We also utilize two loops in the second-pass, while for the background part, the sparsity degree $k_e$ could start from 1 in the outer loop. On the foreground part, we need to make sure that $\hat{k}_e$ is less or greater than sparsity degree $K$. Therefore, we should check whether the halting condition of outer loop is satisfied after adding the step size $\Delta k$ at the first iteration. If it is satisfied, we need to subtract a step size from $\hat{k}_e$, and the whole iterative process would stop whenever the halting condition is not satisfied, and vice versa.

We summarize the algorithm sketch of our two-pass framework in Algorithm 2. The algorithms such as GRASTA and GOSUS also utilized two-stages (pass) framework to achieve the online processing, but they focus on the low-rank property of the matrix composed by video frames, and are RPCA based method which need a batch initialization (first) stage to obtain the background estimation and ensure the quality of background recovery in the second-stage. Otherwise, these methods cannot obtain a satisfactory foreground detection results. Different to GRASTA and GOSUS, our method is a sparse signal recovery based method, which processes images in a frame by frame manner, the first-pass is to rapidly estimate the background and ensure the quality of background recovery in the second-stage. Otherwise, these methods cannot obtain a satisfactory foreground detection results. A further distinction is that GRASTA ignored the relationship between pixels of foreground since it used the $\ell_1$-norm as a convex relaxation of the ideal sparsifying function. In GOSUS, the structured-sparsity is utilized to impose structural contiguity in spatial domain, but the group cluster priors in temporal domain are missing. On the contrary, we take into account the group properties of the foreground signal in both spatial and temporal domains.

V. EXPERIMENT

A. Effects of separate steps

In order to better understand the performance of the proposed algorithm, we analyze the effects of separate steps one by one. First, to evaluate the effectiveness of our spatio-temporal group sparsity recovery (STGS), we only used STGS (excluding use of the first-pass, and the weight $\omega$ can be set at a fixed value such as 0.5) to detect foreground, and compared its performance with two sparse signal recovery related methods subspace pursuit (SP) [53] and DGS [21]. The first sequence
tested here is from [7], which contains a camera motion caused by wind. This is a commonly used test video for evaluating the effectiveness of dynamic background modeling techniques. Fig. 4 (a) shows a qualitative evaluation of algorithms. We can see that SP produces plenty of false positives, and DGS lost a few of the real foreground pixels. It is not difficult to find that SP is the worst since it only uses sparsity without consideration of any foreground group clustering. By using the group sparsity on both spatial and temporal domains, the proposed STGS yields a superior performance when compared to DGS. For the sake of comparison, we compared six state-of-the-art RPCA-based approaches, namely RPCA-PCP (PCP) [20], GOSUS (GOS) [28], TV-RPCA (TV) [29], LSD [22], OR-PCA with MRF (ORM) [44], and GFL [30], also shown in Fig. 4 (a). Clearly, STGS archived the best foreground results. This is confirmed by the performance of the F-Measure in Fig. 4 (b).

Next, to verify our claim that the final result was narrowly affected by the performance of the first-pass, we chose three different methods for the role of first-pass, namely PBAS [11], ViBe [10] and GMM [2]. From Fig. 4 (c) we can easily see that PBAS, GMM and ViBe obtain the best, inter-mediate and worst results, respectively. Although the performances of these three methods are quite different, after the second-pass, their final results are similar, as shown in Fig. 4 (c). Furthermore, it is observed that the two-pass result is better than that of STGS, since weight $\omega$ obtained from first-pass is more accurate than a fixed constant for expressing the group structure of the foreground. To deal with that sequences with frame size $360 \times 240$, SP needs around 1820 ms, DGS 1190 ms, STGS 780 ms, and the proposed two-pass framework only about 110 ms. The proposed method achieved the best foreground detection performance with far fewer computations.

Moreover, in order to verify the effectiveness of the random update strategy of the background dictionary learning, we compared its performance with the first in first out (FIFO) strategy that was commonly utilized by conventional approaches. The sequence evaluated here is “Fall” from the data-set

Fig. 4. Comparison on sequence from [7] for demonstrating the effects of separate steps.

Fig. 5. Comparison of random update (random) and first in first out (FIFO) schemes for background dictionary learning. (a) False positive performance of two dictionary learning schemes on “Fall” [37]. (b) F-measure performance of two dictionary learning schemes on “Fall” [37].
CDnet [37], which contains a typical dynamic background, waving trees caused by wind. In the sequence, the video scene is empty from frames 1000 to 1450. As shown in Fig. 5(a), false positives, the background pixels incorrectly classified as foreground, caused by waving trees were detected. It can be seen that the number of false positives when using random update is much less than when using FIFO. In Fig. 5(b), we also present the F-measure results of two different update schemes, and the frames are from 1451 to 1550, which contain interesting foreground (moving vehicles) and background motion (waving trees). We can see that the performance of the random update strategy is consistently better than that of FIFO.

B. Qualitative and quantitative evaluations

For the sake of comparison, experiments have been conducted qualitatively and quantitatively on real sequences from the I2R\(^6\) data-set [4] and CDNet 2012\(^7\) data-set [37]. We compared the proposed algorithm with 15 methods, including RPCA-PCP (PCP) [20], LBD [24], DGS [21], GOSUS (GOS) [28], DECOLOR (DEC) [25], TV-RPCA (TV) [29], LSD [22], GFL [30], incPCP (IPC) [36] [52], OR-PCA with MRF (ORM) [44]. PCP, LBD, GOSUS, DECOLOR, TV-RPCA, LSD, GFL, incPCP, and ORM are state-of-the-art RPCA based algorithms. DGS is the state-of-the-art sparse signal recovery algorithm. GOSUS, incPCP, ORM and DGS are incremental (online) models whereas PCP, LBD, DECOLOR, TV-RPCA, LSD, GFL are batch models. For all comparison algorithms, the results are generated from the source codes released by original authors, and the parameters use the default settings in their approaches. In ORM, a 5 × 5 median filtering was applied as a post-processing step on binary foreground mask [44]. In other experiments, post-processing was not applied for fair comparisons (i.e., any morphological operations were not conducted).

Fig. 6 shows the detected foreground masks on videos from the I2R data-set [4]. It includes nine sequences, namely Bootstrap, Campus, Curtain, Escalator, Fountain, Hall, Lobby, Shopping Mall, and Water Surface. These videos include a wide range of challenging scenarios: high dynamic background, outdoor and indoor environments, sudden light changes, etc. In the figure, the first three rows present the indoor scene where people are shown walking in and out. As discussed in Section III, a key distinction between the proposed method and DGS is the assumption about the availability of training sequences with/without foreground objects. Sparse signal recovery based methods such as DGS and ProxFlow require a set of background frames without foreground, which is not always available for surveillance of crowded scenes. Therefore, it is difficult for these methods to recover an accurate background. Instead, RPCA-related methods can estimate a clean background from occluded data, while DECOLOR tends to produce a lot of false alarms due to the smoothness constraint imposed on the foreground. On the other hand, LBD, PCP, and incPCP failed to obtain complete foreground without a group or structured sparsity constraint. GOSUS, GFL, and ORM cannot get the complete foreground mask. The proposed method performed similarly to LSD and TV-RPCA, and are found better than the others. The next five rows illustrate scenes with dynamic background caused by motion of tree branches, curtain, fountain, water surface, and escalator. It is not difficult to find out that GOSUS, PCP, LBD, TV-RPCA, GFL and incPCP had difficulty eliminating false positives. DGS and GFL lost plenty of foreground pixels in the “Campus” and “Curtain”. DECOLOR, PCP, and incPCP failed

---

\(^6\)perception.i2r.a-star.edu.sg.

\(^7\)www.changedetection.net.
to detect the person in the “WaterSurface”, and DECOLOR produces over-smooth results in “Curtain”. In the video “Escalator”, the motion of escalators makes the background motion regions difficult to be removed. Furthermore, the background model is hard to establish if there is a steady stream of human flow in the scenes. The results show that our method, LSD, and ORM provided better detection results in handling such non-stationary backgrounds than others can. The last row shown in Fig. 6 is a sequence with a light being turned on and off. What is noteworthy is that DGS, GOSUS had difficulties to deal with such sudden light variations. This is not surprising as the training sequences composed of background variations are required for these sparse signal recovery and batch initialization based methods. We can see that the proposed method removed background motions effectively and detected the complete silhouette.

For better evaluation, we test the proposed method on another widely used data-set CDnet [37], which consists of nearly 90000 frames in 31 video sequences representing six categories selected to cover a wide range of challenges. These six categories are: Baseline (BL), Camera Jitter (CJ), Dynamic Background (DB), Intermittent Object Motion (IM), Shadow (SD) and Thermal (TM). Due to space constraints, we show results on ten sequences in Fig. 7, which represent typical critical situations for the video surveillance systems. The first row in the figure shows the results of “Highway” from Baseline category. Clearly, all algorithms performed well on this fairly easy video. The second and third rows show the results of “Boulevard” and “Traffic” (Camera Jitter category). It can be seen that PCP, LBD, TV-RPCA, LSD, and incPCP are sensitive to the motion of the camera, and produce many false positives. DGS and GFL could not get the complete foreground results of the car in “Traffic”, and DGS also fails in “Boulevard”. GOSUS is able to tolerate the camera jitter, while achieving incomplete detections in “Traffic”. As can be seen, our method, ORM and DECOLOR achieve complete foreground detection with few false positives. The next two rows present the results of “Canoe” and “Fall”. It appears that PCP, TV-RPCA, and incPCP were hard to handle dynamic background with water wave and tree motion. In addition, LBD could not manage to judge background motion. In LBD, the $f_{2,1}$-norm is adopted to detect outliers with column-wise sparsity. However, the block-sparsity property still has no structured information to model any grouped sparse outliers. DECOLOR could detect most of foreground pixels, but it produces many false alarms due to the smoothness constraint imposed on the foreground shapes. Moreover, DECOLOR fails to correctly detect the “canoe”. GOSUS filters out dynamic background variations, but loses many positive foreground pixels. It is noticed that ORM is one of the top-performing methods, according to the evaluation results on the CDnet [37]. The proposed method and ORM are able to tolerate most background motions, while ORM might lose sensitivity in completely detecting real foreground, as in “Fall”. The sixth and seventh rows show the results of the Intermittent Object Motion category, which depicts background objects that are moving away, and foreground objects that are stopping for a short while and then moving away. We can see that GOSUS, LBD, DECOLOR, incPCP, and LSD fail to detect the abandoned box in video “Sofa”. In addition, TV-RPCA, GFL, DGS and GOSUS miss the real foreground (truck) at the parking lot when the background objects move away (car) in “Parking”. As can be seen, in the next two rows, PCP and ORM produce plenty of false positives due to the
TABLE I

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Measure</td>
<td>57.35</td>
<td>58.74</td>
<td>60.25</td>
<td>70.01*</td>
<td>73.08</td>
<td>74.89</td>
<td>75.94</td>
<td>76.22</td>
<td>77.11</td>
<td>77.67</td>
<td>77.81</td>
<td>79.46†</td>
<td>87.51†</td>
<td>88.73†</td>
<td>89.21†</td>
</tr>
</tbody>
</table>

† As reported in [44] [45], a 5 × 5 median filtering was applied as a post-processing step on binary foreground mask.
§ As reported in [47] [33], the superpixel was utilized as the input of their model.
* Experimental results under comparison are come from paper [32].

TABLE II

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>–</td>
<td>–</td>
<td>84.64</td>
<td>86.67</td>
<td>91.09</td>
<td>86.98</td>
<td>91.33</td>
<td>86.26†</td>
<td>91.33</td>
<td>92.15</td>
<td>92.67</td>
<td>93.69†</td>
<td>93.47</td>
<td>94.69†</td>
<td>96.64†</td>
</tr>
<tr>
<td>CI</td>
<td>–</td>
<td>–</td>
<td>66.09</td>
<td>66.59</td>
<td>72.18</td>
<td>76.27</td>
<td>74.82</td>
<td>75.25†</td>
<td>73.52</td>
<td>77.76</td>
<td>78.19</td>
<td>84.57†</td>
<td>76.22</td>
<td>85.19†</td>
<td>86.62†</td>
</tr>
<tr>
<td>DB</td>
<td>71.50</td>
<td>61.95</td>
<td>59.43</td>
<td>69.41</td>
<td>58.17</td>
<td>65.92</td>
<td>72.10†</td>
<td>77.07</td>
<td>70.84</td>
<td>71.73</td>
<td>77.65†</td>
<td>82.36</td>
<td>80.55†</td>
<td>90.57†</td>
<td></td>
</tr>
<tr>
<td>IM</td>
<td>–</td>
<td>57.32</td>
<td>63.16</td>
<td>53.71</td>
<td>63.16</td>
<td>58.78</td>
<td>56.69†</td>
<td>57.67</td>
<td>59.45</td>
<td>67.54</td>
<td>75.49§</td>
<td>67.66</td>
<td>69.88§</td>
<td>78.70§</td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>–</td>
<td>80.15</td>
<td>77.99</td>
<td>76.85</td>
<td>80.26</td>
<td>79.50</td>
<td>74.51†</td>
<td>81.74</td>
<td>83.17</td>
<td>81.58</td>
<td>69.50§</td>
<td>84.61</td>
<td>76.44§</td>
<td>91.77§</td>
<td></td>
</tr>
<tr>
<td>TM</td>
<td>–</td>
<td>59.01</td>
<td>71.83</td>
<td>71.92</td>
<td>70.19</td>
<td>76.32†</td>
<td>70.80</td>
<td>70.81</td>
<td>75.85</td>
<td>79.21†</td>
<td>82.97</td>
<td>81.56§</td>
<td>73.28§</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>–</td>
<td>68.19</td>
<td>70.91</td>
<td>72.86</td>
<td>72.96</td>
<td>73.42</td>
<td>73.52†</td>
<td>75.36</td>
<td>75.70</td>
<td>77.93</td>
<td>80.01†</td>
<td>81.22</td>
<td>81.39§</td>
<td>86.26†</td>
<td></td>
</tr>
</tbody>
</table>

† As reported in [44] [45], a 5 × 5 median filtering was applied as a post-processing step on binary foreground mask.
§ As reported in [47] [33], the superpixel was utilized as the input of their model.

TABLE III

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[320 × 240] × 100</td>
<td>127.37</td>
<td>46.49</td>
<td>17.00</td>
<td>19.22</td>
<td>10.54</td>
<td>9.62</td>
<td>8.93</td>
</tr>
</tbody>
</table>

shadows in “Backdoor” and “Bus Station”. It is noticed that the proposed method could not completely filter out shadows. It is because that we did not introduce any sophisticated procedure to identify shadows. The last row depicts a thermal sequence captured by a far-infrared camera. Our method was able to handle such large-sized foreground object, and obtain much more complete foreground detection than others. What is noteworthy is that DGS failed in “Corridor”. This is because DGS obtains the atoms (background dictionary) directly from n preceding image frames with foreground objects, which leads to inaccurate background estimation.

Qualitatively, the results of the proposed method look better and are the closest to the ground-truth references. A quantitative evaluation provides more solid conclusions on the performance of the proposed method. As a quantitative analysis, we evaluate the performance in terms of F-measure, which is a weighted harmonic mean of precision and recall that can be used to rank different methods. In quantitative evaluation, more state-of-the-art methods are compared, including two online algorithms, OR-PCA with MRF via multiple features (OMM) [45], SODSC (SOD) [47], and three most recent algorithms DSPSS (DSP) [33], SM-RPCA (SM) [34], and TLSFSD (TLS) [32]. As reported in OMM [45], a 5 × 5 median filtering was applied as a post-processing step on binary foreground mask. Also, it is noticed that the superpixel was utilized as the input in SODSC [47] and DSPSS [33]. These pre or post processing would undoubtedly improve the accuracy of foreground detection (while the proposed and other algorithms are just using the pixels value and without any refinement procedure). Like most of other algorithms, in this paper, we just employ pixels value as the input, without using any refinement procedure. In Table I, the average F-measure performance of methods on data-set I2R [4] are compared. It can be seen, in all of the eleven methods without any pre or post processing, our model has achieved the best numerical value, even though against to most recent methods, such as TLSFSD [32] and SM-RPCA [34]. Also, under the same conditions, as shown in Table II, our method obtained the best average metrics on the CDnet [37] data-set. Here, we cannot present complete metrics of SM-RPCA [34] and TLSFSD [32] in Table II, since the source code of these methods are not publicly available, and only a fraction of the results on CDnet were reported in [32].

VI. Conclusion

Our method has been implemented in C++ and MATLAB. Tests were executed on a PC with a 2.6 GHz CPU and 8 GB of RAM. Finally, to complete our analysis, in Table III we present the computational time of seven online (incremental) processing methods. The computational time is recorded as the average processing time on one hundred frames with different resolutions. In the methods evaluated above, GOSUS (GOS), incPCP (IPC), ORM, OMM, SOD, and DGS are online models. SM-RPCA [34] is also an incremental method. But the source code is not publicly available now, and no computational time with different resolutions was reported in [34]. The rest of methods are batch models. Table III shows that the proposed method is very close to the processing speed.
of incPCP, and faster than the others. Our method achieved nearly 22 FPS at a resolution of 160×128, which is close to performing real-time processing.

In this paper, we have proposed a spatio-temporal group sparsity recovery method for background subtraction taking into account the group properties of the foreground signal on both spatial and temporal domains. Moreover, a two-pass framework is proposed to meet the online process requirement. Qualitative and quantitative evaluations on the challenging data-sets demonstrate the superior performance of the proposed method in terms of detection accuracy and time complexity. In the future, the goal will be to further improve computational efficiency to achieve real-time processing, and we plan to develop the moving camera version of STGS that can work efficiently on motion segmentation problems. Furthermore, we plan to apply this method to our research in human behavior recognition and micro gesture analysis.

REFERENCES


Xiaopeng Hong (M’13) received his BEng, MEng, and Ph.D. degree in computer application from Harbin Institute of Technology, Harbin, P. R. China, in 2004, 2007, and 2010 respectively. He has been a scientist researcher in the Center for Machine Vision Research and Signal Analysis, University of Oulu since 2011. He has authored or co-authored more than 30 peer-reviewed articles in journals and conferences, and has served as a reviewer for a few journals and conferences. His current research interests include micro-expression recognition, saliency estimation, pose and gaze estimation, texture classification, visual surveillance, and visual speech recognition.

Xiaohua Huang received the B.S. degree in communication engineering from Huaqiao University, Quanzhou, China in 2006. He received his Ph.D degree in Computer Science and Engineering from University of Oulu, Oulu, Finland in 2014. He has been a scientist researcher in the Center for Machine Vision and Signal Analysis at University of Oulu since 2015. He has authored or co-authored more than 20 papers in journals and conferences, and has served as a reviewer for journals and conference. His current research interests include facial expression recognition, micro-expression analysis, group-level emotion recognition, multi-modal emotion recognition and texture classification.

Ziheng Zhou received his B.Sc. in Computer Science from the Shanghai Jiao Tong University, China in 2002 and his Ph.D. in Computer Vision from the University of Southampton, UK in 2007. He is currently a senior research scientist of the Center of Machine Vision and Signal Analysis at the University of Oulu, Finland. His research interests include computer vision, image processing and human-computer interaction.

Chun Qi received the Ph.D. degree from Xi’an Jiaotong University, Xi’an, China, in 2000. He is currently a Professor and Ph.D. supervisor at School of Electronics and Information Engineering, Xi’an Jiaotong University. His current research interests mainly include image processing, pattern recognition and signal processing.

Guoying Zhao (SM’12) received the Ph.D. degree in computer science from the Chinese Academy of Sciences, Beijing, China, in 2005. She is currently an Associate Professor with the Center for Machine Vision and Signal Analysis, University of Oulu, Finland, where she has been a senior researcher since 2005, and a visiting Professor in Northwest University in China. In 2011, she was selected to the highly competitive Academy Research Fellow position. She has authored or co-authored more than 160 papers in journals and conferences. Her papers have currently over 5700 citations in Google Scholar (h-index 33). She has served as area chairs for FG 2017 and WACV 2017 and is associate editor for Pattern Recognition and Image and Vision Computing Journals. She has lectured tutorials at ICPR 2006, ICCV 2009, and SCIA 2013, authored/edited three books and six special issues in journals. Dr. Zhao was a Co-Chair of 12 International Workshops at ECCV, ICCV, CVPR and ACCV, and two special sessions at FG13 and FG15. Her current research interests include image and video descriptors, facial-expression and micro-expression recognition, gait analysis, dynamic-texture recognition, human motion analysis, and person identification. Her research has been reported by Finnish TV programs, newspapers and MIT Technology Review.