Detecting Unusual Activity in Video

Hua Zhong
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
zhonghh@cs.cmu.edu

Jianbo Shi Mirkó Visontai
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{jshi,mirko}@cis.upenn.edu

Abstract

We present an unsupervised technique for detecting unusual activity in a large video set using many simple features. No complex activity models and no supervised feature selections are used. We divide the video into equal length segments and classify the extracted features into prototypes, from which a prototype–segment co-occurrence matrix is computed. Motivated by a similar problem in document-keyword analysis, we seek a correspondence *relationship between prototypes and video segments which satisfies the transitive closure constraint. We show that an important sub-family of correspondence functions can be reduced to co-embedding prototypes and segments to N-D Euclidean space. We prove that an efficient, globally optimal algorithm exists for the co-embedding problem. Experiments on various real-life videos have validated our approach.*

1. Introduction

Imagine you are given a long video, possibly thousands of hours in duration, depicting every day scenes like those shown in figure 1. You are asked to analyze the video to find *unusual* events. In surveillance applications unusual events are those that should be reported for further examinations. What makes unusual events hard to detect? Unusual events are rare, difficult to describe, hard to predict and can be subtle. However, given a large number of observations it is relatively easy to verify if they are indeed unusual.

1.1. Model-based approach

This "*hard to describe*" but "*easy to verify*" property of unusual events suggests an intuitive two-step solution for their detection. In the first step, one extracts image features from the video, typically achieved by detecting and tracking moving objects [14]. From tracked objects trajectory, speed, and possibly the shape descriptor of the moving objects can be computed [6]. In the second step the extracted features are used to develop models for the "normal" activities, either by hand or by applying supervised machine learning techniques [7]. A common choice is to use Hidden Markov Models [1, 9, 10] or other graphical models [11] which quantize image features into a set of discrete states and model how states change in time. In order to detect unusual events the video is matched against a set of normal models and segments which do not fit the models is considered unusual.

(a) Nursing home (b) Road (c) Poker game

Figure 1: Snapshots from the videos used for experiments.

This model-based approach can be quite effective in situations where "normal" activity is well-defined and constrained. However in a typical real-life video, like those used in our experiments, the number of different "normal" activity types observed can easily surpass the number of unusual types. Hence, defining and modeling what is the "normal" activity in an unconstrained environment can be more difficult than defining what is unusual. If the goal is to detect what unusual events in a long video, the model-based approach is often over-kill.

1.2. Unsupervised approach

We propose a method to utilize the "*hard to describe*" but "*easy to verify*" property of unusual events without building *explicit models* of normal events. One can compare each event with all other events observed to determine how many *similar* events exist. If an event is normal, there should be many similar events in this large data set. If there are no similar events we consider this unusual: although the event is unknown, it is different from the others. Thus, detecting unusual events in a large data set does not require modeling normal events, but rather the ability to compare two events and measure their similarity.

1.3. Feature selection

How do we select the right feature set for event comparison? We must make a compromise between two contradictory desires. On one hand we would like features to be as descriptive as possible: measuring the kinematics and dynamics of the object's movements is very useful for event comparison. On the other hand, we also want feature extraction to be extremely robust across many hours of video. Descriptive features are hard to extract. Object detection and tracking often fails in an unconstrained environment. Basic image features based on spatial/motion histogram of objects are simple and reliable to compute [2, 5, 15]. The only drawback of these methods is that the (important) feature signal might be obscured by noise. Event similarity computed naively could be overestimated, making unusual events appear similar to common ones. This over-dependence on the feature set has been a general weakness for most unsupervised approaches [13].

The situation is vastly improved if we can extract the important feature signal from a large set of simple features. This problem resembles the problem of unusual event detection itself as important signals are "*hard to detect*" but "*easy to verify*". In fact, unusual event detection and important feature selection are two interlocked problems. We propose a *correspondence* function to measure such mutual interdependence thereby detecting unusual events and important features simultaneously. We show that for an important subfamily of correspondence functions an efficient computational solution exists via co-embedding.

The paper is organized as follows: in Section 2 we show our video representation. In Section 3 and 4 we describe our algorithm for unusual event detection. In section 5 we present our experimental results, and we conclude our paper in Section 6.

2. Video Representation

Our overall goal is to extract simple and reliable features which are descriptive, i.e. can be used by an unsupervised algorithm to discover the important image features in a large video set in order to detect unusual events.

2.1. Video Segmentation

Given a video $\mathbf{V}$, we first slice it into N short segments: $\mathbf{V} = (v_1, v_2, ..., v_N)$. Ideally, each segment v_j would contain a single activity event. For simplicity, we slice the video into fixed time duration (4 seconds) with overlapping time window. The segmentation is not perfect, but the video segments typically contain enough information for determining the activity type, e.g. in the nursing home video, in 4 seconds people can take a few steps or pick up an object.

2.2. Image features

For each image frame in the video we extract objects of interest, typically moving objects. We make no attempt to track objects. The motion information is computed directly via spatiotemporal filtering of the imageframes: $I_t(x, y, t) = I(x, y, t) * G_t * G_{x,y}$, where $G_t = te^{-(\frac{t}{\sigma_t})^2}$ is the temporal Gaussian derivative filter and $G_{x,y} = e^{-((\frac{x}{\sigma_x})^2+(\frac{y}{\sigma_y})^2)}$ is the spatial smoothing filter. This convolution is linearly separable in space and time and is fast to compute. To detect moving objects, we threshold the magnitude of the motion filter output to obtain a binary moving object map: $M(x, y, t) = ||I_t(x, y, t)||_2 > a$. The process is demonstrated in figure 2(a)-(b).

(a) (b) (c)

Figure 2: Feature extraction from video frames. (a) original video frame from the card game sequence. (b) binary map of objects (c) spatial histogram of (b).

The image feature we use is the spatial histogram of the detected objects. Let $H_t(i, j)$ be an $m \times m$ spatial histogram, with m typically equal to 10. $H_t(i, j) = \sum_{x,y} M(x, y, t) \cdot \delta(b_i^x \leq x < b_{i+1}^x) \cdot \delta(b_j^y \leq y < b_{j+1}^y)$, where $b_i^x, b_j^y (i, j = 1 \ldots m)$ are the boundaries of the spatial bins. The spatial histograms, shown in 2(c), indicate the rough area of object movement. Similarly, we can compute a motion and color/texture histogram for the detected object using the spatiotemporal filter output. As we will see, these simple spatial histograms are sufficient to detect many complex activities.

2.3. Prototype features

The feature space of $m \times m$ motion histograms is still too large. To detect potentially important feature signals we apply vector quantization to the histogram feature vectors classifying them into a dictionary of K *prototype features*, $\mathbf{P} = \{p_1, \ldots, p_K\}$ using K-means (figure 3).

Note that using this discrete quantization, it is possible that two image frames with similar features will have differ-

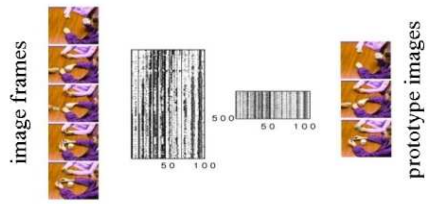


Figure 3: Create prototypes from set of features

ent prototype labels. In this case, the two prototypes must also be similar. This problem is resolved, as we will see later, by keeping track of similarity among the prototypes.

2.4. Video–prototype co-occurrence

A video segment v_j can be represented by the occurrence of prototype features in that video segment. We define a co-occurrence matrix $C \in R^{K \times N}$ between the video segments and prototype features in the following way: $C(i, j) = 1$ iff segment v_j contains and image frame whose image feature is classified as prototype p_i. Figure 4 shows an example for the roadway sequence. C contains all necessary information about what happened in the video. Each row i of C reveals which video segment a prototype p_i occurred in, and each column j of C indicates which prototype features occurred in the video segment v_j.

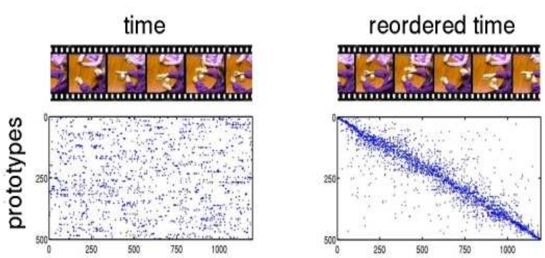


Figure 4: (left) Video-prototype co-occurrence matrix C contains all necessary video information. (right) Reorganization of rows and columns of C reveals the underlying correspondence between thevideo segments and prototype features, making it easy tofind both unusual events and important features.

3. Algorithm

If we think of the video segments as documents, and prototype features as keywords we observe that similar problems arise in the context of document-keyword clustering [8, 4]. To illustrate the algorithm we propose, consider the following example. We took 20 departmental emails and 6 research emails between the authors as our document set, and words which occured more then once (across the 26 emails) as our keyword set. In the example we want to find

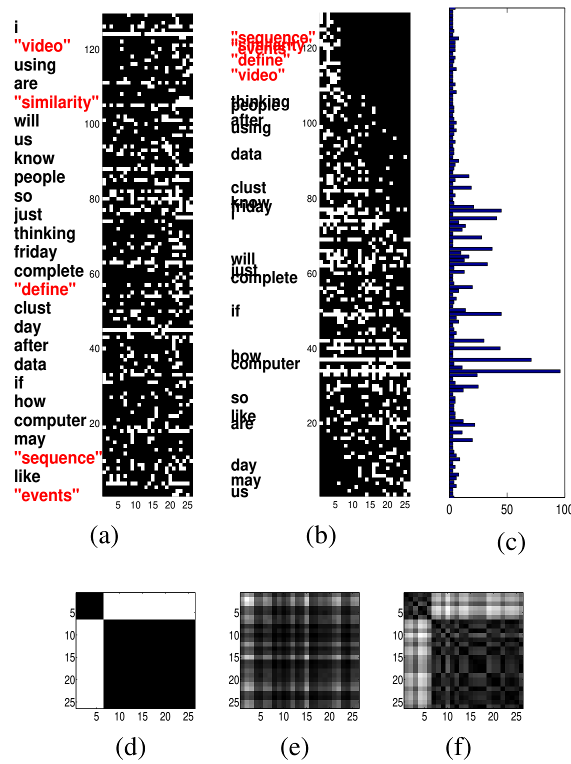


Figure 5: (a) Email-word co-occurrence matrix. (b) A reorganized version of (a), where distinctive keywords related to research emails are quoted. (c) Word occurrence count for corresponding words in (b). (d)-(f) inferred email-email similarity matrices: (d) ideal case, (e) without feature selection, (f) with feature selection.

the cluster of the documents (such as the 6 emails) based on the co-occurrence (figure 5(a)) information. In the analogous video setting we seek the clusters of similar video segments. In figure 5(b) we show that there is an optimal reordering of the keywords and documents (reshuffling rows and columns of C) such that the clustering of documents and keywords can be readily inferred from the reshuffled version of co-occurrance matrix. We can see that the top 20 words such as “sequence” or “define” are exclusively correlated with the first 6 emails, which are the research emails. To spot the 6 research emails, it is sufficient to check for these 20 keywords, which do not occur in other emails. Examining the rest of matrix C, we see that majority of the words do not have exclusive correlation with any particular group of emails. These include words such as “will” or “if”, which occur with almost every email, and words “data” and “people”, which occur randomly.

To cluster documents it is important to determine the informative keywords, which might be only a small subset of the total keywords. Such feature selection is vital in clustering the video segments as well.

3.1. How to extract document clusters from co-occurrance matrix C?

To contrast our algorithm let us compare it with Latent Semantic Indexing (LSI, [3]), a commonly used approach in Natural Language Processing. LSI works in two steps. First it computes the singular value decomposition of the document–keyword co-occurrance matrix: $C_{K\times N} = U_{K\times c}\Sigma_{c\times c}V^T_{c\times N}$. The singular vectors V defines a new feature space where documents can be clustered using similarity defined by VV^T. Similarly, the keywords can be clustered using UU^T. Note that the SVD of C is equivalent to the eigendecomposition $C^TC = V\Sigma^2V^T$. Therefore the clustering information used is contained in C^TC. Expanding out the terms, we see that $(C^TC)(i,j) = \sum_{k=1}^{K} C(k,i)C(k,j)$ defines a document–document similarity measure by counting the number of keywords occuring in both documents i and j. As we have seen in the email example using the complete set of keywords as a basis for similarity measure (5(e)) can obscure the true similarity (5(d)) provided by the informative keywords.

A remedy to this problem is to try to identify non-informative keywords, e.g computing the keyword occurrance frequency count. This is insufficient. As we can see in figure 5(c) the non-informative keywords can have both high frequency (common keywords) and low frequency (random keywords).

3.2. Bipartite graph coclustering

Clustering based on similarity information in C^TC can be though of as the following bipartite graph coclustering. This concept has been explored in the document–keyword analysis by Dhillon [4] and in bioinformatics by Kluger et al. [8]. The document and keywords represent the two sets of nodes in the bipartite graph, and the co-occurring documents and keywords are connected by graph edges. The graph has an edge weight matrix $\begin{pmatrix} 0 & C \\ C^T & 0 \end{pmatrix}$. These methods are based on finding the normalized cut in this graph. Intuitively, this is the right thing to do, we want to select a set of keyword for grouping a particular set of documents. Furthermore, this grouping gives the correspondence between the informative keywords and relevant documents. However, graphcuts on bipartite graphs amounts to *separate* clustering on documents and on keywords, given by the fact that $\begin{pmatrix} 0 & C \\ C^T & 0 \end{pmatrix}$ and $\begin{pmatrix} CC^T & 0 \\ 0 & C^TC \end{pmatrix}$ have the same eigenvectors. Thus, finding the optimal partition on the bipartite graph contradicts the concepts of coclustering and simply results in clustering the documents on the information given by C^TC.

3.3. Document–keyword correspondence

Our goal is to cluster documents by identifying exclusive correlations between keywords and documents. We can think of this problem as graph editing. Edges incident on common and random keywords must be removed, otherwise the entire graph would be densely connected. Edges incident on informative keywords should be preserved, otherwise the graph would be sparsely connected. A brute force strategy would be to search through all possible addition and deletion of edges such that the resulting graph partitioning is neither too coarse nor too fine.

To make this task precise we define a correspondence function $CP(p_i, v_j) \in [0,1]$ between a prototype feature p_i and video segment v_j with following properties: $CP(p_i, v_j)$ should be consistent through transitivity, i.e. $CP(p_i, v_j)$ should be high if there is a chain of correspondences linking them indirectly. We can express this consistency constraint using the notion of *transitive closure*. To allow continuous values in $CP(p_i, v_j)$, we introduce α**-transitive-closure**: $\forall_{i_1,i_2,j_1,j_2}$ and $\forall_{\delta\in[0,1]}[CP(p_{i_1}, v_{j_1}), CP(p_{i_1}, v_{j_2}), CP(p_{i_2}, v_{j_1}) > 1-\delta] \Rightarrow [CP(p_{i_2}, v_{j_2}) > 1-\alpha\delta]$. Transitive closure assures that no random or common features are in any correspondence relationship.

In addition, $CP(p_i, v_j)$ should be close to the co-occurrence $C(p_i, v_j)$, i.e. the graph editing should be minimal. We maximize

$$E_C(CP) = \sum_{p_i\in\mathbf{P}, v_j\in\mathbf{V}} C(p_i, v_j)CP(p_i, v_j) \qquad (1)$$

under certain constraints that prevent the trivial solution.

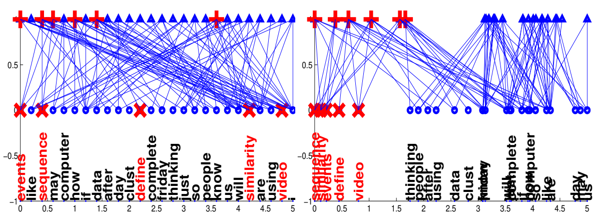
3.4. Correspondence via co-embedding prototypes and video segments

Computing a correspondence relationship satisfying transitive-closure is in general a difficult problem. Therefore we restrict our search to an important sub-family of functions of $CP(p_i, v_j)$, for which we can compute it efficiently. Let $x : \mathbf{V}\cup\mathbf{P} \to \mathrm{R}^{\mathbf{N}}$ denote the *co-embedding* of the prototypes and video segments. The correspondence function $CE(p_i, v_j) = 1 - (x(p_i) - x(v_j))^2$ satisfies α-transitive closure with $\alpha = 9$. By restricting our search for correspondence to CE, we can rewrite the optimization problem in (1):

$$\begin{aligned} \max E_C(CE) &= \textstyle\sum_{p_i,v_j} C(p_i, v_j)CE(p_i, v_j) \Leftrightarrow \\ \min E'_C(x) &= \frac{\sum_{p_i,v_j} C(p_i,v_j)(x(p_i)-x(v_j))^2}{\sigma_x^2}, \end{aligned} \qquad (2)$$

where σ_x^2 is the standard deviation of vector x (it removes an arbitrary scaling factor in the embedding and also prevents the trivial solution of $x = 0$).

The co-embedding optimization problem in (2) can be visualized as finding a placement (embedding) vector x for each prototype and video segment in a low dimensional space. We can imagine the co-occurrence relationships as springs that pull together (or apart) prototypes and video segments, the nodes of figure 6(a), resulting in the co-occurring elements being *maximally aligned*, as shown in figure 6(b). Maximal alignment in this case means that corresponding nodes are placed close to each other (on x axis). In order to arrange the prototypes in the $\mathbf{N}$-D space, we need to know the position of the video segments, and vice versa. Note that the chicken–egg nature of our unusual event detection is inherent in both the correspondence and the embedding problem. Luckily, we can break this chicken-egg deadlock in an optimal and computationally efficient way.



(a) (b)

Figure 6: Co-embedding of email-word example. The top row represents emails, the bottom words, the edges are the co-occurrence between them. (a) random embedding, (b) optimal co-embedding using our method. The springs pull corresponding elements in alignment.

To compute the optimal co-embedding, we define a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We take prototypes *and* video segments as nodes, $\mathcal{V} = \{v_1, ..., v_N\} \cup \{p_1, ..., p_K\}$. The edges of this graph consist of edges between prototypes and video segments which represent the co-occuring relationship (C), and edges between the prototype nodes, which represent the similarity $S_p \in \mathrm{R}^{K \times K}$ between the prototypes: $\mathcal{E} = \{(p_i, v_j) | C(i,j) \neq 0\} \cup \{(p_i, p_j)\}$. The edge weight matrix is defined as:

$$W = \left[\begin{array}{cc} I & C^T \\ C & \beta S_p \end{array} \right] \in \mathrm{R}^{|\mathcal{V}| \times |\mathcal{V}|} \tag{3}$$

where β is a weighting factor. When $\beta = 0$, minimization of (2) is equivalent to minimization of

$$E_W(x) = \frac{\sum_{(i,j) \in \mathcal{E}} W(i,j)(x(i) - x(j))^2}{\sigma_x^2} \tag{4}$$

Expanding the numerator of (4) we get $2x^T(D - W)x$, where $D \in \mathrm{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is a diagonal matrix with $D(i,i) = \sum_j W(i,j)$. Using the fact that

$$\sigma_x^2 = \sum_{i \in \mathcal{V}} x^2(i) P(i) - (\sum_{i \in \mathcal{V}} x(i) P(i))^2 \tag{5}$$

where $P(i)$ is an estimate of the prior occurrence likelihood of each prototype or video segment, which can be estimated by their occurrance frequency: $P(i) = \frac{1}{\gamma} D(i,i)$, with $\gamma = D1$. Centering the embedding around zero (i.e. $x^T D 1 = 0$), we get $\sigma_x^2 = \frac{1}{\gamma} x^T D x$.

Putting all these together, we can rewrite (4) as

$$E_W(x) = 2\gamma \frac{x^T (D - W) x}{x^T D x} \tag{6}$$

The global energy minimum of this function is achieved by the eigenvector corresponding to the second smallest eigenvalue of

$$(D - W)x = \lambda D x \tag{7}$$

Note that the first N elements of vector x contain the coordinates of the video segment nodes, and the next K elements contain the coordinates of the prototype features. Incidentally, this is the same eigenvector used in a different context of image segmentation (Normalized Cuts, [12]).

3.5. Importance of prototype–prototype similarity

When $\beta \neq 0$, the S_p matrix gives us additional cues on how the prototypes are related to each other, thereby providing information on how the video segments should be clustered. In the email example, we used the word-length as cue for keyword similarity and with this cue we are able to pick up the right clustering of the emails (figure 5(f)). Furthermore, the S_p also helps to alleviate the sensitivity to over-clustering of features into prototypes, by strengthening the correspondence between the segments which have similar but not necessarily identical prototypes.

3.6. Unusual event detection algorithm

Unusual events are the video segments that have correspondences to distinctive important features, and vice versa. Using the co-embedding coordinate x computed in (7), we define *inferred similarity* among the video segments v_i, v_j as: $S_x(v_i, v_j) = e^{-||x(v_i) - x(v_j)||_2}$, as seen in figure 5(f). In the co-embedding space, detecting unusual video segments is done by finding spatially isolated clusters. We choose the following simple solution for this:

- We apply K-means algorithm on the embedding coordinates x to cluster video segments into disjoint sets $\{VC_1, \ldots, VC_c\}$,
- and we compute inter-cluster similarity on $S_{VC}(VC_i, VC_j) = \sum_{v_i \in VC_i, v_j \in VC_j} S_x(v_i, v_j)$.
- The clusters VC_k, with small total inter-cluster similarity value, $\sum_i S_{VC}(VC_i, VC_k) < \theta$, are considered as unusual events.

4. Algorithm summary

The outline of our algorithm is the following:

- for each frame t, detect moving object and extract motion and color/texture histogram: $H_t \in \mathrm{R}^{m \times m}$.
- quantize every histogram H_t into K prototypes: $\mathbf{P} = \{p_1, \ldots, p_K\}$
- slice the video into T second long segments: $\mathbf{V} = \{v_1, \ldots, v_N\}$
- compute the co-occurrence matrix $C(i, j) \in \mathrm{R}^{K \times N}$ between each prototype feature p_i and video segment v_j
- compute pairwise similarity between all prototypes using chi-square difference: $S_p(p_i, p_j) = \chi^2(H(p_i), H(p_j))$
- construct graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with associated weight matrix $W = \begin{bmatrix} I_{N \times N} & C^T \\ C & \beta S_p \end{bmatrix} \in \mathrm{R}^{(N+K) \times (N+K)}$.
- solve for the smallest eigenvectors of $(D - W)x = \lambda D x$, the first N rows of the eigenvectors are the coordinates for v_js in the embedding space, and the following K rows are the coordinates for p_is in the embedding space.
- in the co-embedding space compute the *inferred similarity*. Identify isolated clusters of v_js as unusual events.

4.1. Computational running time

The object detection is done by a spatiotemporal separable convolution (section 2). Hence the cost is: $O(n_{fr} \cdot n_{pix})$, where n_{fr}, n_{pix} are the number of frames in the video, the number of pixels in an image frame. The complexity of the K-means algorithm is $O(n_{fr} \cdot K \cdot m^2)$, where K, m, i_k are respectively the number of prototype vectors, the size of the histograms ($m \times m$) and the number of iterations. Building the co-occurrance and the similarity matrix has a cost of $O(n_{fr} + d \cdot K^2)$. Finally finding the second eigenvector of a symmetrical sparse $n \times n$ matrix takes $O(n^{\frac{3}{2}})$ time, where $n = N + K$. For example, the running times for the 20 hours road video are: 8 hours 40 minutes (object detection), 1 hours 36 minutes (K-means) and 3 seconds (eigensolver) on a Pentium IV 2.4GHz.

5. Experiments

In order to demonstrate the algorithm we conducted the following tests on various test data. Table 1 gives a short summary of the different tests. In the following experiments for vector quantization we used $K = 500$ prototypes, for segment length $T = 4s$ and β is defined as $1/M$ where M is the biggest value in S_p.

The first testset is a video shot in a dinning room of a hospital. After removing the motionless frames, we still had 169 880 frames. We tested our embedding algorithm to see if it provides a good separation between different events. We observed that the unusual activities are embedded far from the usual ones, as can be seen in figure 7.

Title	Duration	Type of test
Road	19h50min	surveillance
Poker game	30min	cheating detection
Hospital	12h	patient monitoring
Webcam	3h	analyzing the crowd

Table 1: The test videos used in our experiments

To quantify the "goodness" of the embedding provided in our previous experiment we used another video from a surveillance camera overlooking a road adjacent to a fenced facility. We have tested our system on a continuous video from 16:32pm till 12:22pm the next day, containing both day time and night time videos (in total 1 063 802 image frames). We applied our embedding algorithm and classified the embedded segments into two groups, i.e. usual and unusual. To measure the performance we hand-labeled all the sequences (which contained motion) if they were unusual or not and compared our results to the ground truth. The promising results of this experiment are shown in figure 8. Though, this surveillance sequence is somewhat limited in the type of actions it contains (particularly it has just 23 unusual sequences), we would like to point out that even without motion features, i.e. only with spatial histograms, we were able to detect events such as cars making U-turns, backing off, and people walking on and off the road.

Next experiment was aimed to measure the performance in a more complex setting: we recorded a 30 minutes long poker game sequence, where two players were asked to creatively cheat. The video contains 17 902 frames, and every 4 second hand-labelled with one of the 27 activity labels. There is a wide variety of natural actions, in addition to playing cards and cheating, players were drinking water, talking, hand gesturing, scratching. Many of the cheatings are among detected unusual events. To demonstrate we can detect a specific cheating type, we find those unusual events corresponding to a prototype feature chosen by us. The results of detecting two cheating types are shown in figure 9.

To show that the algorithm can be used for categorizing usual events as well we took 3 hours long video from Berkeley Sproul Plaza webcam (http://www.berkeley.edu/webcams/sproul.html), which contained 28 208 frames. The embedding of video segments, and event category representatives are shown in figure 10 (left). The automatic categorization of events potentially can allow us to develop a statistical model of activities, in an unsupervised fashion.

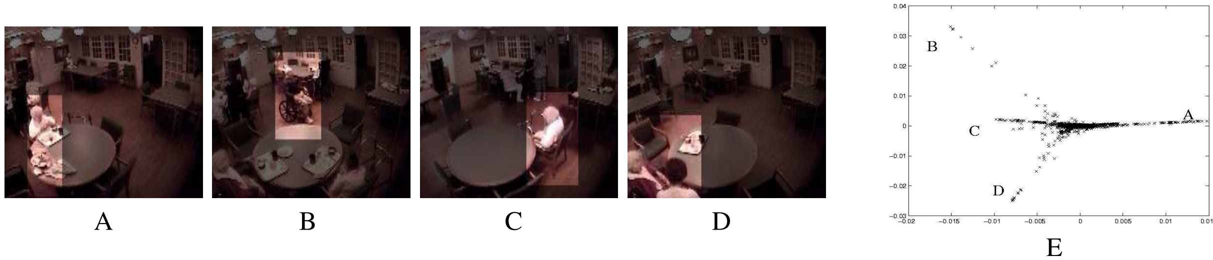


Figure 7: Four unusual activities being discovered, corresponding to four remote clusters in the embedding space: A: a patent eating alone at the near table, B: a man on wheel chair slowing goes in and out of the room while everyone else is eating, C: a patient shaking, D: a nurse feeding a patient one-on-one with no one around. E: the 2-D embedding of the video segments.

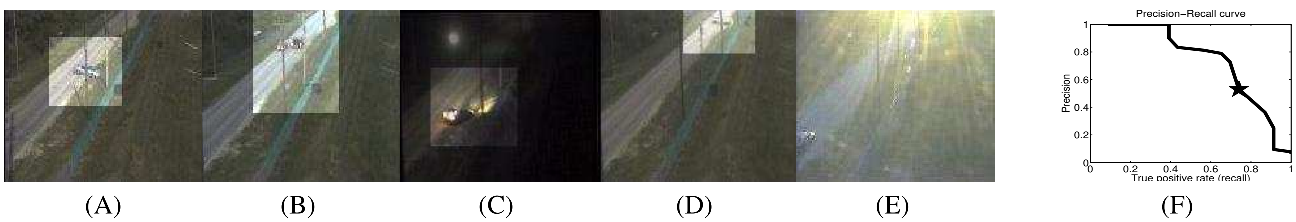


Figure 8: Results for 20 hours long road surveillance video. Usual events consist of cars moving along the road. **Correctly detected** unusual events include: (A) cars pulling off the road, (B) cars stopping and backing up, (C) car making U-turns, and people walking on the road. **Undetected** unusual events include: (D) cars stopping on the far end, due to coarseness of spatial feature. **False-positives** include mainly birds flying by, and direct sunlight into camera (E). the **Precision-Recall curve** of the results is shown in (F). The star indicates the operating condition achieving the precision/false positive and the precision/recall trade-off shown in (A)-(E).

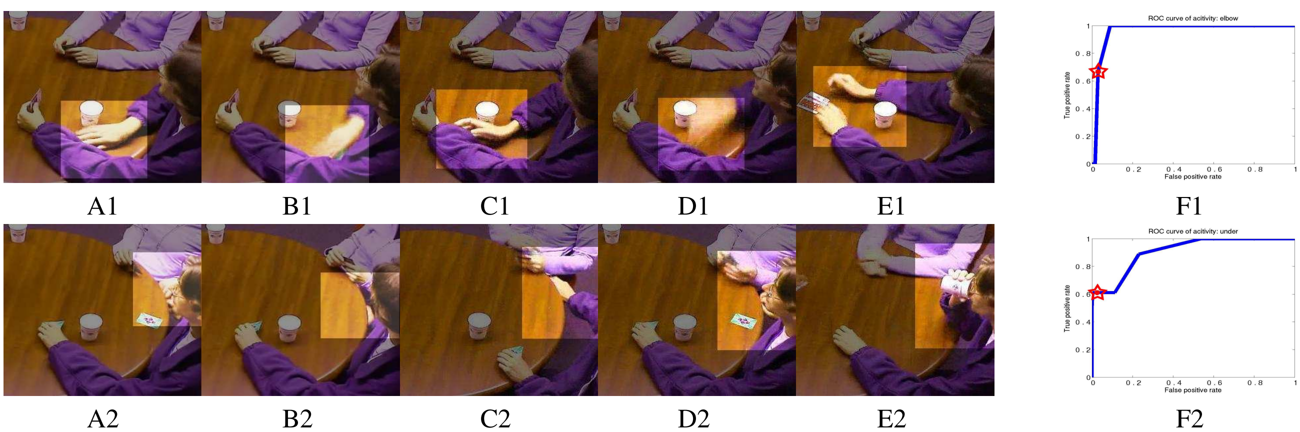


Figure 9: **"Elbow" cheating detection**. A1,B1,C1: examples of detected cheatings, "near player" reaches to his left elbow to hide a card. D1: non-detected cheating, "near player" reaching to his elbow but doesn't hide anything; E1: false positives, the "near player" makes different movement with his hand. **"Under" cheating detection**. A2,B2,C2: example of detected events, two players exchange cards under the table; D2: non-detected cheatings, the exchange is mostly occluded. E2: false positives - the near player is drinking, due to camera angle his hand is in similar position. F1, F2: ROC curves of the two events: The red stars indicate the operating condition for results shown here.

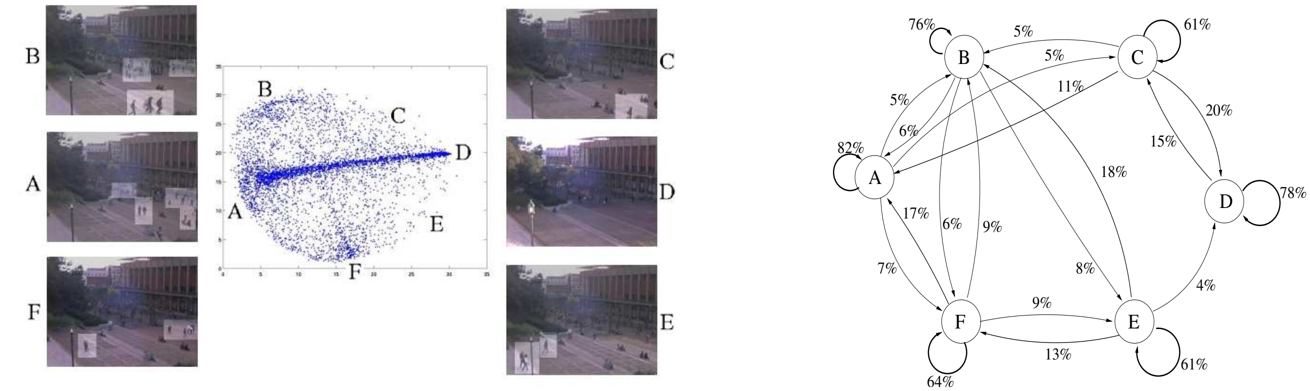


Figure 10: (left) The embedding of the webcam video show videos are best organized by two independent event types in the scene. The horizontal axis (A-D) represents crowd movements along the building: many people walking (A), and few or no people walking (D). In the vertical axis (B-F) events of walking in/out of Sproul Hall are detected, and are organized according to which orientation people entered/left: (B) along the bottom of image frame; (F) diagonally from the lower left corner. (E) and (C) are compound events: (E) is combination of event (F) and (D), (C) is combination of (B) and (D). (right) Given the classification of the video into distinct events, a transition model is estimated.

6. Conclusion

We have developed an unsupervised technique for detecting unusual events in a large video set. This method can utilize extremely simple features by automatically selecting the important feature signal. The computational solution is efficient and stable. We have conducted large scale tests with ground truth comparison on a variety of sequences, processing over a million of video frames.

Acknowledgements

We would like to thank Takeo Kanade, Alyosha Efros for helpful suggestions.

References

[1] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[2] J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[3] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *J. of Society for Information Science*, 41(6):391–407, 1990.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 269–274, San Francisco, August 2001.

[5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *International Conference on Computer Vision*, Nice, October 2003.

[6] I. Haritaoglu, D. Harwood, and L. S. Davis. W^4: Who? when? where? what? a real time system for detecting and tracking people. In *International Conference on Face and Gesture Recognition*, Nara, April 1998.

[7] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615, August 1996.

[8] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13(4):703–716, April 2003.

[9] G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):873–889, 2001.

[10] D. J. Moore, I. A. Essa, and I. Monson H. Hayes. Exploiting human actions and object context for recognition tasks. In *International Conference on Computer Vision*, volume 1, pages 80–86, Corfu, September 1999.

[11] M. R. Naphade and T. S. Huang. A probabilistic framework for semantic indexing and retrieval in video. In *IEEE International Conference on Multimedia and Expo*, 2000.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[13] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[14] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[15] L. Zelnik-Manor and M. Irani. Event-based video analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, December 2001.