Abnormal Event Detection at 150 FPS in MATLAB

Cewu Lu Jianping Shi Jiaya Jia
The Chinese University of Hong Kong
{cwlu, jpshi, leojia}@cse.cuhk.edu.hk

Abstract

Speedy abnormal event detection meets the growing demand to process an enormous number of surveillance videos. Based on inherent redundancy of video structures, we propose an efficient sparse combination learning framework. It achieves decent performance in the detection phase without compromising result quality. The short running time is guaranteed because the new method effectively turns the original complicated problem to one in which only a few costless small-scale least square optimization steps are involved. Our method reaches high detection rates on benchmark datasets at a speed of 140∼150 frames per second on average when computing on an ordinary desktop PC using MATLAB.

1. Introduction

With the increasing demand of security, surveillance cameras are commonly deployed. Detecting abnormal events is one critical task based on what cameras capture, which is traditionally labor-intensive and requires non-stop human attention. What makes this interminable and boring process worse is that abnormal events generally happen with a frustratingly small chance, making over 99.9% of the effort for one to watch videos wasted.

This predicament catalyzes important research in computer vision, aiming to find abnormal events automatically [8, 1, 13, 23, 16, 21, 11, 22, 5]. It is not a typical classification problem due to the difficulty to list all possible negative samples. Research in this area commonly follows the line that normal patterns are first learned from training videos, and are then used to detect events deviated from this representation.

Specifically, in [8, 20], extracted trajectories were utilized by tracking object-of-interest to represent normal patterns. Outliers are regarded as abnormal. Another line is to learn normal low-level video feature distributions, such as exponential [1], multivariate Gaussian mixture [13] or clustering as a special case [23, 16]. Graph model normal event representations were proposed in [21, 11, 10, 3, 19, 15, 6, 2], which utilize co-occurrence patterns. Among these methods, normal patterns were fitted in a space-time Markov random field in [21, 11, 10, 3]. Kwon and Lee [9] used a graph editing framework for abnormal event detection. Topic models, such as latent Dirichlet allocation, were employed [19, 15]. Recently, sparse representation [12, 17] has attracted much attention and sparsity-based abnormality detection models [22, 5] achieved state-of-the-art performance reported in many datasets.

Although realtime processing is a key criterion to an practically employable system given continuously captured videos, most sparsity-based methods cannot be performed fast enough. The major obstruction to high efficiency is the inherently intensive computation to build the sparse representation. Note a slow process could delay alarm and postpone response to special events.

We provide brief analysis below about this issue with respect to the general sparsity strategies and present our new framework with an effective representation. It fits the structure of surveillance video data and leads to an extremely cheap testing cost.

1.1. Sparsity Based Abnormality Detection

Sparsity is a general constraint [22, 5] to model normal event patterns as a linear combination of a set of basis atoms. We analyze abnormality detection in one local region to show that this process is computationally expensive by nature.

Given training features $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ extracted from the history video sequence in a region, a normal pattern dictionary $\mathbf{D} \in \mathbb{R}^{p\times q}$ is learned with a sparsity prior. In the testing phase for a new feature $\mathbf{x}$, we reconstruct it by sparsely combining elements in $\mathbf{D}$, expressed as

$$\min_{\boldsymbol{\beta}} \|\mathbf{x} - \mathbf{D}\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq s \tag{1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{q\times 1}$ contains sparse coefficients. $\|\mathbf{x} - \mathbf{D}\boldsymbol{\beta}\|_2^2$ is the data fitting term; $\|\boldsymbol{\beta}\|_0$ is the sparsity regularization term; and s ($\ll q$) is a parameter to control sparsity. With this representation, an abnormal pattern can be naturally defined as one with large error resulted from $\|\mathbf{x} - \mathbf{D}\boldsymbol{\beta}\|_2^2$.

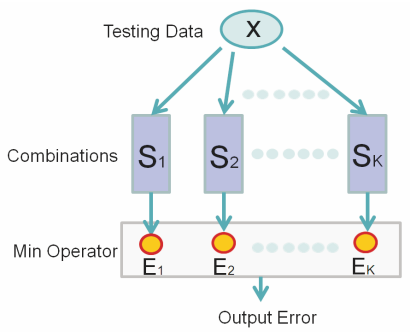


Figure 1. Our testing architecture. $\mathbf{X}$ denotes testing data. $\{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ are learned combinations, with each $\mathbf{S}_i \in \mathbb{R}^{p \times s} (s \ll q)$. E_i is the corresponding least square reconstruction error. The final error is the minimum among all combinations.

Previous work verified that this form can lead to high detection accuracy.

Efficiency Problem A high testing cost is inevitable when adopting Eq. (1), which aims to find the suitable basis vectors (with scale s) from the dictionary (with scale q) to represent testing data $\mathbf{x}$. The search space is very large, as $\binom{q}{s}$ different combinations exist. Although much effort has been put to reducing the dictionary size [5] and adopting fast sparse coding solvers [22], in general, seconds are needed to process a frame as reported in prior papers.

The efficiency problem is thus critical to address before this type of methods can be deployed practically. A realtime process needs to be 100 times faster than the current fastest sparsity-based methods, which is difficult without tremendous hardware advancement. We tackle this problem from an algorithm perspective. Our method yields decent performance and naturally accelerates sparse coding by 400+ times even using MATLAB implementation.

1.2. Our Contribution

We propose *sparse combination learning* for detection. With high structure redundancy in surveillance videos, instead of coding sparsity by finding an s basis combination from $\mathbf{D}$ in Eq. (1), we code it directly as a set of possible combinations of basis vectors. Each combination here corresponds to a set with s dictionary bases in Eq. (1). With this change, other than searching s bases from p of them for each testing feature, we only need to find the most suitable combination by evaluating the least square error. The testing framework is shown in Fig. 1.

This framework is efficient since only small-scale least square optimization is required in detection with simple matrix projection. In our experiments, testing is on a small number of combinations, each takes $10^{-6} \sim 10^{-7}$ second in MATLAB.

The effectiveness of our approach is well guaranteed by the inherent sparsity constraint on the combination size.

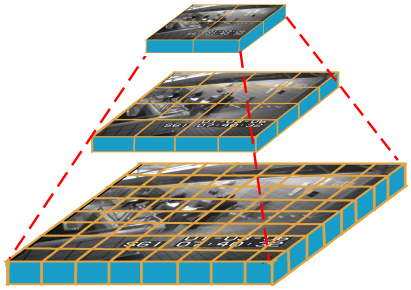


Figure 2. Pyramid region architecture. A frame is resized into 3 different scales. In each scale the frame is partitioned into several regions.

Compared to original sparse coding, our model is more faithful to the input data. When freely selecting s basis vectors from a total of q vectors by Eq. (1), the reconstructed structure could much deviate from input due to the large freedom. But in our trained combinations, it is unlikely to happen, since each combination finds its corresponding input data, better constraining reconstruction quality. Our method therefore is robust to distinguish between normal and abnormal patterns.

We have verified our model on a large set of surveillance videos in Sec. 3.2. We also benchmark it on existing datasets for abnormal event detection. It reaches 140∼150 FPS using a desktop with 3.4GHz CPU and 8G memory in MATLAB 2012.

2. Method

We describe our framework that learns sparse basis combinations. To extract usable data, we resize each frame into different scales as [5] and uniformly partition each layer to a set of non-overlapping patches. All patches have the same size. Corresponding regions in 5 continuous frames are stacked together to form a spatial-temporal cube. An example is illustrate in Fig. 2. This pyramid involves local information in fine-scale layers and more global structures in small-resolution ones.

With the spatial-temporal cubes, we compute 3D gradient features on each of them following [11]. These features in a video sequence are processed separately according to their spatial coordinates. Only features at the same spatial location in the video frames are used together for training and testing.

2.1. Learning Combinations on Training Data

For each cube location, 3D gradient features in all frames are denoted as $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{p \times n}$, gathered temporally for training. Our goal is to find a sparse basis combination set $\mathcal{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ with each $\mathbf{S}_i \in \mathbb{R}^{p \times s}$ containing s dictionary basis vectors, forming a unique combination, where $s \ll q$. Each $\mathbf{S}_i$ belongs to a closed, convex and bounded set, which ensures column-wise unit

norm to prevent over-fitting.

Our sparse combination learning has two goals. The first goal – effective representation – is to find K basis combinations, which enjoy a small reconstruction error t. It is coarsely expressed as

$$\begin{aligned} t \triangleq \min_{\mathcal{S},\boldsymbol{\gamma},\boldsymbol{\beta}} & \sum_{j=1}^{n}\sum_{i=1}^{K}\gamma_j^i\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2 \\ \text{s.t.} & \sum_{i=1}^{K}\gamma_j^i=1,\gamma_j^i=\{0,1\} \end{aligned} \tag{2}$$

where $\boldsymbol{\gamma}=\{\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_n\}$ and $\boldsymbol{\gamma}_j=\{\gamma_j^1,\ldots,\gamma_j^K\}$. Each γ_j^i indicates whether or not the i^{th} combination $\mathbf{S}_i$ is chosen for data $\mathbf{x}_j$. $\boldsymbol{\beta}_j^i$ is the corresponding coefficient set for representing $\mathbf{x}_j$ with combination $\mathbf{S}_i$. The constraints $\sum\gamma_j^i=1$ and $\gamma_j^i=\{0,1\}$ require that only one combination is selected. The objective function makes each training cube $\mathbf{x}$ constructible by at least one basis combination in $\mathcal{S}$.

The second goal is to make the total number K of combinations small enough based on redundant surveillance video information. It is natural and inevitable because a very large K could possibly make the reconstruction error t in Eq. (2) always close to zero, even for abnormal events.

2.2. Optimization for Training

Our two objectives contradict each other in a sense. Reducing K could increase reconstruction errors. It is not optimal to fix K as well, as content may vary among videos. This problem is addressed in our system with a *maximum representation strategy*. It automatically finds K while not wildly increasing the reconstruction error t. In fact, error t for each training feature is upper bounded in our method.

We obtain a set of combinations with a small K by setting a reconstruction error upper bound λ uniformly for all elements in $\mathcal{S}$. If the reconstruction error for each feature is smaller than λ, the coding result is with good quality. So we update function (2) to

$$\begin{aligned} \forall j\in\{1,\cdots,n\},\ t_j=\sum_{i=1}^{K}\gamma_j^i\{\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2-\lambda\}\le 0, \\ \text{s.t.}\ \sum_{i=1}^{K}\gamma_j^i=1,\gamma_j^i=\{0,1\} \end{aligned} \tag{3}$$

Algorithm Our method performs in an iterative manner. In each pass, we update only one combination, making it represent as many training data as possible. This process can quickly find the dominating combinations encoding important and most common features. Remaining training cube features that cannot be well represented by this combination are sent to the next round to gather residual maximum commonness. This process ends until all training data are computed and bounded. The size of combination K reflects how informative the training data are.

Specifically, in the i^{th} pass, given the leftover training data $\mathcal{X}_c\subseteq\mathcal{X}$ that cannot be represented by previous combinations $\{\mathbf{S}_1,\ldots,\mathbf{S}_{i-1}\}$, we compute $\mathbf{S}_i$ to bound most data in $\mathcal{X}_c$. Our objective function becomes

$$\begin{aligned} \min_{\mathbf{S}_i,\boldsymbol{\gamma},\boldsymbol{\beta}} & \sum_{j\in\Omega_c}\gamma_j^i(\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2-\lambda) \\ \text{s.t.} & \sum_{i=1}^{K}\gamma_j^i=1,\gamma_j^i=\{0,1\} \end{aligned} \tag{4}$$

where Ω_c is the index set for $\mathcal{X}_c$. It is easy to prove that this cost satisfies condition (3) and the resulting $\mathbf{S}_i$ can represent most data. Specifically, if $\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2-\lambda\ge 0$, setting $\gamma_j^i=0$ yields a smaller value compared to setting $\gamma_j^i=1$. Contrarily, γ_j^i should be 1 if $\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2-\lambda<0$, complying with condition (3).

In each pass i, we solve the function in Eq. (4) by dividing it into two steps to iteratively update $\{\mathbf{S}_s^i,\boldsymbol{\beta}\}$ and $\boldsymbol{\gamma}$ using the following procedure.

Update $\{\mathbf{S}_s^i,\boldsymbol{\beta}\}$ With fixed $\boldsymbol{\gamma}$, Eq. (4) becomes a quadratic function

$$L(\boldsymbol{\beta},\mathbf{S}_i)=\sum_{j\in\Omega_c}\gamma_j^i\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2. \tag{5}$$

Following the traditional procedure, we optimize $\boldsymbol{\beta}$ while fixing $\mathbf{S}_i$ for all $\gamma_j^i\neq 0$ and then optimize $\mathbf{S}_i$ using block-coordinate descent [14]. These two steps alternate. The closed-form solution for $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}_j^i=(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T\mathbf{x}_j. \tag{6}$$

$\mathbf{S}_s^i$ finds its solution as

$$\mathbf{S}_i=\prod[\mathbf{S}_i-\delta_t\nabla_{\mathbf{S}_i}L(\boldsymbol{\beta},\mathbf{S}_i)], \tag{7}$$

where δ_t is set to $1E-4$ and $\prod$ denotes projecting the basis to a unit column. Block-coordinate descent can converge to a global optimum due to its convexity [4]. Therefore, the total energy for $L(\boldsymbol{\beta},\mathbf{S}_i)$ decreases in each iteration, guaranteeing convergence.

Update $\boldsymbol{\gamma}$ With the $\{\mathbf{S}_i,\boldsymbol{\beta}\}$ output, for each $\mathbf{x}_j$, the objective function becomes

$$\begin{aligned} \min_{\gamma_j^i} & \ \gamma_j^i\|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2-\lambda\gamma_j^i \\ \text{s.t.} & \ \gamma_j^i=0 \text{ or } 1\ . \end{aligned} \tag{8}$$

γ_j^i has a closed-form solution

$$\gamma_j^i=\begin{cases} 1 & if\ \ \|\mathbf{x}_j-\mathbf{S}_i\boldsymbol{\beta}_j^i\|_2^2<\lambda \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The update step guarantees condition (3).

Algorithm 1 Training for Sparse Combination Learning

Input: $\mathcal{X}$, current training features $\mathcal{X}_c = \mathcal{X}$
initialize $\mathcal{S} = \emptyset$ and $i = 1$
repeat
  repeat
    Optimize $\{\mathbf{S}_i, \boldsymbol{\beta}\}$ with Eqs. (6) and (7)
    Optimize $\{\boldsymbol{\gamma}\}$ using Eq. (9)
  until Eq. (5) converges
  Add $\mathbf{S}_i$ to set $\mathcal{S}$
  Remove computed features $\mathbf{x}_j$ with $\gamma_j^i = 0$ from $\mathcal{X}_c$
  $i = i + 1$
until $\mathcal{X}_c = \emptyset$
Output: $\mathcal{S}$

Algorithm Summary and Analysis In each pass, we learn one $\mathbf{S}_i$. We repeat this process to obtain a few combinations until the training data set $\mathcal{X}_c$ is empty. This scheme reduces information overlap between combinations. We summarize our training algorithm in Algorithm 1. The initial dictionary $\mathbf{S}_i$ in each pass is calculated by clustering training data $\mathcal{X}_c$ via K-means with s centers.

Our algorithm is controlled by λ, the upper bound of reconstruction errors. Reducing it could lead to a larger K. Our approach is expressive because all training normal event patterns are represented with controllable reconstruction errors under condition (3). We train $20K$-sample data, whose scales are in $\mathbb{R}^{100}$, in 20 minutes on a PC with 8GB RAM and an Intel 3.4GHz CPU.

2.3. Testing

With the learned sparse combinations $\mathcal{S} = \{\mathbf{S}_1 \dots \mathbf{S}_K\}$, in the testing phase with new data $\mathbf{x}$, we check if there exists a combination in $\mathcal{S}$ fitting the reconstruction error upper bound. It can be quickly achieved by checking the least square error for each $\mathbf{S}_i$:

$$\min_{\boldsymbol{\beta}^i} \|\mathbf{x} - \mathbf{S}_i\boldsymbol{\beta}^i\|_2^2 \quad \forall\, i = 1, \dots, K \tag{10}$$

It is a standard quadratic function with the optimal solution

$$\widehat{\boldsymbol{\beta}^i} = (\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T\mathbf{x}. \tag{11}$$

The reconstruction error in $\mathbf{S}_i$ is

$$\|\mathbf{x} - \mathbf{S}_i\widehat{\boldsymbol{\beta}^i}\|_2^2 = \|(\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T - \mathbf{I}_p)\mathbf{x}\|_2^2, \tag{12}$$

where $\mathbf{I}_p$ is a $p \times p$ identity matrix. To further simplify computation, we define an auxiliary matrix $\mathbf{R}_i$ for each $\mathbf{S}_i$:

$$\mathbf{R}_i = \mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T - \mathbf{I}_p. \tag{13}$$

Reconstruction error for $\mathbf{S}_i$ is accordingly $\|\mathbf{R}_i\mathbf{x}\|_2^2$. If it is small, $\mathbf{x}$ is regarded as a normal event pattern. The final testing scheme is summarized in Algorithm 2.

Algorithm 2 Testing with Sparse Combinations

Input: $\mathbf{x}$, auxiliary matrices $\{\mathbf{R}_1, \dots, \mathbf{R}_K\}$ and threshold T
for $j = 1 \to K$ **do**
  if $\|\mathbf{R}_k\mathbf{x}\|_2^2 < T$ **then**
    return normal event;
  end if
end for
return abnormal event;

It is noted that the first a few dominating combinations represent the largest number of normal event features, which enable us to determine positive data quickly. In our experiments, the average combination checking ratio is 0.325, which is the number of combinations checked divided by the total number K. Also, our method can be easily accelerated via parallel processing to achieve $O(1)$ complexity although it is generally not necessary.

2.4. Relation to Subspace Clustering

Our approach can be regarded as enhancement of subspace clustering [7] with the major difference on the working scheme. The relationship between subspace clustering and our method is similar to that between K-means and hierarchical clustering [18]. Specifically, subspace clustering method [7] takes the number of clusters k as known or fixed beforehand, like K-means. In video abnormality detection applications, it is however difficult to know the optimal number of bases in prior. Our approach utilizes the allowed representation error to build combinations, where the error upper bound is explicitly implemented with clear statistical meaning. There is no need to define the cluster size in this method. Our extensive experiments manifest that our strategy is both reliable and efficient.

3. Experiments

We empirically demonstrate that our model is suitable to represent general surveillance videos. We apply our method to different datasets. Quantitative comparisons are provided.

3.1. System Setting

In our method, size of $\mathbf{S}_i \in \mathbb{R}^{p\times s}$ controls the sparsity level. We experimentally set $s = 0.1 \times p$ where p is the data dimension. λ in Eq. (4) is the error upper bound, set to 0.04 in experiments.

Given the input video, we resize each frame to 3 scales with 20×20, 30×40, and 120×160 pixels respectively and uniformly partition each layer to a set of non-overlapping 10×10 patches, leading to 208 sub-regions for each frame in total shown in Fig. 2. Corresponding sub-regions in 5

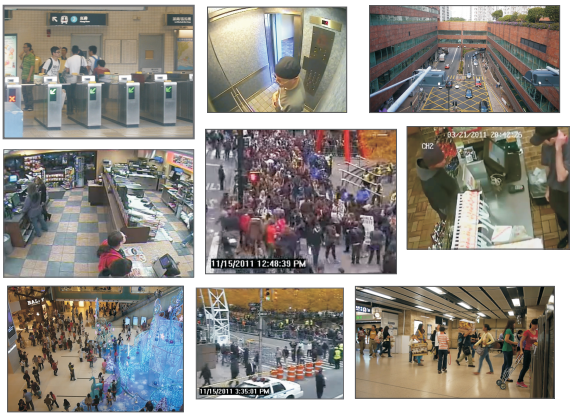


Figure 3. A few frames from the surveillance videos used for verification.

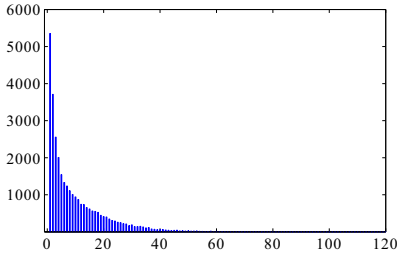


Figure 4. Different numbers of basis combinations to represent normal events in 31,200 groups (x-axis: K; y-axis: number of groups that use K combinations).

continuous frames are stacked together to form a spatial-temporal cube, each with resolution $10 \times 10 \times 5$. We compute 3D gradient features on each of them following [11]. Those gradients are concatenated to a 1500-dimension feature vector for each cube and are then reduced to 100 dimensions via PCA. Normalization is performed to make them mean 0 and variance 1.

For each frame, we compute an abnormal indicator V by summing the number of cubes in each scale with weights. It is defined as $V = \sum_{i=1}^{n} 2^{n-i} v_i$, where v_i is the number of abnormal cubes in scale i. The top scale is with index 1 while the bottom one is with n. All experiments are conducted using MATLAB.

3.2. Verification of Sparse Combinations

Surveillance videos consist of many redundant patterns. For example, in subway exit, people generally move in similar directions. These patterns share information coded in our sparse combinations. To verify it, we collect 150 normal event surveillance videos with a total length of 107.8 hours. The videos are obtained from sources including dataset UCSD Ped1 [15], Subway datasets [1] (excluding abnormal event frames), 68 videos from YouTube, and 79 videos we captured. The scene includes subway, mall, traffic, indoor, elevator, square, etc. We show a few example

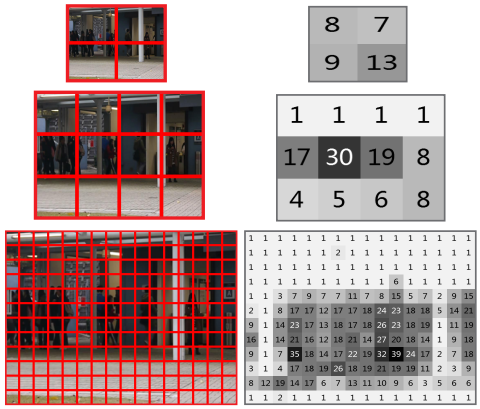


Figure 5. Spatial distribution of combination numbers to represent normal structures in the Avenue data.

	Run	Loiter	Throw	False Alarm
Ground Truth	4	5	5	N/A
Ours	4	4	4	1

Table 1. Detection results in the Avenue dataset.

frames in Fig. 3.

Each video contains 208 regions as illustrated in Fig. 2. With the 150 different videos, we gather a total of 31,200 (208×150) groups of cube features with each group corresponding to a set of patches (cubes). They are used separately to verify the combination model. Each group contains 6,000-120,000 features. The number of combinations for each group is denoted as K. We show in Fig. 4 the distribution of K in the 31,200 groups. Its mean is 9.75 and variance is 10.62, indicating 10 combinations are generally enough in our model. The largest K is 108. About 99% of the Ks are smaller than 45.

We illustrate the K distributions spatially in the Avenue data (described below) in Fig. 5. Many regions only need 1 combination because they are static. Largely varying patches may need dozens of combinations to summarize normal events. The statistical regression error is as small as $0.0132 \pm 1.38E-4$, which indicates our dictionaries contain almost all normal patterns.

3.3. Avenue Data Benchmark

We build an avenue dataset, which contains 15 sequences. Each sequence is about 2 minutes long. The total number of frames is 35,240. There are 14 unusual events including running, throwing objects, and loitering. 4 videos are used as training data with 8,478 frames in total.

A video sequence and its abnormal event detection result are demonstrated in Fig. 6. Fig. 7 contains two important frames and their abnormal event regions in two image scales. We list the detection statistics in Table 1. The performance of our method is satisfactory with the average detection rate of 141.34 frames per second.

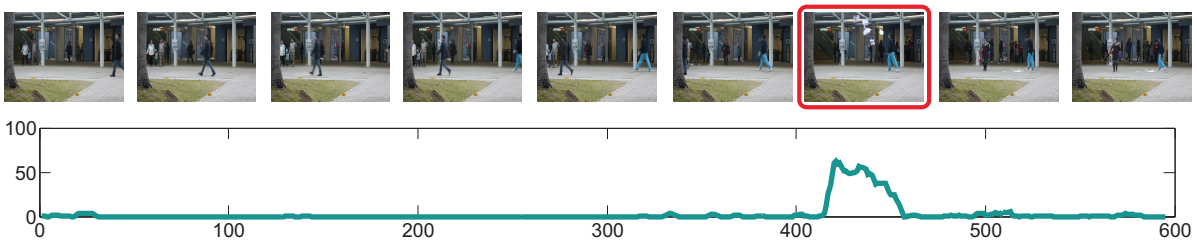
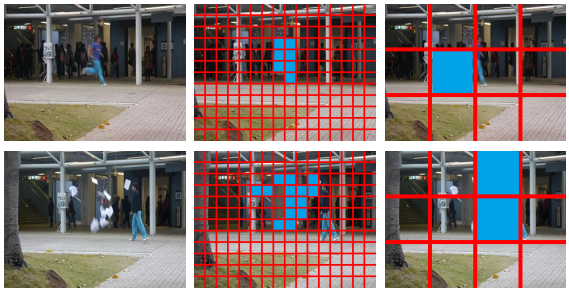
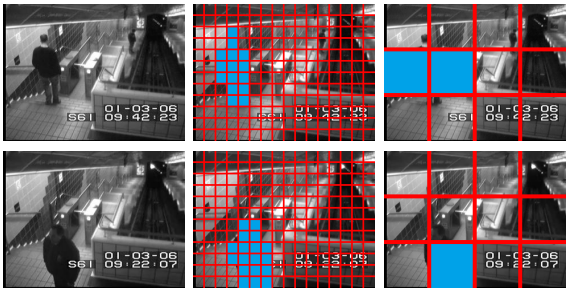


Figure 6. Detection results in a video sequence. The bottom plot is the response. A peak appears when an abnormal event – paper throwing – happens. The x value indexes frames and y-index denotes response strength.



(a) events (b) maps (c) maps

Figure 7. Two abnormal events and their corresponding abnormal patches under two different scales in the Avenue dataset.



(a) events (b) maps (c) maps

Figure 8. Subway dataset (Exit-Gate): Three abnormal events and their corresponding detection maps in two different scales in the Subway-Exit video.

	WD	LT	MISC	Total	FA
Ground Truth	9	3	7	19	0
[22]	9	3	7	19	2
[10]	9	3	7	19	3
[5]	9	-	-	-	2
subspace	6	3	5	14	4
Ours	9	3	7	19	2

Table 2. Comparison with other sparsity-based methods [22, 5] on the Exit-Gate Subway dataset. WD: wrong direction; LT: loitering; FA: false alarm. "-" means the results are not provided. Subspace: results by replacing our combination learning by subspace clustering [7].

	WD	NP	LT	II	MISC	Total	FA
GT	26	13	14	4	9	66	0
[22]	25	9	14	4	8	60	5
[10]	24	8	13	4	8	57	6
[5]	21	6	-	-	-	-	4
subspace	21	6	9	3	7	46	7
Ours	25	7	13	4	8	57	4

Table 3. Comparison using the Subway-Entrance video with several previous methods. GT: ground truth; WD: wrong direction; NP: no payment; LT: loitering; II: irregular interactions; MISC: misc; FA: false alarm. "-" means results are not provided. Subspace: replacing our combination learning by subspace clustering [7].

3.4. Subway Dataset

We conduct quantitative comparison with previous methods on the Subway dataset [1]. The videos are 2 hours long in total, containing 209,150 frames with size 512×384. There are two types of videos, i.e., "exit gate" and "entrance gate" videos.

Exit Gate The subway exit surveillance video contains 19 different types of unusual events, such as walking in the wrong direction and loitering near the exit. The video sequence in the first 15 minutes is used for training. This configuration is the same as those in [10, 22].

The abnormal event detection results for a few frames are shown in Fig. 8. Table 2 lists the comparison with other methods. Our false alarm rate is low mainly because each combination can construct many normal event features, thus reducing the chance of constructing an abnormal structure with a small error. This representation tightens feature modeling and makes it not that easy to misclassify abnormality as normal events. In this dataset, our combination number K varies from 1 to 56 for different cube features.

	Second/Frame	Platform	CPU	Memory
[22]	2	MATLAB 7.0	2.6 GHz	2.0GB
[5]	4.6	-	2.6 GHz	2.0GB
Ours	**0.00641**	MATLAB 2012	3.4 GHz	8.0GB

Table 4. Running time comparison on the Subway dataset.

Entrance Gate In this video, again, the first 15 minutes are used for training. Detection statistics are listed in Table 3. Our results are comparable to those of [22, 10, 5]. The proposed method yields high detection rates together with

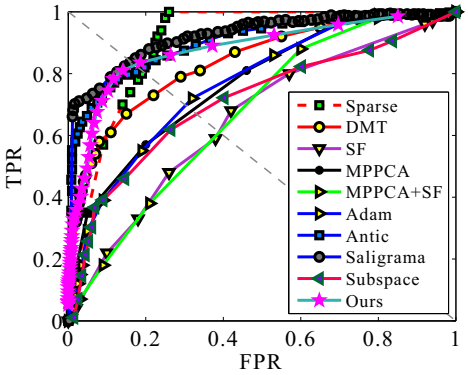


Figure 9. Frame-level comparison of the ROC curves in UCSD Ped1 dataset. Method abbreviation: MPPCA+SF [13], SF [13], MDT [13], Sparse [5], Saligrama [16], Antic [2], Subspace: replacing our combination learning by subspace clustering [7].

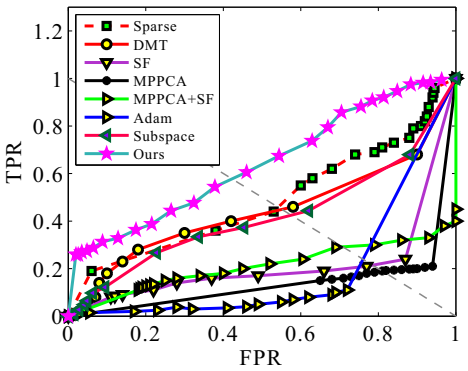


Figure 10. Pixel-level comparison of the ROC curves in UCSD Ped1 dataset. Method abbreviation: MPPCA+SF [13], SF [13], MDT [13], Sparse [5], Saligrama [16], Antic [2], Subspace: replacing our combinations learning by subspace clustering [7].

low false alarm.

Running Time Comparison We compare our system with other sparse dictionary learning based methods [22, 5] in terms of running time on the Subway dataset in Table 4. The speed of methods [22, 5] is reported in their respective papers. The difference on detection speed is much larger than that of working environment.

3.5. UCSD Ped1 Dataset

The UCSD Ped1 dataset [13] provides 34 short clips for training, and another 36 clips for testing. All testing clips have frame-level ground truth labels, and 10 clips have pixel-level ground truth labels. There are 200 frames in each clip.

Our configuration is similar to that of [13]. That is, the performance is evaluated on frame- and pixel-levels. We show the results via ROC curves, Equal Error Rate (EER), and Equal Detected Rate (EDR).

	Second/Frame	Platform	CPU	Memory
[13]	25	-	3.0 GHz	2.0GB
[5]	3.8	-	2.6 GHz	2.0GB
[2]	$5 \sim 10$	MATLAB	-	-
Ours	**0.00697**	MATLAB 2012	3.4 GHz	8.0GB

Table 5. Running time comparison on the UCSD Ped1 dataset.

ROC Curve Comparison According to [13] in frame-level detection, if a frame contains at least one abnormal pixel, it is considered as successful detection. In our experiment, if a frame contains one or more abnormal patches, we label it as an abnormal event. For frame-level evaluation, we alter frame abnormality threshold to produce a ROC curve shown in Fig. 9. Our method has a reasonably high detection rate when the false positive value is low. It is vital for practical detection system development.

In pixel level evaluation, a pixel is labeled as abnormal, if and only if the regions it belongs to in all scales are abnormal. We alter threshold for all pixels. Following [13], if more than 40% of truly anomalous pixels are detected, the corresponding frame is considered as being correctly detected. We show the ROC curve in Fig. 10. Besides all methods that are compared in [13], we also include the performance of subspace clustering [7]. Our method achieves satisfactory performance.

EER and EDR Different parameters could affect detection and error rates. Following [13], we obtain these rates when false positive number equals to the missing value. They are called equal error rate (EER) and equal detected rate (EDR). We compute the area under the ROC curve (AUC). We report EER, ERD and AUC in the pixel-level comparison (Table 6) and calculate EER and AUC in the frame-level (Table 7). These results indicate that our results are with high quality in both measures.

We compare the running time in Table 5. The detection time per frame and working platforms of [13, 5, 2] are obtained from the original papers.

3.6. Separate Cost Analysis

Our testing includes two main steps: feature extraction (3D cube gradient computing and PCA) and combination testing using Algorithm 2. Other minor procedures are frame resizing, matrix reshape, etc. We list the average running time spent for each step to process one frame in the three datasets in Table 8.

4. Conclusion

We have presented an abnormal event detection method via *sparse combination learning*. This approach directly learns sparse combinations, which increase the testing speed hundreds of times without compromising effective-

	SF [13]	MPPCA [13]	SF-MPPCA [13]	MDT [13]	Sparse[5]	Adam[1]	Antic [2]	Subspace [7]	Ours
EDR	21 %	18 %	18 %	45 %	46 %	24 %	68 %	39.3%	59.1 %
AUC	19.7 %	20.5 %	21.3 %	44.1 %	13.3%	46.1 %	76 %	43.2 %	63.8 %

Table 6. Comparison of pixel-level EDR and AUC curves on the UCSD Ped1 dataset.

	SF-MPPCA [13]	SF [13]	MDT [13]	Sparse[5]	Saligrama [16]	Antic [2]	Subspace [7]	Ours
EER	40 %	31 %	25 %	19 %	16 %	18 %	29.6%	15 %
AUC	59 %	67.5 %	81.8%	86 %	92.7 %	91 %	68.4 %	91.8 %

Table 7. Comparison of frame-level EER and AUC curves on the UCSD Ped1 dataset.

	Feature extraction (ms)	Combinations testing (ms)	Others (ms)	All (ms)	FPS
Avenue	4.513	1.792	0.770	7.075	141.34
UCSD Ped1	4.496	1.724	0.743	6.965	143.57
Subway	4.634	1.409	0.625	6.412	155.97

Table 8. Average running time of processing one frame in each step on the three datasets. "ms" is short for millisecond.

ness. Our method achieves state-of-the-art results in several datasets. It is related to but differ largely from traditional subspace clustering. Our future work will be to extend the sparse combination learning framework to other video applications.

Acknowledgments

This research has been supported by General Research Fund (No. 412911) from the Research Grants Council of Hong Kong.

References

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE TPAMI*, 30(3):555–560, 2008.

[2] B. Antic and B. Ommer. Video parsing for abnormality detection. In *ICCV*, pages 2415–2422, 2011.

[3] Y. Benezeth, P.-M. Jodoin, V. Saligrama, and C. Rosenberger. Abnormal events detection based on spatio-temporal co-occurences. In *CVPR*, 2009.

[4] D. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, MA, 1999.

[5] Y. Cong, J. Yuan, and J. Liu. Sparse reconstruction costs for abnormal event detection. In *CVPR*, pages 3449–3456, 2011.

[6] X. Cui, Q. Liu, M. Gao, and D. Metaxas. Abnormal detection using interaction energy potentials. In *CVPR*, pages 3161–3167, 2011.

[7] E. Ehsan and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.

[8] F. Jianga, J. Yuan, S. A. Tsaftarisa, and A. K. Katsaggelosa. Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, 115(3):323–333, 2011.

[9] K. Jouseok and L. Kyoungmu. A unified framework for event summarization and rare event detection. In *CVPR*, 2012.

[10] J. Kim and K. Grauman. Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In *CVPR*, pages 2921–2928, 2009.

[11] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *CVPR*, pages 1446–1453, 2009.

[12] C. Lu, J. Shi, and J. Jia. Online robust dictionary learning. In *CVPR*, 2013.

[13] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *CVPR*, 2010.

[14] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[15] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *CVPR*, 2009.

[16] V. Saligrama and Z. Chen. Video anomaly detection based on local statistical aggregates. In *CVPR*, pages 2112–2119, 2012.

[17] J. Shi, X. Ren, G. Dai, J. Wang, and Z. Zhang. A non-convex relaxation approach to sparse dictionary learning. In *CVPR*, pages 1809–1816, 2011.

[18] H. Trevor, T. Robert, and J. H. Friedman. *The elements of statistical learning*. Springer New York, 2001.

[19] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception by hierarchical bayesian models. In *CVPR*, pages 1–8, 2007.

[20] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In *CVPR*, 2010.

[21] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *CVPR*, 2005.

[22] B. Zhao, L. Fei-Fei, and E. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *CVPR*, 2011.

[23] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR*, 2004.