Complex Background Subtraction by Pursuing Dynamic Spatio-Temporal Models

Liang Lin, Yuanlu Xu, Xiaodan Liang, Jian-Huang Lai

Abstract

Although it has been widely discussed in video surveillance, background subtraction is still an open problem in the context of complex scenarios, e.g., dynamic backgrounds, illumination variations, and indistinct foreground objects. To address these challenges, we propose an effective background subtraction method by learning and maintaining an array of dynamic texture models within the spatio-temporal representations. At any location of the scene, we extract a sequence of regular video bricks, i.e., video volumes spanning over both spatial and temporal domain. The background modeling is thus posed as pursuing subspaces within the video bricks while adapting the scene variations. For each sequence of video bricks, we pursue the subspace by employing the ARMA (Auto Regressive Moving Average) Model that jointly characterizes the appearance consistency and temporal coherence of the observations. During online processing, we present a robust method to incrementally update the subspaces to cope with disturbances from foreground objects and scene changes. In the experiments, the proposed method is validated in several complex scenarios, and it outperforms other state-of-the-art background subtraction approaches. The empirical studies of parameter setting and component analysis are presented as well.

Index Terms

Background modeling; visual surveillance; spatio-temporal representation.

I. INTRODUCTION

Background subtraction (also referred as foreground extraction) has been extensively studied in the last decades[1], [2], [3], [4], [5], [6], [7], yet still remains open in surveillance applications due to the following difficulties:

Contact: liang@stat.ucla.edu. This work was supported by Program for New Century Excellent Talents in University (no. NCET-12-0569), Program of Guangzhou Zhujiang Star of Science and Technology (no. 2013J2200067), the Guangdong Science and Technology Program (no. 2012B031500006), and the Guangdong Natural Science Foundation (no. S2013050014548).
Dynamic backgrounds. A scene environment is not always static but sometimes highly dynamic, e.g., rippling water, heavy rain and camera jitter.

Lighting and illumination variations, particularly with sudden changes.

Indistinct foreground objects having similar appearances with surrounding backgrounds.

In this paper, we address the above mentioned difficulties by building the background models with the online pursuit of spatio-temporal models. Some results generated by our system for the challenging scenarios are exhibited in Fig. 1. Prior to unfolding the proposed approach, we first review the existing works in literature.

A. Related Work

Due to their pervasiveness in various applications, the existing works of background subtraction have been afforded by several researchers, and there is no unique categorization of them. Here we introduce the related methods basically according to their representations, to distinguish with our approach.

The pixel-processing approaches modeled scenes as a set of independent pixel processes, which were widely applied in video surveillance applications. In these methods [8], [9], [10], [11], [1], each pixel in the scene can be described by different parametric distributions (e.g. Gaussian Mixture Models) to temporally adapt to the environment changes. The parametric models, however, were not always compatible with
real complex data, as they were defined based upon some underlying assumptions. To overcome this problem, some other non-parametric estimations [12], [13], [14], [15] were proposed, and effectively improved the robustness. For example, Barnich et al. [15] presented a sample-based classification model that maintained a fixed number of samples for each pixel and classified a new observation as background when it matches with a predefined number of samples. Liao et al. [16] recently employed the kernel density estimation (KDE) technique to capture pixel-level variations. Some distinct scene variations, \textit{i.e.} illumination changes and shadows, can be explicitly alleviated by introducing the extra estimations [17], [18]. Guyon et al. [19] proposed to utilize the low rank matrix decomposition for background modeling, where the foreground objects constituted the correlated sparse outliers. Despite acknowledged successes, this category of approaches may have problems on complex scenarios, as the pixel-wise representations overlooked the spatial correlations between pixels.

The \textbf{region-based} methods built background models by taking advantages of inter-pixel relations, and demonstrated promising results on handling dynamic scenes. A batch of diverse approaches were proposed to model spatial structures of scenes, such as joint distributions of neighboring pixels [13], [20], block-wise classifiers [21], structured adjacency graphs [22], auto-regression models [23], [24], random fields [25], and multi-layer models [26] etc. And a number of fast learning algorithms were discussed to maintain their models online, accounting for environment variations or any structural changes. For example, Monnet et al. [23] trained and updated the region-based model by generative subspace learning. Cheng et al. [22] employed the generalized 1-SVM algorithm for model learning and foreground prediction. In general, methods in this category separated the spatial and temporal information, and their performances were somewhat limited in some highly dynamic scenarios, \textit{e.g.} heavy rains or illumination sudden changes.

The third category modeled scene backgrounds by exploiting both spatial and temporal information. Mahadevan et al. [27] proposed to separate foreground objects from surroundings by judging the distinguished video patches, which contain different motions and appearances with the majority of the whole scene. Zhao et al. [28] addressed the outdoor night background modeling by performing subspace learning within video patches. Spatio-temporal representations was also extensively discussed in other vision tasks such as action recognition [29] and trajectory parsing [30]. These methods motivated us to build models upon the spatio-temporal representations, \textit{i.e.} video bricks.

In addition, several \textbf{saliency-based} approaches provided alternative ways based on spatio-temporal saliency estimations [31], [32], [27], [33]. The moving objects can be extracted according to their salient appearances and/or motions against the scene backgrounds. For example, Wixson et al. [31] detected the salient objects according their consistent moving directions over time. Kim et al. [34] used a discriminant
center-surround hypothesis to extract foreground objects around their surroundings.

Along with the above mentioned background models, a number of robust image features were utilized to better handle the background noises. Exemplars included the Local Binary Pattern (LBP) features [35], [36], [37] and color texture histograms [38]. The LBP operators described each pixel by the relative graylevels of its neighboring pixels, and their effectiveness has been demonstrated in several vision tasks such as face recognition and object detection [35]. The Center-Symmetric LBP was proposed in [37] to further improve the computational efficiency. Tan and Triggs [36] extended LBP to LTP (Local Ternary Pattern) by thresholding the graylevel differences with a small value, to enhance the robustness on flat image regions.

B. Overview

In this work, we propose to learn and maintain the dynamic models within spatio-temporal video patches (i.e. video bricks), accounting for real challenges in surveillance scenarios. The algorithm can process 15 ∼ 20 frames per second in the resolution 352 × 288 (pixels) on average. We briefly overview the proposed framework of background modeling in the following aspects.

1. Spatio-temporal representations. We represent the observed scene by video bricks, i.e. video volumes spanning over both spatial and temporal domain, in order to jointly model spatial and temporal information. Specifically, at every location of the scene, a sequence of video bricks are extracted as the observations, within which we can learn and update the background models. Moreover, to compactly encode the video bricks against illumination variations, we design a brick-based descriptor, namely Center Symmetric Spatio-Temporal Local Ternary Pattern (CS-STLTP), which is inspired by the 2D scale invariant local pattern operator proposed in [16]. Its effectiveness can be validated in the experiments.

2. Pursuing dynamic subspaces. We treat each sequence of video bricks at a certain location as a consecutive signal, and generate the subspace within these video bricks. The linear dynamic system (i.e. Auto Regressive Moving Average, ARMA model [39] are adopted to characterize the spatio-temporal statistics of the subspace. Specifically, given the observed video bricks, we express them by a data matrix, in which each of the columns contains the feature of a video brick. The basis vectors (i.e. eigenvectors) of the matrix can be then estimated analytically, representing the appearance parameters of the subspace, and the parameters of dynamical variations are further computed based on the fixed appearance parameters. It is worth mentioning that our background model jointly captures the information of appearance and motion as the data (i.e. features of the video bricks) are extracted over both spatial and temporal domains.
3. Maintaining dynamic subspaces online. Given newly appearing video bricks with our model, moving foreground objects are segmented by estimating the residuals within the related subspaces of the scene, while the background models need to be maintained simultaneously to account for scene changes. The raising problem is to update parameters of the subspaces incrementally against disturbance from foreground objects and background noises. The new observation may include noise pixels (i.e. outliers), resulting in degeneration of model updating [28], [23]. Furthermore, one video brick could be partially occluded by foreground objects in our representation, i.e. only some of pixels in the brick are true positives. To overcome this problem, we present a novel approach to compensate observations (i.e. the observed video bricks) by generating data from the current models. Specifically, we replace the pixels labeled as non-background by the generated data to synthesize the new observations. The algorithm for online model updating performs two steps: (i) update appearance parameters using the robust incremental subspace learning technique, and (ii) update dynamical variation parameters by analytically solving the linear reconstruction. The experiments show that the proposed method effectively improves the robustness during the online processing.

The remainder of this paper is arranged as follows. We first present the model representation in Section II, and then discuss the initial learning, foreground segmentation and online updating mechanism in Section III, respectively. The experiments and comparisons are demonstrated in Section IV and finally comes the conclusion in Section V with a summary.

II. Dynamic Spatio-Temporal Model

In this section, we introduce the background of our model, and then discuss the video brick representation and our model definition, respectively.

A. Background

In general, a complex surveillance background may include diverse appearances that sometimes move and change dynamically and randomly over time flying. There is a branch of works on time-varying texture modeling [40], [41], [42] in computer vision. They often treated the scene as a whole, and pursued a global subspace by utilizing the linear dynamic system (LDS). These models work well on some natural scenes mostly including a few homogeneous textures, as the LDS characterizes the subspace with a set of linearly combined components. However, under real surveillance challenges, it could be intractable to pursue the global subspace. In this work, we represent the observed scene by an array of small and independent subspaces, each of which is defined by the linear system, so that our model is able to handle
better challenging scene variations. Our background model can be viewed as a mixed compositional model consisting of the linear subspaces. Specifically, we conduct the background subtraction with our model based on the following observations.

**Assumption 1:** The local scene variants (i.e. appearance and motion changing over time) can be captured by the low-dimensional subspace.

**Assumption 2:** It is feasible to separate foreground moving objects from the scene background by fully exploiting spatio-temporal statistics.

### B. Spatio-temporal Video Brick

Given the surveillance video of one scene, we first decompose it with a batch of small brick-like volumes. We consider the video brick of small size (e.g., $4 \times 4 \times 5$ pixels) includes relative simple content that can be thus generated by few bases (components). And the brick volume integrates both spatial and temporal information so that we can better capture complex appearance and motion variations compared with the traditional image patch representations.

We divide each frame $I_i (i = 1, 2, \ldots, n)$ into a set of image patches with the width $w$ and height $h$. A number $t$ of patches at the same location across the frames are combined together to form a brick. In this way, we extract a sequence of video bricks $V = \{v_1, v_2, \ldots, v_n\}$ at every location for the scene.

Moreover, we design a novel descriptor, CS-STLTP, to encode the video brick instead of using RGB values. For any video brick $v_i$, we first apply the CS-STLTP operator on each pixel, and pool all the feature values into a histogram. For a pixel $x_c$, we construct a few 2D spatio-temporal planes centered at it, and compute the local ternary patterns (LTP) operator [36] on each plane. The CS-STLTP then encodes $x_c$ by combining the LTP operators of all planes.

We index the neighborhood pixels of $x_c$ by $\{0, \ldots, M\}$, the feature vector on each spatio-temporal plane can be then calculated as:

$$ F^j(x_c) = \bigcup_{k=0}^{M/2-1} s_\tau(p_k, p_k + M/2), \quad (1) $$

where pixel $k$ and $k + M/2$ are two symmetric neighbors of pixel $x_c$. $p_k$ and $p_k + M/2$ are the graylevels of the two pixels, respectively. The sign $\bigcup$ indicates stretching elements into a vector. The function $s_\tau$ is defined as follows:
Fig. 2. An example of computing the CS-STLTP feature. For one pixel in the video brick, we construct four spatio-temporal planes. The center-symmetric local ternary patterns for each plane is calculated, which compares the intensities in a center-symmetric direction with a contrasting threshold $\tau$. The CS-STLTP feature is concatenated by the vectors of the four planes.

$$s_{\tau}(p_{k}, p_{k+\frac{M}{2}}) = \begin{cases} 1, & \text{if } p_{k} > (1 + \tau)p_{k+\frac{M}{2}}, \\ -1, & \text{if } p_{k} < (1 - \tau)p_{k+\frac{M}{2}}, \\ 0, & \text{otherwise.} \end{cases}$$

where $\tau$ is a constant threshold for the comparing range.

Suppose that we take $M = 8$ neighborhood pixels for computing the operator in each spatio-temporal plane, and the number of planes is 4. The resulting CS-STLTP vector contains $M/2 \times 4 = 16$ bins. Fig. 2 illustrates an example of computing the CS-STLTP operator, where we apply the operator for one pixel on 4 spatial-temporal planes displayed with different colors (e.g., green, blue, purple and orange).

Then we build the histogram for each video brick based on the CS-STLTP features on all pixels. We quantize the all possible values of the CS-STLTP feature into $m = 48$ levels, and the histogram can be generated by accumulating the CS-STLTP values for all pixels in the video brick. To further improve the descriptiveness, we can generate histograms in each color channel and concatenate them together in practice.

The CS-STLTP descriptor has two advantages. First, it is computationally efficient and compact to describe the video brick. Second, by introducing a tolerative comparing range in the LTP operator computation, it is robust to local spatio-temporal noises within a range.
C. Model Definition

Let \( m \) be the length of the descriptor for each brick, and \( V = \{v_1, v_2, \ldots, v_n\}, v_i \in \mathbb{R}^m \) be a sequence of video bricks at a certain location of the observed background. We can use a set of bases (components) \( C = [C_1, C_2, \ldots, C_d] \) to represent the subspace where \( V \) lies in. Each video brick \( v_i \) in \( V \) can be represented as

\[
v_i = \sum_{j=1}^{d} z_{i,j} C_j + \omega_i, \tag{3}
\]

where \( C_j \) is the \( j \)-th basis (\( j \)-th column of matrix \( C \)) of the subspace, \( z_{i,j} \) the coefficient for \( C_j \), and \( \omega_i \) the appearance residual. We regard \( C \) representing appearance consistency of the sequence of video bricks. In some traditional background models by subspace learning, \( z_{i,j} \) can be solved and kept as a constant, with the underlying assumption that the appearance of background would be stable within the observations. In contrast, we treat \( z_{i,j} \) as the variable term that can be further phrased as the time-varying state, accounting for temporally coherent variations (i.e. the motions). For notation simplicity, we neglect the subscript \( j \), and we denote \( Z = \{z_1, z_2, \ldots, z_n\} \) for all the bricks. Then we have the dynamic model,

\[
z_{i+1} = Az_i + \eta_i, \tag{4}
\]

where \( \eta_i \) is the state residual, and \( A \) is a matrix of \( d \times d \) dimensions to constrain the variations. With this definition, we consider \( A \) representing the temporal coherence among the observations.

Therefore, the problem of pursuing dynamic subspace is posed as solving the appearance consistency \( C \) and the temporal coherence \( A \), within the observations. Since the sequence states \( Z \) are unknown, we shall jointly solve \( C, A, Z \) by minimizing an empirical energy function \( f_n(C, A, Z) \):

\[
\min f_n(C, A, Z) = \frac{1}{2n} \sum_{i=1}^{n} \|v_i - Cz_i\|_2^2 + \|z_i - Az_{i-1}\|_2^2. \tag{5}
\]

Here \( f_n(C, A, Z) \) is not completely convex but we can solve it by fixing either \( Z \) or \( (C, A) \). Nevertheless, its computation cost is expensive for online learning of the entire background. In this work, following the auto-regressive moving average process (ARMA), we simplify the dynamic model in Equation (4) to be a linear system. Specifically, we introduce a robustness term (i.e. matrix) \( B \), which includes a number \( d_\epsilon \) of bases, and we set \( \eta_i = B\epsilon_i \), where \( \epsilon_i \) denotes the noise.

We further summarize the proposed dynamic model, where we add the subscript \( n \) to the main components, indicating they are solved within a number \( n \) of observations, as,
\( v_i = C_n z_i + \omega_i, \)
\( z_{i+1} = A_n z_i + B_n \epsilon_i, \)
\( \omega_i \overset{iid}{\sim} N(0, \Sigma_\omega), \quad \epsilon_i \overset{iid}{\sim} N(0, I_d). \)  

In this model, \( C_n \in \mathbb{R}^{m \times d} \) and \( A_n \in \mathbb{R}^{d \times d} \) represent the appearance consistency and temporal coherence, respectively. \( B_n \in \mathbb{R}^{d \times d} \) is the robustness term constraining the evolution of \( Z \) over time. \( \omega_i \in \mathbb{R}^m \) indicates the residual corresponding to observation \( v_i \), and \( \epsilon_i \in \mathbb{R}^d \) the noise of state variations. During the subspace learning, \( \omega_i \) and \( \epsilon_i \) are assumed to follow the zero-mean Gaussian distributions. Given a new brick mapped into the subspace, \( \omega_i \) and \( \epsilon_i \) can be used to measure how likely the observation is suitable with the subspace, so that we utilize them for foreground object detection during online processing.

The proposed model is time-varying, and the parameters \( C_n, A_n, B_n \) can be updated incrementally along with the processing of new observations, in order to adapt our model with scene changes.

III. LEARNING ALGORITHM

In this section, we discuss the learning for spatio-temporal background models, including initial subspace generation and online maintenance.

A. Initial Model Learning

In the initial stage, the model defined in Equation (6) can be degenerated as a non-dynamic linear system, as the \( n \) observations are extracted and fixed. Given a brick sequence \( V = \{ v_1, v_2, \ldots, v_n \} \), we present an algorithm to identify the model parameters \( C_n, A_n, B_n \), following the sub-optimal solution proposed in [40].

To guarantee the Equation (6) has an unique and canonical solution, we postulate

\[ n \gg d, \quad \text{Rank}(C_n) = d, \quad C_n^T C_n = I_d, \]

where \( I_d \) is the identity matrix of dimension \( d \times d \). The appearance consistency term \( C_n \) can be estimated as,

\[ C_n = \arg \min_{C_n} | W_n - C_n [ z_1 \ z_2 \ \cdots \ z_n ] | \]
where $W_n$ is the data matrix composed of observed video bricks $[v_1, v_2, \cdots, v_n]$. The equation (8) satisfies the full rank approximation property and can be thus solved by the singular value decomposition (SVD).

We have,

$$W_n = U\Sigma Q^\top,$$

$$U^\top U = I, Q^\top Q = I,$$

where $Q$ is the unitary matrix, $U$ includes the eigenvectors, and $\Sigma$ is the diagonal matrix of the singular values. Thus, $C_n$ is treated as the first $d$ components of $U$, and the state matrix $[z_1 \ z_2 \ \cdots \ z_n]$ as the product of $d \times d$ sub-matrix of $\Sigma$ and the first $d$ columns of $Q^\top$.

The temporal coherence term $A_n$ is calculated by solving the following linear problem:

$$A_n = \arg \min_{A_n} | [z_2 \ z_3 \ \cdots \ z_n] - A_n[z_1 \ z_2 \ \cdots \ z_{n-1}] |.$$

(10)

The statistical robustness term $B_n$ is estimated by the reconstruction error $E$

$$E = [z_2 \ z_3 \ \cdots \ z_n] - A_n[z_1 \ z_2 \ \cdots \ z_{n-1}]$$

$$= B_n[\epsilon_1 \ \epsilon_2 \ \cdots \ \epsilon_{n-1}],$$

where $B_n \approx \frac{1}{\sqrt{n-1}} E$. Since the rank of $A_n$ is $d$ and $d \ll n$, the rank of input-to-state noise $d_\epsilon$ is assumed to be much smaller than $d$. That is, the dimension of $E$ can be further reduced by SVD: $E = U_\epsilon \Sigma_\epsilon Q_\epsilon^\top$, and we have

$$B_n= \frac{1}{\sqrt{n-1}} \begin{bmatrix} U_\epsilon^1 & \cdots & U_\epsilon^{d_\epsilon} \end{bmatrix} \begin{bmatrix} \Sigma_\epsilon^1 \\ \vdots \\ \Sigma_\epsilon^{d_\epsilon} \end{bmatrix}.$$  

(12)

The values of $d, d_\epsilon$ essentially imply the complexity of subspace from the aspects of appearance consistence and temporal coherence, respectively. For example, video bricks containing static content can be well described with a function of low dimensions while highly dynamic video bricks (e.g., from an active fountain) require more bases to generate. In real surveillance scenarios, it is not practical to pre-determine the complexity of scene environments. Hence, in the proposed method, we adaptively determine $d, d_\epsilon$ by thresholding eigenvalues in $\Sigma$ and $\Sigma_\epsilon$, respectively.
\[ d^* = \arg \max_d \Sigma_d^d > T_d, \]
\[ d^*_\epsilon = \arg \max_{d_\epsilon} \Sigma_{d_\epsilon}^{d_\epsilon} > T_{d_\epsilon}, \]

where \( \Sigma_d \) indicates the \( d \)-th eigenvalue in \( \Sigma \) and \( \Sigma_{d_\epsilon}^{d_\epsilon} \) the \( d_\epsilon \)-th eigenvalue in \( \Sigma_\epsilon \).

212

\[ (13) \]

B. Online Model Maintenance

Then we discuss the online processing with our model that segments foreground moving objects and keeps the model updated.

(I) Foreground segmentation. Given one newly appearing video brick \( v_{n+1} \), we can determine whether pixels in \( v_{n+1} \) belong to the background or not by thresholding their appearance residual and state residual. We first estimate the state of \( v_{n+1} \) with the existing \( C_n \),

\[ z'_{n+1} = C_n^\top v_{n+1}, \]

(14)

and further the appearance residual of \( v_{n+1} \)

\[ \omega_{n+1} = v_{n+1} - C_n z'_{n+1}. \]

(15)

As the state \( z_n \) and the temporal coherence \( A_n \) have been solved, we can then estimate the state residual \( \epsilon_n \) according to Equation (6),

\[ B_n \epsilon_n = z'_{n+1} - A_n z_n \]

\[ \Rightarrow \epsilon_n = \text{pinv}(B_n) (z'_{n+1} - A_n z_n), \]

(16)

where \( \text{pinv} \) denotes the operator of pseudo-inverse.

With the state residual \( \epsilon_n \) and the appearance residual \( \omega_{n+1} \) for the new video brick \( v_{n+1} \), we conduct the following criteria of foreground segmentation by introducing two thresholds.

1) \( v_{n+1} \) is classified into background, only if all dimensions of \( \epsilon_n \) are less than a threshold \( T_\epsilon \).

2) If \( v_{n+1} \) has been labeled as non-background, perform the pixel-wise segmentation by comparing \( \omega_{n+1} \) with a threshold \( T_\omega \): the pixel is segmented as foreground if its corresponding dimension in \( \omega_{n+1} \) is greater than \( T_\omega \).

(II) Model updating. During the online processing, the key problem for model updating is to deal with foreground disturbance, \( i.e. \) to avoid absorbing pixels from foreground objects or noises.
In this work, we develop an effective approach to update the model with the synthesized data. We first generate a video brick from the current model, namely noise-free brick, \( \hat{v}_{n+1} \), as

\[
\hat{z}_{n+1} = A_n z_n,
\]

\[
\hat{v}_{n+1} = C_n \hat{z}_{n+1}.
\] (17)

Then we extract pixels from \( \hat{v}_{n+1} \) to compensate occluded (i.e. foreground) pixels in the newly appearing brick. Concretely, the pixels labeled as non-background are replaced by the pixels from the noise-free video brick at the same place. We can thus obtain a synthesized video brick \( \tilde{v}_{n+1} \) for model updating.

Given the brick \( \tilde{v}_{n+1} \), the data matrix \( W_n \) composed of observed video bricks is extended to \( W_{n+1} \).

Then we update the model \( C_{n+1} \) according to Equation (8).

We discuss the algorithm of model updating including two steps: (i) update parameters for appearance consistency \( C_{n+1} \) by employing the incremental subspace learning technique, and (ii) update parameters of state variations \( A_{n+1}, B_{n+1} \).

(i) Step 1. For the \( d \)-dimension subspace, with eigenvectors \( C_n \) and eigenvalues \( \Lambda_n \), its covariance matrix \( \text{Cov}_n \) can be approximated as

\[
\text{Cov}_n \approx \sum_{j=1}^{d} \lambda_{n,j} c_{n,j} c_{n,j}^\top = C_n \Lambda_n C_n^\top,
\] (18)

where \( c_{n,j} \) and \( \lambda_{n,j} \) denote the \( j \)-th eigenvector and eigenvalue, respectively. With the newly synthesized data \( \tilde{v}_{n+1} \), the updated covariance matrix \( \text{Cov}_{n+1} \) is formulated as

\[
\text{Cov}_{n+1} = (1 - \alpha) \text{Cov}_n + \alpha \tilde{v}_{n+1} \tilde{v}_{n+1}^\top
\]

\[
\approx (1 - \alpha) C_n \Lambda_n C_n^\top + \alpha \tilde{v}_{n+1} \tilde{v}_{n+1}^\top
\]

\[
= \sum_{i=1}^{d} (1 - \alpha) \lambda_{n,i} c_{n,i} c_{n,i}^\top + \alpha \tilde{v}_{n+1} \tilde{v}_{n+1}^\top,
\] (19)

where \( \alpha \) denotes the learning rate. It can be re-formulated to simplify computation, as,

\[
\text{Cov}_{n+1} = Y_{n+1} Y_{n+1}^\top,
\] (20)

where \( Y_{n+1} = [y_{n+1,1}, y_{n+1,2}, \ldots, y_{n+1,d+1}] \) and each column \( y_{n+1,j} \) in \( Y_{n+1} \) is defined as

\[
y_i = \begin{cases} 
\sqrt{1 - \alpha \lambda_{n,j}} c_{n,i}, & \text{if } 1 < j < d, \\
\sqrt{\alpha} \tilde{v}_{n+1}, & \text{if } j = d + 1.
\end{cases}
\] (21)
To reduce the computation cost, we can estimate $C_{n+1}$ by a smaller matrix $Y_{n+1}^\top Y_{n+1}$, instead of the original large matrix $\text{Cov}_{n+1}$.

$$(Y_{n+1}^\top Y_{n+1}) e_{n+1,j} = \lambda_{n+1,j} e_{n+1,j} \quad j = 1, 2, \ldots, d + 1,$$

where $e_{n+1,j}$ and $\lambda_{n+1,j}$ are the $j$-th eigenvector and eigenvalue of matrix $Y_{n+1}^\top Y_{n+1}$, respectively. Letting $c_{n+1,j} = Y_{n+1} e_{n+1,j}$, we re-write Equation (22) as

$$Y_{n+1} Y_{n+1}^\top e_{n+1,j} = \lambda_{n+1,j} e_{n+1,j},$$

$$\text{Cov}_{n+1} c_{n+1,j} = \lambda_{n+1,j} c_{n+1,j} \quad j = 1, 2, \ldots, d + 1.$$

We thus obtain the updated eigenvectors $C_{n+1}$ and the corresponding eigenvalues $\Lambda_{n+1}$ of the new covariance matrix $\text{Cov}_{n+1}$. Note that the dimension of the subspace is automatically increased along with the newly added data $\bar{v}_{n+1}$. To guarantee the appearance parameters remain stable, we keep the main principal (i.e. top $d$) eigenvectors and eigenvalues while discarding the least significant components.

The above incremental subspace learning algorithm has been widely applied in several vision tasks such as face recognition and image segmentation [43], [44], [45], and also for background modeling in [28], [46], [6]. However, the noise observations caused by moving objects or scene variations often disturb the subspace maintenance, e.g. the eigenvectors could change dramatically during the processing.

Many efforts have been dedicated to improve the robustness for the incremental learning, such as the robust PCA [47] [48]. On the other hand, several discriminative learning algorithms were investigated to construct the classification models to subtract backgrounds [49]. In this work, we utilize the Robust Incremental PCA (RIPCA) [50] method to cope with the outliers in $\bar{v}_{n+1}$. Note that $\bar{v}_{n+1}$ consists of pixels either from the generated data $\bar{v}_{n+1}$ or real videos, where outliers may exist in some dimensions.

In the traditional PCA learning, the solution is derived by minimizing a least-squared reconstruction error,

$$\min |r_{n+1}|^2 = |C_n C_n^\top \bar{v}_{n+1} - \bar{v}_{n+1}|^2. \quad (24)$$

Following [50], we impose a robust function $w(t) = \frac{1}{1+|t|^\rho}$ over each dimension of $r_{n+1}$, and the target can be re-defined as,

$$\min \sum_j (r_{n+1}^k)^2 \leftarrow w(r_{n+1}^k)(r_{n+1}^k)^2, \quad (25)$$
where the superscript \( k \) indicates the \( k \)-th dimension. The parameter \( \rho \) in the robust function is estimated

\[
\rho = [\rho^1, \rho^2, \ldots, \rho^{\bar{v}_{n+1}}]^T
\]

\[
\rho_k = \max_{i=1}^d \beta \sqrt{\lambda_{n,i} |c_{n,j}^k|}, \quad j = 1, 2, \ldots, |\bar{v}_{n+1}|
\]  

(26)

where \( \beta \) is a fixed coefficient. The \( k \)-th dimension of \( \rho \) is proportional to the maximal projection of the current eigenvectors on the \( k \)-th dimension, \( i.e. \rho^k \) is weighted by their corresponding eigenvalues). Note that \( w(r_{n+1}^k) \) is a function of the residual error which should be calculated for each vector dimension. And the computation cost for \( w(r_{n+1}^k) \) can be neglected in the analytical solution. This robust function effectively ensures that the algorithm is robust to every element of a vector.

Accordingly, we can weight the observation \( \bar{v}_{n+1} \) over each dimension by the robust function, and transfer it into a new \( \mathbf{v}_{n+1}^k \)

\[
\tilde{v}_{n+1}^k = \sqrt{w(r_{n+1}^k)} \bar{v}_{n+1}.
\]  

(27)

That is, we can replace the original \( \bar{v}_{n+1} \) by \( \tilde{v}_{n+1} \) during the incremental learning procedure.

(i) Step 2. With the fixed \( C_{n+1} \), we then update the parameters of state variations \( A_{n+1}, B_{n+1} \). We first estimate the latest state \( z_{n+1} \) based on the updated \( C_{n+1} \) as,

\[
z_{n+1} = C_{n+1}^T \tilde{v}_{n+1}.
\]  

(28)

\( A_{n+1} \) can be further calculated, by re-solving the linear problem of a fixed number of latest observed states,

\[
A_{n+1} \begin{bmatrix} z_{n-l+1} & \cdots & z_n \end{bmatrix} = \begin{bmatrix} z_{n-l+2} & \cdots & z_{n+1} \end{bmatrix},
\]  

(29)

where \( l \) indicates the number of latest observed states, \( i.e. \) the span of observations. And similarly, we update \( B_{n+1} \) by computing the new reconstruction error \( E = [z_{n-l+2} \cdots z_{n+1}] - A_{n+1} [z_{n-l+1} \cdots z_n] \).

We present an empirical study in Fig. 3 to demonstrate the effectiveness of this updating method. The video for background modeling includes dynamic water surfaces. In Fig. 3, we visualize the original and predicted intensities for a fixed position (denoted by the red star), with the blue and red curves, respectively. We can observe that the model remains stable against foreground occlusion.

Time complexity analysis. We mainly employ SVD and linear programming in the initial learning. The time complexity of SVD is \( O(n^3) \) and the learning time of linear programming is \( O(n^2) \). For a certain
Fig. 3. An example to demonstrate the robustness of the model maintenance. In the scenario of dynamic water surfaces, we visualize the original and predicted intensities for a fixed position (denoted by the red star), with the blue and red curves, respectively. With our updating scheme, when the position is occluded by a foreground object during from frame 551 to 632, the predicted intensities are not disturbed by foreground, i.e. the model remains stable.

location, the time complexity of initial learning is \( O(n^3) + O(n^2) = O(n^3) \) for each subspace, where \( n \) denotes the number of video bricks for model learning. As for online learning, incremental subspace learning and linear programming are utilized. Given a \( d \)-dimension subspace, the time complexity for component updating (i.e. step 1 of the model maintenance) is \( O(dn^2) \). Thus, the total time complexity for online learning is \( O(dn^2) + O(l^2) \), where \( l \) is the number of states used to solve the linear problem.

We summarize the algorithm sketch of our framework in Algorithm 1.

IV. EXPERIMENTS

In this section, we first introduce the datasets used in the experiments and the parameter settings, then present the experimental results and comparisons. The discussions of system components are proposed at last.

A. Datasets and settings

We collect a number of challenging videos to validate our approach, which are publicly available or from real surveillance systems. Two of them (AirportHall and TrainStation) from the PETS database\(^1\)

\(^1\)Downloaded from http://www.cvg.rdg.ac.uk/slides/pets.html.
Algorithm 1: The sketch of the proposed algorithm.

**Input**: Video brick sequence \( V = \{v_1, v_2, \ldots, v_n\} \) for every location for the scene.

**Output**: Maintained Background models and foreground regions

forall the locations for the scene do

  Given the observed video bricks \( V \), extract the CS-STLTP descriptor;

  Initialize the subspace by estimating \( C_n, A_n, B_n \) using Equation (7)-(13);

  for the newly appearing video brick \( v_{n+1} \) do

    (1) Extract the CS-STLTP descriptor for \( v_{n+1} \);
    (2) Calculate its state residual \( \epsilon_n \) and appearances residual \( \omega_{n+1} \) by Equation (15) and (16);
    (3) For each pixel of \( v_{n+1} \), classify it into foreground or background by thresholding the two residuals with \( \epsilon_n, \omega_{n+1} \);
    (4) Generate the noise-free brick \( \hat{v}_{n+1} \) from the current model by Equation (17);
    (5) Synthesize video brick \( \bar{v}_{n+1} \) for model updating;
    (6) Compute the robust brick \( \tilde{v}_{n+1} \) by weighting \( \bar{v}_{n+1} \);
    (7) Update the new appearance parameter \( C_{n+1} \) by calculating the covariance matrix \( \text{Cov}_{n+1} \) with the learning rate \( \alpha \);
    (8) Update the state variation parameters \( A_{n+1}, B_{n+1} \);

  end

end

include crowded pedestrians and moving cast shadows; five highly dynamic scenes \(^2\) include waving curtain active fountain, swaying trees, water surface; the others contain extremely difficult cases such as heavy rain, sudden and gradual light changing. Most of the videos include several thousand frames, and some of the frames are manually annotated as the ground-truth provided by the original databases.

Our algorithm has been adopted in a real video surveillance system and achieves a satisfactory performance. The system is capable of processing 15 \( \sim \) 20 frames per second in the resolution 320 \( \times \) 256 pixels. The hardware architecture is an Intel i7 2600 (3.4 GHz) CPU and 8GB RAM desktop computer.

All the parameters are fixed in the experiments, including the contrast threshold of CS-STLTP descriptor \( \tau = 0.2 \), the dimension threshold of ARMA model \( T_d = 0.5 \), \( T_{d_c} = 0.5 \), the span of observations for

\(^2\)Downloaded from http://perception.i2r.a-star.edu.sg
TABLE I
QUANTITATIVE RESULTS AND COMPARISONS ON THE 10 COMPLEX VIDEOS USING THE F-SCORE (%) MEASUREMENT. THE LAST TWO COLUMNS REPORT THE RESULTS OF OUR METHOD USING EITHER RGB OR CS-STLTP AS FEATURE.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1# Airport</td>
<td>46.99</td>
<td>47.36</td>
<td>62.72</td>
<td>60.23</td>
<td>65.35</td>
<td>68.14</td>
<td>70.52</td>
<td>66.40</td>
</tr>
<tr>
<td>2# Floating Bottle</td>
<td>57.91</td>
<td>57.77</td>
<td>43.79</td>
<td>45.64</td>
<td>47.87</td>
<td>59.57</td>
<td>69.04</td>
<td>78.17</td>
</tr>
<tr>
<td>3# Waving Curtain</td>
<td>62.75</td>
<td>74.58</td>
<td>77.86</td>
<td>72.72</td>
<td>77.34</td>
<td>78.01</td>
<td>79.74</td>
<td>74.93</td>
</tr>
<tr>
<td>4# Active Fountain</td>
<td>52.77</td>
<td>60.11</td>
<td>70.41</td>
<td>68.53</td>
<td>74.94</td>
<td>76.33</td>
<td>76.85</td>
<td>85.46</td>
</tr>
<tr>
<td>5# Heavy Rain</td>
<td>71.11</td>
<td>81.54</td>
<td>78.68</td>
<td>75.88</td>
<td>82.62</td>
<td>76.71</td>
<td>79.35</td>
<td>75.29</td>
</tr>
<tr>
<td>6# Sudden Light</td>
<td>47.11</td>
<td>51.37</td>
<td>37.30</td>
<td>52.26</td>
<td>47.61</td>
<td>52.63</td>
<td>51.56</td>
<td>74.57</td>
</tr>
<tr>
<td>7# Gradual Light</td>
<td>51.10</td>
<td>50.12</td>
<td>13.16</td>
<td>47.48</td>
<td>62.44</td>
<td>54.86</td>
<td>54.84</td>
<td>77.41</td>
</tr>
<tr>
<td>8# Train Station</td>
<td>65.12</td>
<td>68.80</td>
<td>36.01</td>
<td>57.68</td>
<td>61.79</td>
<td>67.05</td>
<td>73.43</td>
<td>66.35</td>
</tr>
<tr>
<td>9# Swaying Trees</td>
<td>19.51</td>
<td>23.25</td>
<td>63.54</td>
<td>45.61</td>
<td>42.38</td>
<td>42.54</td>
<td>43.71</td>
<td>75.89</td>
</tr>
<tr>
<td>10# Water Surface</td>
<td>79.54</td>
<td>86.01</td>
<td>77.31</td>
<td>84.27</td>
<td>83.13</td>
<td>74.30</td>
<td>88.54</td>
<td>88.68</td>
</tr>
<tr>
<td>Average</td>
<td>55.39</td>
<td>59.56</td>
<td>57.02</td>
<td>60.23</td>
<td>59.79</td>
<td>63.08</td>
<td>68.75</td>
<td>76.31</td>
</tr>
</tbody>
</table>

model updating \( l = 60 \), and the size of bricks \( 4 \times 4 \times 5 \). For foreground segmentation, the threshold of appearance residual \( T_{\omega} = 3 \), update threshold \( T_{\epsilon} = 3 \) and \( T_{\omega} = 5 \), \( T_{\epsilon} = 4 \) for RGB. In online model maintenance, the robust coefficient \( \beta = 2.3849 \), the learning rate \( \alpha = 0.05 \) for RIPCA.

In our experiment, we find that foreground region detected with CS-STLTP is sensitive to the contours of foreground objects and yet insensitive to the flat regions inside foreground objects. Therefore, we utilize a standard post-processing to fill contours whose area is greater than 20 pixels and eliminate area less than 20 pixels. Note that we apply the post-processing for all other approaches in the comparisons.

We utilize the F-score as the benchmark metric, which measures the segmentation accuracy by considering both the recall and the precision. The F-score is defined as

\[
F = \frac{2 TP}{2 TP + FP + FN},
\]

where TP is true positives (foreground objects), FN false negatives (false background pixels), FP false positive (false foreground pixels).

B. Experimental results

Experimental results. We compare the proposed method (STDM) with six state-of-the-art online background subtraction algorithms including Gaussian Mixture Model (GMM) [8] as baseline, improved
Fig. 4. Experimental results generated by our approach and competing methods on 5 videos: first row left, the scene including a dynamic curtain and indistinguishable foreground objects (i.e., having similar appearance with backgrounds); first row right, the scene with heavy rain; second row left, an indoor scene with the sudden lighting changes; second row right, the scene with dynamic water surface; third row, a busy airport. The precision-recall (PR) curve is introduced as the benchmark measurement for all the 6 algorithms.
Fig. 5. Sampled results of background subtraction generated by our approach (using RGB or CS-STLTP as the feature and RIPCA as the update strategy) and other competing methods.
GMM [9], online auto-regression model [23], non-parametric model with scale-invariant local patterns [16], discriminative model using generalized Struct 1-SVM [22], and the Bayesian joint domain-range (JDR) model [13]. We adopt the provided codes of the methods [8], [9], [22], [13] and implement the methods [16], [23] according to their descriptions. The F-scores (%) over all 10 videos are reported in Table I, where the last two columns report results of our method using either RGB or CS-STLTP as feature. We also exhibit the results and comparisons using the precision-recall (PR) curves, as shown in Fig. 4. Due to space limitation, we only show results on 5 videos. From the results, we can observe that the proposed method outperforms the other methods in most videos in general. For the scenes with highly dynamic backgrounds (e.g., the #2, #5, and #10 scenes), the improvements made by our method are more than 10%. And the system enables us to well handle the indistinctive foreground objects (i.e. small objects or background-like objects in the #1, #3 scenes). Moreover, we make significant improvements (i.e. 15% ~ 25%) in the scene #6 and #7 including both sudden and gradual lighting changes. A number of sampled results of background subtraction are exhibited in Fig. 5.

The benefit of using the proposed CS-STLTP feature is clearly validated by observing the results shown in Table I and Fig. 5. In general, our approach simply using RGB values can achieve satisfying performances for the common scenes, e.g., with fair appearance and motion changes, while the CS-SILTP operator can better handle highly dynamic variations (e.g. sudden illumination changing, rippling water). In addition, we also compare CS-STLTP with the existing scale invariant descriptor SILTP proposed in [16]. We reserve all settings in our approach except replacing the feature by SILTP, and achieve the average precision over all 10 videos: 69.70%. This result shows that CS-STLTP is very suitable and robust for the video brick representation.

C. Discussion

Furthermore, we conduct the following empirical studies to justify the parameter determinations and settings of our approach.

Adaption efficiency. Like other online-learning background models, there is a trade-off between the model stability and adaption efficiency. The corresponding parameter in our method is the learning rate α. We tune α in the range of 0 ~ 0.3 by fixing the other model parameters and visualize the quantitative

\[^3\text{Available at http://dparks.wikidot.com/background-subtraction}\]

\[^4\text{Available at http://www.cs.mun.ca/~gong/Pages/Research.html}\]

\[^5\text{Available at http://www.cs.cmu.edu/~yaser/}\]
Fig. 6. Discussion of parameter selection: (i) learning rate $\alpha$ for model adaption (in (a)) and (ii) the contrast threshold of CS-STLTP feature $\tau$ (in (b)). In each figure, the horizontal axis represents the different parameter values; the three lines in different colors denote, respectively, the false positive (FP), false negative (FN), and the sum of FP and FN.

Fig. 7. Empirical study for the size of video brick in our approach. We carry on the experiments on the 10 videos with different brick size while keeping the rest settings. The vertical axis represents the average precisions of background subtraction and the horizontal represents the different sizes of video bricks with respect to background decomposition.

results of background subtraction, as shown in Fig. 6(a). From the results, we can observe this parameter is insensitive in range $0 \sim 0.1$ in our model. In practice, once the scene is extremely busy and crowded, it could be set as a relative small value to keep model stable.

Feature effectiveness. The contrast threshold $\tau$ is the only parameter in CS-STLTP operator, which affects the characterizing power for dynamic textures (i.e. spatio-temporal information within video
bricks). From the empirical results of parameter tuning, as shown in Fig. 6 (b), we can observe that the appropriate range for $\tau$ is $0.15 \sim 0.25$. In practice, a very small value (i.e. $< 0.15$) for $\tau$ could lead the model sensitive to noises while some foreground regions of homogeneous appearances could be missed under a great value setting (i.e. $> 0.25$).

Size of video brick. One may be interested in how the performances are affected by the size of video brick for background decomposition, so that we present an empirical study on different sizes of video bricks in Fig. 7. We observe that the best performances are achieved with the certain brick size of $4 \times 4 \times 3$, and the performances with the sizes of $4 \times 4 \times 1$ and $4 \times 4 \times 5$ are also satisfied. As of very small bricks (e.g. $1 \times 1 \times 3$), few spatio-temporal statistics are captured and the models may have problems on handling scene variations. The bricks of large sizes (e.g. $8 \times 8 \times 5$ ) carry too much information, and their subspaces cannot be effectively generated by the linear ARMA model. The experimental results are also accordant with our motivations in Section I. In practice, we can flexibly set the size according to the resolutions of surveillance videos.

Incremental model updating. We also present the empirical study on alternative algorithms for online model updating. We adopt another incremental subspace learning algorithm, namely candid covariance-free incremental PCA (CCIPCA) algorithm [51], to update the parameters of the subspace. CCIPCA was proposed to incrementally estimate the principal components without computing the covariance matrix. We reserve the settings of the rest parameters for the comparison. Fig. 8 reports the results on 10 videos, where the RIPCA algorithm achieves the superior performance substantially. The results show that the
imposed robust function effectively handles the noises caused by outliers. For example, RIPCA greatly improve the results on the “Swaying trees” scene (i.e. #9 in Fig. 8), as the scene includes heavy variations in appearance and motion.

V. CONCLUSION

This paper studies an effective method for background subtraction, addressing the all challenges in real surveillance scenarios. In the method, we learn and maintain the dynamic texture models within spatio-temporal video patches (i.e. video bricks). Sufficient experiments as well as empirical analysis are presented to validate the advantages of our method.

In the future, we plan to improve the method in two aspects. (1) Some efficient tracking algorithms can be employed into the framework to better distinguish the foreground objects. (2) The GPU-based implementation can be developed to process each part of the scene in parallel, and it would probably significantly improve the system efficiency.

REFERENCES


