Finding (Un)Usual Events in Video

CMU-RI-TR-03-05

Hua Zhong[1]

The Robotics Institute[2]
Dept. Computer Science[1]
Carnegie Mellon University
Pittsburgh, PA 15213

Jianbo Shi[2,3]

Dept. Computer & Information Science[3]

University of Pennsylvania
Philladelphia, PA 19104

Abstract

We propose an algorithm for detecting and categorizing (un)usual human activity in a video which might be a few days long. The proposed approach is unsupervised, and uses the co-occurrence among large number of simple visual image features to define the activity categories, and to identify what are unusual events automatically. A video is divided into short segments(clips), and motion/color histogram is extract for the foreground object for each image frame. The image features are Vector Quantized into a smaller set of prototype features.A weighted graph is constructed by taking *clips* and *prototype features* as nodes, and the co-occurrence relationship between them as the graph edges.We compute an optimal graph embedding that maps the clips and prototype features in a common low dimensional space. This unified embedding ensures that all pair of co-occurring clip and feature are as close as possible. We define event categories by identify cluster of clips in this embedding space, and those isolated clusters are detected as *unusual* events. We can also classify a new video clips based on the embedding of its co-ocurring features. We demonstrated this algorithm on several long surveillance video recorded at a nursing home.

Keyword: Human Activity Detection, Video Event Recognition, Abnormally Detection

1 Introduction

Video surveillance technology today is limited in its abilities to understand human activity. To understand some of the practical difficulties in understanding human activity in a surveillance video, we have examined a hospital video sequence taken over a period of week at multiple locations. This video illustrates a number of difficulties in achieving human activity understanding. First, it is hard to predict what type of activities will occur in the video. Watching the video ourselves, we have found large number of unexpected events, and those events vary day to day. It is not clear how one can identify and summarize all activities in a video if one doesn't know what to look for to begin with. Second, detailed people detection and tracking is very difficult. In each frame of the video, quite often several body parts are occluded.

Many research initiatives are under way. Most researches have been focusing on detecting a set of predefined activities, such as people walking vs. running[6, 2, 4], certain gesture and interaction[3, 18, 21] or specific events such as someone stealing a suitcase[11, 14]. Typically, simple activities are modelled as spatial-temporal patterns, and complex activities are represented as a hierarchical combination of simpler activities conditioned on various temporal constraints[15, 11, 16].

To a large extend, research in this direction has been bogged down by two fundamental problems. First, we are limited in our ability to detect and track human body reliably in the image. Since most activity recognition algorithms require precise information on body limb movements, failure of the body detection and tracking stage has major impact on our final recognition result. Second, there are vast number of actions we need to recognize, and within each action type there is large variation depending on who is carrying it out, and context under which it is carried out.

As an alternative to the model based approach[17, 8], [13, 20] suggested a data-driven approach that *learns* a category of action events from the data, rather than defining it by hand. In [20], they built a hierarchical classifier based on the image feature co-occurrence to classify the video clips into different categories.

We too take a unsupervised data-driven approach. The key concept we use here is the co-occurrence between video clips and image features. This is different from the co-occurrence matrix in [20] which is between the image features. By identifying the correlation between the video clips and image features, we are able to organize the video into groups of similar activities, and find the characteristic image features for detecting each of the activities.

Detecting the correlation between the video clips and image features requires us to solve an chicken-and-egg problem. To discover the action categories, we need to know which are the key image features, and to know which are the key features we need to know the list of action categories. We solve this problem by finding a unified embedding of video clips and image features into a common low dimensional space. The feature-clip co-occurrence relationship acts like springs that pull together(or apart) clips and features, so that the co-occurring video clips and image features are maximally *aligned*,

In the unified embedded space, common events of the video are organized into closely spaced clusters, and the unusual events are pulled far away from all common events. In addition the projections of the features in this space can be used to directly predict the location of any new video clip in this embedding space, and therefore leads to a classification by nearest neighbor rule.

In the following sections, we will describe the representation of the video in section

2, and the main computational algorithm in section 3. The result of this algorithm on the hospital video sequences are shown in section 4. Finally, we conclude in section 5.

2 Feature Extraction and Video Representation

2.1 Feature for Image Frame

Because we need to process the surveillance video whose length is about days, we need to convert the frames of the video into features so we can process them faster and easier. Before we do the feature extraction, we first drop all the video frames with no foreground objects. In our case, since the surveillance cameras are indoor, the lighting conditions seldom change through the day, so we can easily find the background image. Then we compute difference between the video frames and the background image, if the difference is below a threshold, we think there is no foreground objects in that frame and we drop that frame. With this step, the surveillance video can be reduced to about 1/5 of its original size. For the surviving video frames, we convert them into binary mask images. Each pixel in the binary mask images is 1 if it is a foreground pixel, 0 otherwise like shown in Figure(1)(b). Then we divided each mask image into 5 by 5 tiles and count the number of foreground pixels in each tile. Actually it is a 2D histogram of the foreground pixels in the image like shown in Figure(1)(c) and we call this the motion histogram of the foreground object. Besides this histogram, we also count the histogram of the color of all foreground pixels in the image. These two histograms are our features for image frame.

Now each image frame is represented by its motion and color histogram of the foreground object which can be thought as a feature vector. This feature representation is still too large for analyzing a day's video. To further reduced the size of data, we sample the image frames by keep 1 frame for every 10 frames. Note that normally a semantic meaningful events always lasts for about a few seconds, so this sampling on a 30fps video won't discard the important information of events in the video. However, these feature vectors are also highly redundant, only a tiny fraction of all the possible feature vectors will occur in real life video. To utilize this fact, we can further "compress" the feature vectors using the Vector Quantization(VQ) approach which is popular in speech recognition and textons in image analysis.[5] For example, if we compress all the possible feature vectors into K vectors, then each feature vector (or video frame) can be represented by a single integer label of $[1, K]$.

After all these steps, a n frame video is compressed to about $n/(5 * 10)$ integer labels and a $K * x$ feature vector codebook where x is the dimension of a feature vector. In this paper, we call the K feature vectors in the codebook generated by VQ the prototype features. Every feature vector of a video frame can be represented by one of these prototype features.

Note, the motion prototypes by themselves have no particular meaning. It only serves the purpose of compression. Also note, using this discrete quantization, it is possible that two image frames with similar features might have different prototype label. In this case, the two prototypes must be also similar. This problem can be correct, as we should see later, by keep track of the similarity among the prototypes.

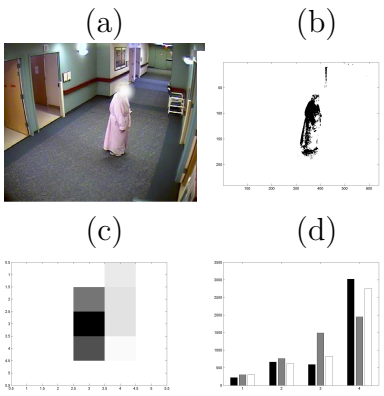

Figure 1: For each image frame, we detect the moving object and compute a coarse description of its motion and quantize it into a motion histogram. We also extract color and texture histogram on the moving object. The color-texture histograms provide rough identifications of human or objects, while the motion histogram provide information on human action.

2.2 Video Representation

The video, which used to be a sequence of image frames, is now a sequence of motion prototype labels. Our video representation is a bag-of-clips. We sliced the compressed video V into fixed length short video clips with overlapping: $(v_1, v_2, ..., v_i, ...)$. For example if the length of the clip is set to 4 seconds. The v_1 will be video frames from 0 second to 4th second, the v_2 will be video frames from 2nd second to 6th second and so on. Noted that each video frame now is a motion prototype label.

We further simplify this representation by removing the time ordering among them. This leads to a bag-of-prototype presentation of a clip. For example, a 4 second clip has 4*30/10=12 video frames in the compressed video. Then there are 12 prototype labels for the clip. Also we don't care about the order of these 12 labels. The advantage of such approach is that it is extremely simple and efficient to build. A disadvantage is that the time ordering of the motion prototypes are discarded. While this could be potentially an problem, in practice, we observe that the in a short clips the ordering of the image frames are quite predictable.

Another issue is how to determine the length of clips. In our experiments we found this is directly linked to what kind of events in the video we want to deal with. For short-term events like people walking around, dropping something on the ground, picking something on the shelf, we found length of 4 seconds works well. For long-term activities, we can increase the clip length to make sure it can capture the characteristics of the activity.

In summary, each video is represented by a bag-of-clips, and each clips is represented by a bag-of-Prototypes derived from its sequence of image frames. The goal is to classify each clips into distinct event categories based on the information in its bag-of-Prototypes.

2.3 Co-occurrence Matrix

An important concept that we will explore is the co-occurrent relationship between *video clips* and motion *prototype features*. We define the *clip-feature* co-occurrence matrix $C \in R^{N \times M}$, such that $C(i, j) = 1$ iff prototype feature i occurs in clip j. Figure (2.3) shows this for the hallway sequence.

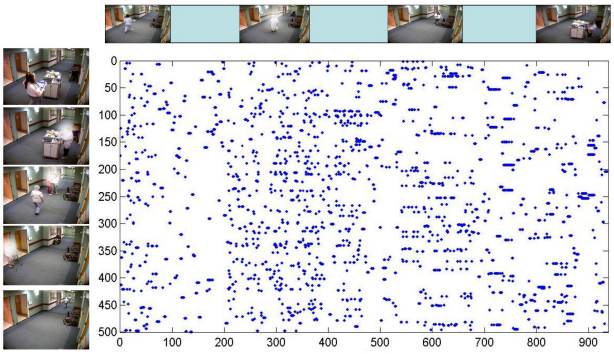

Figure 2: Co-occurrence matrix C, which is a n by m matrix with each row corresponding to a prototype feature and each column corresponding to a video clip.

Note everything occurred in the video is represented in the matrix C. Each column of C, tells us which prototype motion features occurred in a video clip, and each row of C tells us in which video clip each prototype motion feature appears in. However, since the orders of clips and prototype features are unspecified, the C matrix is only defined upto an arbitrary permutation on this columns and rows.

However there is one permutation of the matrix C which can clearly reveal the underlying correlation between the clips and prototypes. In that permutation, all the 1's are packed near the diagonal of the matrix. It is like to re-order the prototype features and clips simultaneously so that the position of a feature in the new feature order is similar to the position of the clip which is closely correlated with the feature in the new clip order.

3 Identifying Activity by Video Summarization

The challenge now is to classify and summarize action events in a video whose length might be on the order of days. The key difficulties, besides the computational one, are that 1) there will be a vast number of action event types we need to identify, and 2) within each action event type will be large variations depending on how who is doing it, and the context under which the task is carried out.

Furthermore, we are interested in detecting unusual events, which by definition don't occur often. To detect such events, we need to have a *complete* characterization of the "normal" action events. But for different scenarios, the definition of "normal" actions may vary greatly, so does the definition of "unusual". In this setting, traditional model based approach, where one trains classifiers to detect a pre-specified list of of action events, is not a viable option. Instead, we will propose a complete data-driven unsupervised approach that will learn from the large amount of video itself, what are

complete list the action events, how we can identify them, and how are action events related to each.

Given a long video, there are far fewer number of distinct action events than number of video clips. We can think the clips are grouped into distinct clusters representing events in the video. Clusters that are isolated from the rest of them are considered to be unusual. To cluster the clips, we need to measure the similarity between any two clips. It is usually done by checking if two clips have a large common set of prototype features occur in them.

This is typically done by computing the inner product of two column vectors corresponding to the two clips in the co-occurrence matrix C. The inner product tells us how many common prototype features they have. However, this similarity criteria has an obvious drawback. Take the following case, suppose we have two prototype features F_1, F_2 that are similar to each other, and F_1 happens in in clip V_1 but not clip V_2, and F_2 happens in clip V_2 but not V_1. We would like to think that, even V_1 and V_2 don't have common prototype features for F_1, F_2, they still share something similar. If we only consider the inner product, this similarity information will be lost. Therefore, it is necessary to use the addition knowledge of the similarity between features in computing the similarity between clips. Recall that the prototype features are histograms, so we can compute the prior similarity between the features using χ^2 distance on the histograms.

An opposite problem can also occur: maybe two prototype features F_1 and F_2 are quite different from each other according to the prior but they always happen in same clips or similar clips and seldom happen in different clips. Then we would like to think that these two features are actually similar to each other in the given video sequence. Now this becomes a *chicken-and-egg* problem: the similarity of clips is determined by the similarity of prototype features, while the similarity of prototype features are affected by the similarity of clips (also the co-occurrence information).

Our solution to this problem is to find a unified embedding space in which the projections of prototype features and clips are near each other if they are co-occurring, and far away otherwise. Furthermore, features that are similar based on histogram comparison should also be mapped closer to each other. One important property of this embedding space is that we can infer the similarity between clip-clip, and feature-feature directly from their coordinates in this unified embedding space. Furthermore, for a new clip, we can predict its embedding location of by simply averaging the locations of its co-occurring features, and classify this clip by seeing which cluster it falls onto.

3.1 Computing unified embedding for features and clips

To quantify constraints on the embedding of features and clips, we will define a weighted graph $G = (V, E)$. We take the clips *and* prototype features as nodes, $V = \{V_1, ..., V_m\} \cup \{F_1, ..., F_K\}$. There are two types of edges in this graph. Type 1 edges define the co-occurring relationship between clip nodes and feature nodes, and Type 2 edges define the similarity between the feature node based on their histogram comparison. $E = \{(V_i, F_j)|C(j, i) \neq 0\} \cup \{(F_i, F_j)\}$. Together the edge weight matrix is defined as:

$$W = \begin{bmatrix} I & C^T \\ C & \beta S_f \end{bmatrix}, \tag{1}$$

where $D, W \in R^{|V|\times|V|}$,and D is a diagonal matrix with $D(i,i) = \sum_j W(i,j)$, and S_f is a pairwise similarity matrix between the features computed by histogram comparison, and β is a weighting factor.

The problem is now finding a placement(embedding) vector $x \in R^{|V|\times k}$ for each node, in a k dimensional space. As we place the nodes in this low dimensional space, one can imagine the graph edges as springs that act to pull together(or apart) clips and features, so that the co-occurring nodes are *aligned*, as shown in figure 3(b).

There are many possible ways we can embed a graph in a lower dimensional space. Some of the techniques include graph drawing[1, 9, 10]. They differ from each other in the criteria being minimized and computation efficiency and accuracy. Here, we adopt a spectral graph method which has the an intuitive energy function:

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

where σ_x is the standard deviation of vector x.

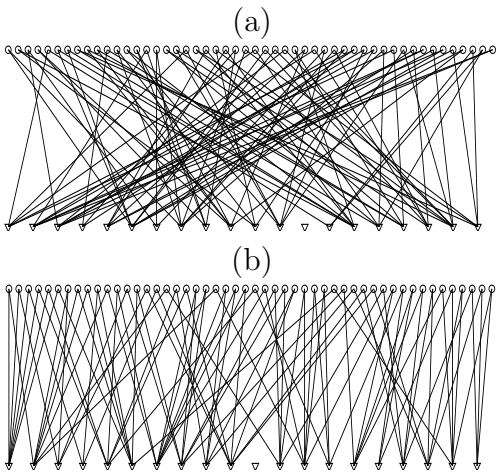

Figure 3: (a) The co-occurrence relationship between video clips(in the top row) and prototype features(in the lower row) define graph edges. One can also imagine these graph edges as springs that pull together(and apart) clips and features. (b) The same graph after the nodes are repositioned along the 1D line so to minimize the overall spring tension. Note that the video clips and prototype featuress are also appear to be in direct correspondence with each other.

To minimize $E(x)$, we need to introduce a node-edge incident matrix A. Define $A \in R^{|V|\times|E|}$: $A(i,k) = \pm W(i,j)^{\frac{1}{2}}$ if node i is incident on edge $k \in E = (i,j)$. $A(i,k)$ takes on the positive value iff if $i < j$. It is easy to verify we can then re-write the energy equation (2) as:

$$E(x) = \frac{\sum_{k\in E} W(i,j)(x(i) - x(j))^2}{\sigma_x} = \frac{x^t A A^t x}{\sigma_x}. \tag{3}$$

Expanding out the term AA^t, we have $(AA^T)(i,i) = \sum_k A(i,k)^2 = d(i)$, where $d(i) = D(i,i)$ is the the degree of node i in G; $(AA^T)(i,j) = \sum_k A(i,k)A(j,k) = -W(i,j)$ iff node i and j are connected in graph G. Therefore,

$$AA^t = D - W. \tag{4}$$

Let's now expand out the term σ_x as:

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

$P(i)$ is an estimate of the prior occurrence likelihood of each story or action event. To estimate it, we count the co-occurrence of each story or action events: $P(i) = \frac{\sum_{k \in E, A(i,k)}}{\sum_{i \in V, k \in E} A(i,k)}$. We can verify that $P(i) = \alpha D(i,i)$, where $\alpha = 1/\sum_{i \in V, k \in E} A(i,k)$. Putting all these together, we can rewrite equation (2) as

$$E(x,y) = \alpha \frac{x^t(D-W)x}{x^t D x - x^t D 1}. \tag{6}$$

This energy is minimized by computing the second smallest eigenvectors of

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

as shown in [19]. Note that the first m elements of vector x contain the coordinate of the clips nodes, and the next K elements contain the coordinate of the prototype features.

3.2 (Un)Usual event categorization

With this unified feature-clip embedding, we have a direct method for evaluate *inferred similarity* among the clips and features: $S(i,j) = e^{-||x(i)-x(j)||_2}$, where i, j, could be two clips or two features. Once the the clips are mapped into this unified embedding space, it is also relatively easy to identify clusters of the clips. One can run K-mean algorithm on the coordinate of the clips to group them into K distinct clusters. The grouping of clips provides a categorization of the action events, one action event for one cluster.

In this setting, detecting unusual events are also relatively easy. We define cluster that is more isolated as unusual events.

To test this concept, we staged a controlled activity video in our lab. During the 10 minute long video, the person(A) is asked to walks to the shelf which is located on the right side of the image, picks up something then walks away. He was also asked to "accidentally" drop and pick up a box he was carrying 3 times. Figure(4), shows the result of unified feature-clip embedding using first 3 eigenvectors. We see the clips are organized into 4 clusters. Two of the larger clusters, which are also connected to each other, consists of the activity of walking around(figure 4(a)), and the activity of picking things from the bookshelf(figure 4(b)). We also see two other smaller cluster that embedded far away from the rest of clips. One of them shown in figure 4(c) consists of all clips from the 3 "droping/picking up" instance. We are surprised to discover the fourth cluster shown in figure 4(d), in fact picking the activity of the person squats down to sort something on the bottom of the shelf. This is not part of our script, but in the context of what else he is doing, this action is quite usual.

3.3 On-line classification with unified embedding

The unified feature-clip embedding not only provides a way to organize clips into action categories, but also a direct method for classifying new clips on-line. This direct event

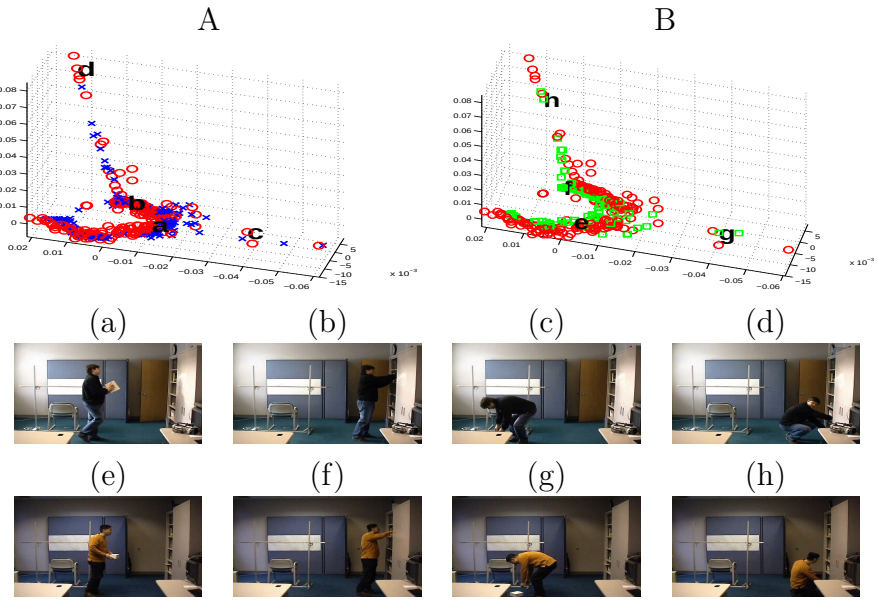

Figure 4: Two staged video activity sequences. During the 10 minute long video, the person is asked to walks to the book shelf to pick up something and walk away. He is also instructed to *accidentally* drop a box and pick it up. (a)-(d) shows distinct events discovered, and their locations in the embedded space (A). The feature description and event category learning from the first sequence is used to classify the second sequence on-line. (e)-(f) are the 4 types of corresponding events detected. (B): The red dots are clips of the 1st day's video, the green squares are clips of the 2nd day's video computed using the on-line method.

classifier allows us to detect events on-line. This property is very important if we want to detect events like someone is drowning in swimming pool or someone is trying break into a room because if we can only detect these events after processing the whole day's video, that will be too late.

The on-line classification consists of three simple steps. First, given a new video clip, we compute its feature occurrence with the existing set of prototype features: checking which prototype features occurred in that clip. Let's call this feature occurrence vector $C^n \in R^{1\times K}$. Then we predict its embedding coordinate based on the embedding coordinates of its features. This is done by adding the new video clip into the existing graph, and approximate the NCut eigenvectors for the expanded weight matrix:

$$W' = \begin{bmatrix} W & (0_m, C^n)^T \\ (0_m, C^n) & 1 \end{bmatrix}, \tag{8}$$

where 0_m is the a vector of 0s of dimension m, m is the number of clips nodes in W. Let (V, S) be the NCut eigenvector/value of W, we can show that the new clip will have the NCut eigenvector value $v^n = (0_m, C^n) * V * diag(S^{-1})$. This amounts to average the embedding coordinates of its co-occurring features. Once we have projected the new clip into the embedding space, we classify it by a simple K-nearest neighbor(KNN) rule.

To test this idea, we labelled the clips shown in figure 4 that belong to the two unusual events of "squats down" and "dropping/picking". On the second day, we shot a video of another person(B) who does the similar things in the lab. Note from figure 4(e) that the lighting conditions, camera parameters and the backgrounds are all different from the video shot on the first day. We compute the image features with the background extracted from the 2nd day's video, but the same motion prototypes as that of the first day's video. Then we can re-construct the coordinates of the clips in the 2nd day's video. The results are shown in Figure 4(B). The result is shown that all two squatting events and picking events are correctly detected with the $k = 3$ in KNN.

3.4 Overall procedure

To summarize, our algorithm consists of the following steps:

1. Process the video to remove the motionless frames and sample the image with a ratio of 1/10.
2. For each frame, k, detect moving object, and extract motion and color/texture histogram: $H(k)$. Apply vector quantization to produce 500 prototype features : $F = \{f_1, ..., f_n\}$ and label all the frames.
3. Slice the video into 4 seconds long video clips: $V = \{v_1, ..., v_m\}$
4. Compute the co-occurrence matrix $C(i, j)$ between each clip v_j and prototype feature f_i
5. Compute pair-wise similarity between the prototype features using chi-eq difference: $S(i1, i2) = \chi^2(H(i1), H(i2))$
6. Construct graph $G = (V, E)$ with associated weight matrix $W = \begin{bmatrix} I & C^T \\ C & \beta S_f \end{bmatrix}$.

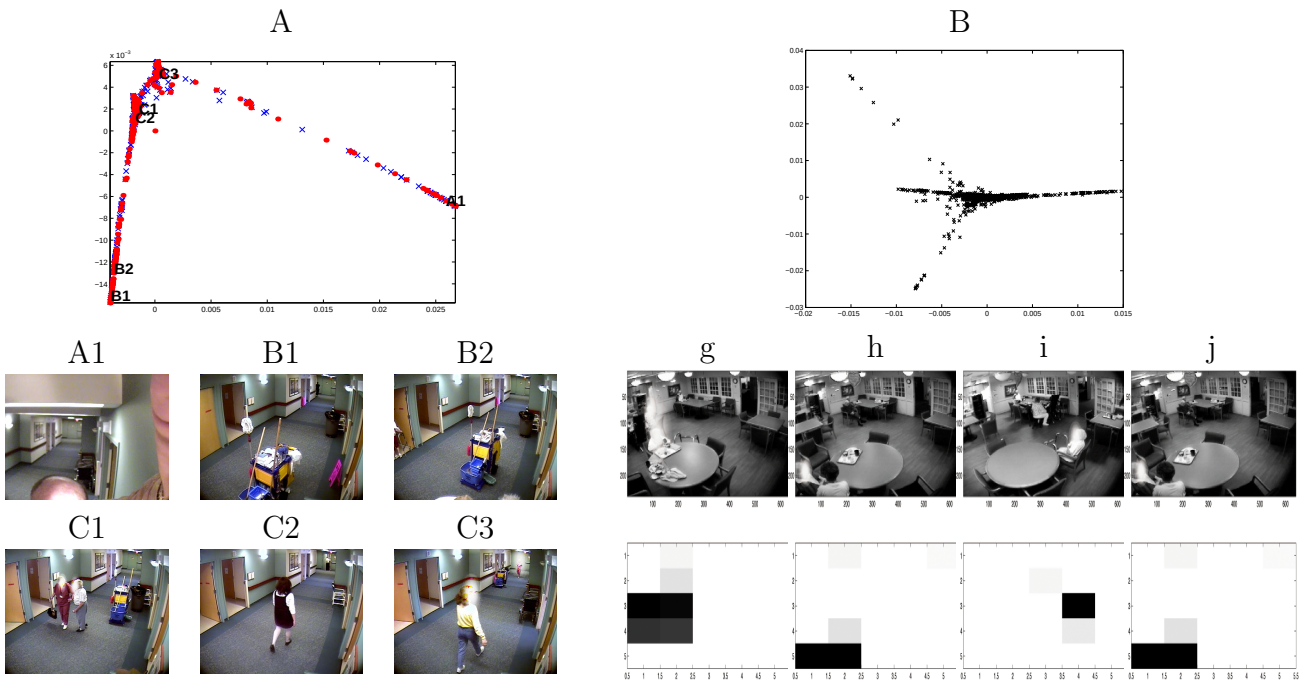

Figure 5: The hospital video. **A**: Hall way sequence clip-feature embedding. The video clips are organized into 3 groups. The *interesting* events found are someone playing around the camera (A1), and a nurse doing house cleaning (B1-B2). **B**: Dinning Hall sequence clip embedding. Four unusual activities being discovered, corresponding to four clusters in the embedding space: (g) one lady eats alone at the near table, (h) a man on wheel chair goes in and out of the room, (i) a patient shaking , (j) a nurse feeding a patient one-on-one.

7. Solve for the smallest eigenvectors of $(D - W)x = \lambda Dx$,
8. The first m rows of the eigenvectors are the coordinates for clips in the embedding space, and the following n rows are the coordinates for prototype features in the embedding space.
9. With the coordinates we can find the cluster of common and unusual activities
10. Given a new clip, we first label each frame of it by the prototype features.
11. Average the coordinates of the prototype features happen in it to get its coordinate in the embedding space.
12. Use KNN to classify this new video clip.

4 Experiments

Now we have validated our algorithm in a more controlled setting, we turn our attention to the unrestricted surveillance of the nursing home where the video length is much longer, and contain much greater number of actions.

Figure(5) shows our result on the hospital video. In the hallway sequence, our algorithm detects the walking activities as the main event, not surprisingly. But it also detects activities of nursing cleaning the rooms, and someone playing around the video

camera. While none of them are unusual in the absolute sense, in the context of people mostly walking through the hall way, they are very different. In the dinning room sequence, we see greater number of activities being detected. Using just 2 embedding dimension, we detected the unusual activities of a) a patient keep rubbing herself as to keep warm, b) a patient eating alone for a long time, c) a man in wheel chair keep coming in and out of the room while everyone else is eating, and d) a nursing feeding the patient one-on-one. Some of these are quite unusual, and could provide useful feedback to the nursing staff.

We should also point out that upon the review of the sorted hospital video, we have found interesting activities that have not been picked out by our algorithm. For example, in the hallway sequence, one of the patient keep pushing the clearing bin down the hallway while the nurse is cleaning inside a room; in the dinning sequence, a patient walked to another patient, and repeatedly kicked(lightly) the other person. To pick out the second example, we probably need to have greater spatial body tracking resolution. The first example, however, presents a greater challenge, as it requires us to reason events at a large time window. Predicting the exact time window of the events is a difficult problem, and it is similar to the problem of predicting the right image window for texture analysis.

5 Conclusion

We proposed an algorithm that automatically detects and summarizes activities in a video. It takes advantage of the large video database itself to identify *interesting activities* and organizes the video into an ordered set of activities. We have demonstrated our algorithm both on a controlled video sequence with ground truth and and several general surveillance video sequences. Our algorithm uses extremely simple image features, and we found the computation efficient and effective. It is well suited for detecting events in a long surveillance video, both in a on-line realtime as well as off-line batch processing mode.

6 Acknowledgment

This work is supported by DARPA HumanID: ONR N00014-00-1-0915, NSFCareMedia: IIS-0105219, ARDA VACE: MDA908-00-C-0037.

References

[1] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, "Graph Drawing: Algorithms for the Visualization of Graphs," Prentice-Hall, 1999.

[2] A. F. Bobick and J. W. Davis. "An appearance-based representation of action." In 13th *International Conference on Pattern Recognition,* Vienna, Austria, August 1996.

[3] Matthew Brand, Nuria Oliver, and Alex Pentland. "Coupled hidden markov models for complex action recognition." In *Proceedings of IEEE CVPR97*, 1997.

[4] C. Bregler. "Learning and Recognizing Human Dynamics in Video Sequences." In *CVPR '97*, pp. 568-574, 1997.

[5] C. Bregler. "Tracking people with twists and exponential maps." In *CVPR98,* 1998.

[6] Davis, J. and Bobick, A. "The representation and recognition of action using temporal templates." *Proceedings Computer Vision and Pattern Recognition* (CVPR'97). pp.928-934. 1997.

[7] P. Duygulu, N. de Freitas, K. Barnard and D.A. Forsyth, "Object Recognition as Machine Translation." *Proc. European Conf. Computer Vision,* 2002

[8] Y. Gong and X. Liu, "Generating Optimal Video Summaries," In *IEEE International Conference on Multimedia and Expo,* 2000.

[9] K.M. Hall, "An r-dimensional Quadratic Placement Algorithm." *Management Science*, 17:212-229, 1970.

[10] M. Kaufmann and D. Wagne, "Drawing Graphs: methods and Models", LNCS 2025, Springer Verlag, 2001.

[11] S. Hongeng and R. Nevatia. "Multi-agent event recognition," In *IEEE Proceedings of the International Conference on Computer Vision*, 2001.

[12] H. Ismai and D. Harwood and L. Davis, "W^4: Who? When? Where? What? A Real Time System for Detecting and Tracking People", In *Third International Conference on Automatic Face and Gesture Recognition*, 1998.

[13] N. Johnson and D.C. Hogg, "Learning the Distribution of Object Trajectories for Event Recognition", In *Image and Vision Computing*, 1995.

[14] G. Medioni, R. Nevatia and I. Cohen. "Event Detection and Analysis from Video Streams," DARPA98, 1998, pp63-72.

[15] Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes III. "Exploiting human actions and object context for recognition tasks." In *IEEE International Conference on Computer Vision, volume 1,* pages 80-86, Corfu, Greece, September 1999

[16] Moore, Essa, "Recognizing Multitasked Activities using Stochastic Context-Free Grammar." In *Proceedings of Workshop on Models versus Exemplars in Computer Vision*, held in Conjunction with IEEE CVPR 2001, Kauai, Hawaii, 2001.

[17] Milind R. Naphade and Thomas S. Huang, "A Probabilistic Framework for Semantic Indexing and Retrieval in Video." In *IEEE International Conference on Multimedia and Expo,* 2000

[18] N. Oliver, B. Rosario, and A. Pentland. "Graphical models for recognizing human interactions." In *Proceedings of NIPS98*, Denver, Colorado,USA, November 1998.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–7, June 1997.

[20] C. Stauffer and Eric Grimson, "Learning Patterns of Activity Using Real-Time Tracking", In *PAMI*, 22(8):747-757, 2000.

[21] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. "Pfinder: Real-time tracking of the human body." In *Photonics East,* SPIE, volume 2615, 1995. Bellingham,