From Videos to Verbs: Mining Videos for Events using a Cascade of Dynamical Systems

Pavan K. Turaga, Ashok Veeraraghavan and Rama Chellappa
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
{pturaga,vashok,rama}@umiacs.umd.edu

Abstract

Clustering video sequences in order to infer and extract events from a single video stream is an extremely important problem and has significant potential in video indexing, surveillance, activity discovery and event recognition. Clustering a video sequence into events requires one to simultaneously recognize event boundaries (event consistent subsequences) and cluster these event subsequences. In order to do this, we build a generative model for events (in video) using a cascade of dynamical systems and show that this model is able to capture and represent a diverse class of events. We then derive algorithms to learn the model parameters from a video stream and also show how a single video sequence may be clustered into different clusters where each cluster represents an event. We also propose a novel technique to build affine, view, rate invariance of the activity into the distance metric for clustering. Experiments are shown both for far field and near field activity videos. The clusters found by the algorithm are shown to correspond to semantically meaningful events in both scenarios.

1. Introduction

Classical approaches to event analysis and recognition are based on building a parametric or non-parametric model for a restricted set of pre-defined events [1],[2]. This approach involves an extensive and often expensive training phase where the model for each event is learnt from several training examples of the event. However, in real world applications, one is not provided with an exhaustive set of the events that may occur in a given setting. Moreover, this approach also requires the application to be retrained for every new deployment so that it is 'aware' of all the events that are expected to occur within its field of view. These limitations have led researchers to look for unsupervised methods for video mining. But, such unsupervised approaches to mining events from videos present a few interesting challenges. Firstly, since the system has no prior knowledge about the nature of the various events, the model for an event must be rich enough to support a wide variety of events that might occur. For example, the typical events that occur in a parking lot are very different from those that occur in an airport and the model must be rich enough to accommodate both scenarios. Secondly, the unsupervised approach has to recognize event boundaries and cluster consistent event subsequences into a single cluster. We show that a 'cascade of dynamical systems' is a very rich model for events and this model also provides a natural way of extracting event consistent subsequences from a long video.

Prior Work: There has been significant interest in recent years to classify videos according to the patterns of activity in them. Such activity-based analysis has primarily focused on supervised approaches which learn a predefined class of activities from a training set such as HMMs [3]. [4] computes the mean 3D structure tensor at each pixel and uses this for automatically discovering motion patterns in an intersection. Most earlier unsupervised approaches to video summarization dealt with problems such as shot boundary detection and scene classification [5]. [6] produces natural language descriptions of video by attaching a semantic concept to the features extracted. Approaches such as [7], [8] attempt to model the problem of learning activity patterns as one of clustering. Most such approaches cast it in the framework of a time-series clustering problem by breaking down the video stream into overlapping subsequences using 'sliding-windows', and then cluster these subsequences. Subsequence based approaches do not account for the fact that activities may have different temporal spans. [9] presents a technique for segmenting motion primitives. In our method, as we first find the temporal span of each activity via segmentation, the model is able to account for activities with vastly varying temporal spans. Then, we model an activity as a sequence of linear time invariant (LTI) dynamical models. Finally, we show how to build certain invariances into the clustering scheme.

Contributions: Our contributions in this paper are two-fold. First, we demonstrate how a cascade of LTI systems can be used to model a wide variety of activity patterns in several domains and how to learn the models and cluster them in a completely unsupervised manner. Then, we show a novel way of building three types of invariances into the distance metric used for clustering - affine invariance, a limited version of view invariance and invariance to execution rate of activities.

Outline: Section 2 describes the features used and the cascade of LTI systems for an event in detail. Section 3 discusses the details of segmenting a video into action elements and learning the model parameters for each segment. In section 4 we discuss a novel approach to build invariances. Section 5 is a short discussion about some of the properties of the model and provides experimental results.

2. Modeling Events in Videos

An event consists of an actor (subject) executing a series of action elements (verbs) in order to achieve a certain goal. For example, a man driving a car into a parking lot, parking the car, alighting from it, walking out of the parking lot (series of action elements-verbs) contributes to a typical event. Any model for events must be able to represent each of the verbs (action elements) separately while simultaneously being able to detect the boundaries between them.

2.1. Modeling Action Elements

A complex event can be broken down into its constituent action elements. During each action element, the motion of the actor remains consistent. In fact, it is this consistency of motion that segments an event into action elements. Therefore, each action element is modeled using a time invariant dynamical system and the event is modeled as a cascade of dynamical systems.

2.1.1 Representation Feature

Since we are interested in mining events from video sequences, the feature extracted from each image frame must capture the motion of objects in that frame. Generally, optical flow serves as a good approximation to the true motion fields. Therefore, we use optical-flow as the feature for representing motion in each image. We use the algorithm developed in [10] for computing the flow. In cases where optical flow computation is not robust, one can use background subtracted masks or even silhouettes as the feature. The appropriate feature to use depends on the end application. In the ensuing discussions, we will assume that feature to be optical flow, but it can also be interpreted using any other appropriate feature.

2.1.2 Linear Dynamical System for Action Elements

As already discussed, the dynamics of each action element can be modeled using a time-invariant dynamical system. In several scenarios (like far-field surveillance, objects moving on a plane etc), it is reasonable to model constant motion in the real world using an LTI model on the image plane. Given the boundaries between action elements, we model each of these segments using an LTI model. Lets assume that the $P+1$ consecutive frames $s_k, ..., s_{k+P}$ belong to the k^{th} segment and let $f(i)$ denote the observations (flow/silhouette etc) from that frame. Then, the dynamics during this segment can be represented as

$$f(t) = Cz(t) + w(t) \quad w(t) \sim N(0, R) \tag{1}$$
$$z(t+1) = Az(t) + v(t) \quad v(t) \sim N(0, Q) \tag{2}$$

z is the hidden state vector, A the transition matrix and C the measurement matrix. w and v are noise components modeled as normal with 0 mean and covariance R and Q respectively. When flow is used as the feature, we can write similar equations for the x and y components independently. We assume independence of flow components for simplicity and to reduce the dimensionality of the estimation problem. Similar models have been successfully applied in several tasks like dynamic texture synthesis and analysis [11], comparing silhouette sequences [12], [13] etc. But we differ from these as we do not assume that we know the temporal span of the segments. We explicitly deal with the temporal segmentation problem in section 3.1. In summary, the parametric model for each segment consists of the measurement matrix C and the transition matrix A.

2.2. Sequence of Dynamical Systems

An event is composed of a series of action elements. We have modeled each action element using an LTI system. The event model is now composed of a linear cascade or a sequence of such dynamical systems. In reality, most events have a very specific temporal order for the execution of action elements. For example, if our goal is to get to the office, then the sequence of actions executed might be - drive into parking lot, park car, alight from car, walk away from the parking lot. Inherently, there is a sequential ordering of the actions in the event. Therefore, we model an event as a cascade of action elements with each action element modeled as an LTI system. Figure 1 illustrates the complete model for such an event (using flow as the feature).

Switching between Dynamical Systems: In order to completely specify the model we also need to specify the switching times between these dynamical systems or equivalently, the amount of time (or frames) spent executing an action element i.e. the *dwell* time. We considered modeling the event as a Markov model, in which case the probability

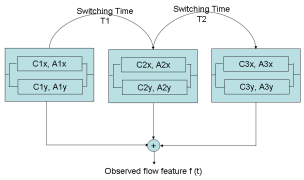


Figure 1. Illustration of a cascade of three linear dynamical systems. The temporal order of the execution of these dynamical models and their switching times are shown with arrows.

distribution of the dwell time turns out to be an exponential distribution whose mode is at 0. But, physically the amount of time spent doing one particular action takes a finite amount of time. Thus, to model the dwell time, we need a continuous distribution over time that satisfies the following requirements - a) Support set which is the entire non-negative real line, b) Non-zero mode. The Gamma distribution satisfies both the above requirements. Simpler choices such as Gaussian, exponential, double exponential violate one or the other requirement. Thus, we model the dwell time for each action element as a Gamma distribution with parameters α_k and β_k with $\alpha_k > 1$ (this constraint ensures a non-zero mode). The Poisson distribution also shares the above properties except that it is a discrete distribution.

3. Learning the Event Model

We have modeled an event as a cascade of dynamical systems. But given a video sequence, we first need to segment the video into action elements and discover the relationship between them. The challenge is to accomplish all of this in a completely unsupervised manner while being invariant to variabilities in an event like execution rate, resolution of video, rotation and translation etc. We will now describe an algorithm to automatically segment the video and learn the model parameters in an unsupervised manner.

3.1. Discovering Action Elements and their boundaries

During each action segment, the motion of features is modeled using an affine motion model as is usually the case with traditional tracking algorithms. The crucial difference is that, we do not actually segment and track individual objects in the scene, but instead model the entire feature during a segment using the affine motion model. For the first few (about 3 or 4) set of frames after the beginning of a new segment, we cumulatively learn a single set of affine parameters for the change in the feature. For every incoming new frame, we evaluate whether it is consistent with the predictions of the learnt affine parameters. If so, we add the frame to the current segment. Otherwise, we detect the presence of a boundary. Learning the affine parameters for each segment can be achieved in closed-form using the properties of the fourier-transform (FFT) [14].

Figure 2. Detected Segment Boundaries for (a)Squatting and (b) Bending

This segmentation scheme is suboptimal due to the assumption of affine motion. To overcome this we iterate back and forth between learning the LTI model for each segment and tweaking the segment boundaries till convergence is reached. Taking the output of the above scheme as an initial point, we learn the LTI model for each segment. Without loss of generality, let $S_1 = (A_1, C_1)$ and $S_2 = (A_2, C_2)$ be two adjacent segments and their corresponding LTI models. Suppose the temporal span of S_1 is $[t_1, t_b)$ and that of S_2 is $[t_b, t_2]$. Here t_b denotes the boundary between the segments. As will be described in section 3.2, columns of C_k correspond to the top d principal components (PCs) of the observations in segment k. To *evaluate* the boundary according to the learnt models, we compute the reconstruction error of all the observations according to the PCs in the corresponding segments. We move the boundary by an amount τ in forward and backward directions and choose the one that minimizes this error. Thus, we search for the minima of the following cost functional:

$$\Delta(\tau) = \sum_{t=t_1}^{t_b+\tau} \left\| C_1(C_1^T f_t) - f_t \right\|^2 + \sum_{t=t_b+\tau}^{t_2} \left\| C_2(C_2^T f_t) - f_t \right\|^2 \quad (3)$$

f_t is the observation at time t and $\tau \in [-T, T]$. In our experiments we typically chose T to be 10. The new boundary is found as $t_b^{new} = t_b^{old} + \arg\min_\tau \Delta(\tau)$. With the new boundary the models are learnt again, and the process is repeated till convergence, i.e. the boundary does not change anymore. We show some segmentation results on a near-field video sequence of an actor performing 5 different activities. Each activity is repeated several times at random. Sample segment boundaries for two activities are shown in figure 2. Note that the segmentation algorithm is independent of the rate of execution of the activity. The video sequence was consistently segmented at the same pose in several instances of the same activity. (More detailed results are presented in the supplemental material.)

Relation with Switching Linear Dynamical Systems: Learning the switching instants between LTI models is also encountered in the area of Switching Linear Dynamical Systems (SLDS). In SLDS, usually an extra hidden state is used to model switches. Any change in this hidden state corresponds to a switch between the LTI models such as in [15] and [16]. Usually, the number of states to switch amongst is assumed to be known, but we do not make any such assumption. An approach was presented in [17] for a special class of systems to estimate the number of states as well as to learn the dynamics of each system. In our experiments, we found that our algorithm for segmentation works reasonably well with a far smaller computational burden.

3.2. Learning the LTI Models for each segment

As described earlier, each segment is modeled as an LTI system. We use tools from system identification to estimate the model parameters for each segment. This estimate can be obtained in closed form. The algorithm is described in [18] and adopted for texture modeling in [11]. Let observations $f(1), f(2), \ldots f(\tau)$, represent the features for the frames $1, 2, ...\tau$. Let $[f(1), f(2), \ldots f(\tau)] = U\Sigma V^T$ be the singular value decomposition of the data. Then $\hat{C} = U, \hat{A} = \Sigma V^T D_1 V (V^T D_2 V)^{-1} \Sigma^{-1}$, where D_1 = [0 0;$I_{\tau-1}$ 0] and D_2 = [$I_{\tau-1}$ 0;0 0]. These estimates of C and A constitute the model parameters for each action segment. For the case of flow, the same estimation procedure is repeated for the x and y-components of the flow separately.

3.3. Clustering Action Element Prototypes

We have now segmented a long video sequence into several distinct segments and learnt the model parameters $(\hat{C}, \hat{A})$ for each of these segments. Even though a long video might consist of several segments, not all of them will be distinct. We need to cluster these segments to discover the distinct action elements. In order to perform this clustering, we need a distance measure on the space of LTI models. We use subspace angles $(\theta_i, i = 1, 2, ...n)$ between two ARMA models as defined in [19].

Using these subspace angles $\theta_i, i = 1, 2, ...n$, three distances, Martin distance (d_M), gap distance (d_g) and Frobenius distance (d_F) between the ARMA models are defined as follows:

$$d_M^2 = \ln \prod_{i=1}^{n} \frac{1}{\cos^2(\theta_i)}, \quad d_g = \sin \theta_{max}, \quad d_F^2 = 2 \sum_{i=1}^{n} \sin^2 \theta_i \tag{4}$$

We use the Frobenius distance in all the results shown in this paper. Suppose we have N segments in the video sequence, then we create an $N \times N$ matrix D whose $(i, j)^{th}$ element contains the distance between the models of segment i and segment j.

Clustering the Segments: Standard clustering techniques like the k-Nearest Neighbors suffer from the limitation that the number of cluster centers need to be known ahead of time. But, in the current setting, we do not know the number of clusters k. A similar problem has been tackled in multibody motion segmentation [20], by rearranging the distance matrix D using row and column permutations such that the most similar segments are clustered together while the dissimilar segments are separated. We refer the interested reader to [20] for details. Let the K cluster centers thus obtained be given by $C_1, C_2, C_3, \ldots C_K$. The segmented video is then given by a sequence of these labels.

3.4. Discovering prototype relationships using co-occurrence statistics

After clustering the action elements each segment is assigned a label. Suppose we have the following sequence of labels $(C_1, C_3, C_2, C_6, C_7, C_8, C_1, C_3, C_5, C_2, C_6, C_1, C_7, C_8)$. Persistent activities in the video would appear as a repetitive sequence of these labels. From this sequence, we need to find the *approximately* repeating patterns. We say *approximate* because oversegmentation may cause the patterns to be not exactly repetitive. We can say that (C_1, C_3, C_2) and (C_6, C_7, C_8) are the repeating patterns, up to one insertion error. To discover the repeating patterns, we build the n-gram statistics of the segment labels. We start by building a bi-gram, tri-gram and four-gram models. In our experience, oversegmentation of the video is more common than undersegmentation. Thus, we allow for up to one insertion error while building the n-gram statistics. We prune the bi-grams which appear as a subsequence of a tri-gram. We prune the tri-grams in a similar fashion. Finally, we declare the n-grams with a count above a threshold (depending on the length of the video) as the repeating patterns in the video.

3.5. Learning the Cascade structure of Events

The cascade structure of individual events is the exact sequence of the prototypes in the n-grams. Once we have the cascade structure, we can go one step further and build a generative model by learning the statistics of the duration of each action prototype. We model the duration of each action prototype as a Gamma distribution with parameters $\alpha_k > 1$ and β_k. The parameters (α_k, β_k) are learnt directly from the results of segmentation.

4. Building Invariances into the Distance Metric

The distance metrics defined in the previous section will break down when there is a change in viewpoint or there is an affine transformation of the low-level features. We propose a technique to build these invariances into the distance metrics defined above.

4.1. Affine and View Invariance

In our model, under feature level affine transforms or view-point changes, the only change occurs in the measurement equation and not the state equation. As described in section 3.2 the columns of the measurement matrix (C) are the principal components (PCs) of the observations of that segment. Thus, we need to discover the transformation between the corresponding C matrices under an affine/view change. We start by proving a theorem that relates low level

feature transforms to transformation of the principal components.

Theorem 4.1: Let $\{X(\overline{p})\}$ be a zero-mean random field where $\overline{p} \in D_1 \subseteq R^2$. Let $\{\lambda_n^X\}$ and $\{\phi_n^X\}$ be the eigenvalues and corresponding eigenfunctions in the K-L expansion of the covariance function of X. Let $T : D_2 \longrightarrow D_1$, where $D_2 \subseteq R^2$ be a continuous, differentiable one-to-one mapping. Let $\{G(\overline{q})\}, \overline{q} \in D_2$ be a random field derived from X as $G(\overline{q}) = X(T(\overline{q}))$. If the Jacobian of T, denoted by $J_T(\overline{r})$, is such that $det(J_T(\overline{r}))$ is independent of $\overline{r}$, then the eigenvalues and eigenfunctions of G are given by $\lambda_n^G = \frac{\lambda_n^X}{|J_T|^{1/2}}$ and $\phi_n^G(\overline{q}) = \frac{\phi_n^X(T(\overline{q}))}{|J_T|^{1/2}}$.

Proof: Let $K_X(\overline{p}, \overline{s})$ be the covariance function of X. Then by the definition of the K-L expansion the following equations hold.

$$\int_{D_1} K_X(\overline{p}, \overline{s})\phi_n^X(\overline{s})d\overline{s} = \lambda_n^X \phi_n^X(\overline{p}), \quad \int_{D_1} \phi_m^X(\overline{s})\phi_n^X(\overline{s}) = \delta(m, n) \tag{5}$$

where both $\overline{p}, \overline{s} \in D_1$ and $\delta(m, n) = \{1 \text{ if m = n, 0 otherwise}\}$. Now, $\{G(\overline{q})\}$ is related to X as $G(\overline{q}) = X(T(\overline{q}))$. For $\overline{q}, \overline{r} \in D_2$, the covariance function of G is given by $K_G(\overline{q}, \overline{r}) = E[G(\overline{q})G(\overline{r})] = E[X(T(\overline{q}))X(T(\overline{r}))] = K_X(T(\overline{q}), T(\overline{r}))$. Now consider the following equation.

$$\int_{D_2} K_G(\overline{q}, \overline{r})\phi_n^X(T(\overline{r}))d\overline{r} = \int_{D_2} K_X(T(\overline{q}), T(\overline{r}))\phi_n^X(T(\overline{r}))d\overline{r} \tag{6}$$

$$= \int_{D_1} K_X(\overline{p}, \overline{s})\phi_n^X(\overline{s})\frac{1}{|J_T(\overline{r})|}d\overline{s} \tag{7}$$

where (7) is obtained by a change of variables given by $\overline{p} = T(\overline{q}), \overline{s} = T(\overline{r})$, and $|J_T(\overline{r})|$ is the determinant of the Jacobian of T with respect to $\overline{r}$ evaluated at $\overline{r} = T^{-1}(\overline{s})$. Now, if $|J_T(\overline{r})| = |J_T| = constant$, then it comes out of the integral in (7), and using (5) we obtain

$$\int_{D_2} K_G(\overline{q}, \overline{r})\phi_n^X(T(\overline{r}))d\overline{r} = \frac{\lambda_n^X}{|J_T|}\phi_n^X(T(\overline{q})) \tag{8}$$

It can further be shown that the set of functions $\{\frac{\phi_n^X(T(\overline{q}))}{|J_T|^{1/2}}\}$ form an orthonormal set. Thus, we have shown that the eigenvalues and eigenfunctions of G are given by $\{\frac{\lambda_n^X}{|J_T|^{1/2}}\}$ and $\{\frac{\phi_n^X(T(\overline{q}))}{|J_T|^{1/2}}\}$ respectively. The utility of this theorem is that if the low-level features like flow/silhouettes undergo a spatial transformation which satisfies the conditions stated in the theorem, then the corresponding PCs also undergo the same transformation.

Application to Invariances: When two images are related by a general spatial transform (affine, homography etc), they are related by $I_2(x, y) = I_1(T(x, y))$. Consider the set of 2-D affine-transforms given by $T(\overline{p}) = A\overline{p} + \overline{t}$. Writing this in inhomogeneous coordinates $\overline{p} = [x, y]'$

$$T(\overline{p}) = \begin{bmatrix} a_{11}x + a_{12}y + t_1 \\ a_{21}x + a_{22}y + t_2 \end{bmatrix} \tag{9}$$

The Jacobian for the transformation is given by $J_T = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ whose determinant is a constant. Thus, by the above theorem, if a set of observations are affine transformed then their principal components also get transformed by the same affine parameters. Consider now a 2-D plane homography given by $H = [h_{ij}]$. In the inhomogeneous coordinates the transformation is given by

$$T(\overline{p}) = \begin{bmatrix} (h_{11}x + h_{12}y + h_{13})/(h_{31}x + h_{32}y + h_{33}) \\ (h_{21}x + h_{22}y + h_{23})/(h_{31}x + h_{32}y + h_{33}) \end{bmatrix} \tag{10}$$

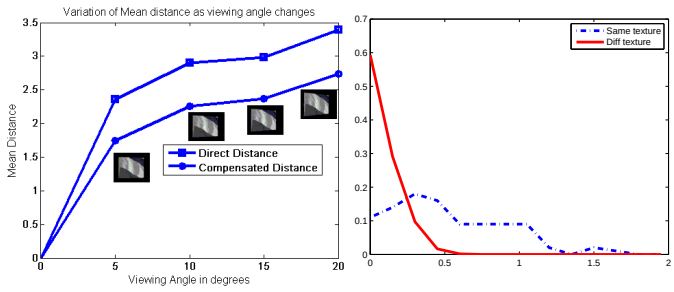
If $h_{31}, h_{32} << h_{33}$ which is often the case in small changes in viewpoint (due to space constraints a rigorous justification for this is provided in the supplemental material), then the Jacobian of the above transformation can be approximated by $J_T = \frac{1}{h_{33}}\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$ whose determinant is also a constant. Thus, the above theorem can be used even in the case where observations are transformed by a homography.

Note: The theorem was proved for continuous random fields. In real images, spatial transforms are not one-to-one maps due to the discrete nature of the underlying lattice. But, our experiments suggest that this theorem can be used to get very good approximations even in the discrete case.

Modified Distance Metric: Proceeding from the above, to match two ARMA models of the same activity related by a spatial transformation, all we need to do is to transform the C matrices (the observation equation). Given two systems $S_1 = (A_1, C_1)$ and $S_2 = (A_2, C_2)$ we modify the distance metric as

$$d_{compensated}(S_1, S_2) = \min_T d(T(S_1), S_2) \tag{11}$$

where $d(.,.)$ is any of the distance metrics in (4), T is the transformation. $T(S_1) = (A_1, T(C_1))$. Columns of $T(C_1)$ are the transformed columns of C_1. The optimal transformation parameters are those that achieve the minimization in (11). Depending on the complexity of the transformation model, one can use featureless image registration techniques such as [14], [21] to arrive at a good initial estimate of T. Computing the gradient of the proposed distance metric is extremely difficult due to the recursive way the subspace angles are defined (section 3.3). We could not arrive at closed form expressions for the gradients. Instead, we resort to using Nelder-Mead's (NM) simplex method to perform the optimization. The NM method is a direct search algorithm that is used when gradients cannot be computed



(a) (b)

Figure 3. (a)Variation of Mean Distance as viewing angle changes. Sample views shown, (b)Histogram of difference between Frobenius and $d_{compensated}$ as seen from different views

or accessed. Even though only limited convergence results for the NM method are known, it is known to work well in practice [22].

To illustrate the effectiveness of our proposed technique to build invariances into the distance metric, we conducted the following experiment. We took a set of 10 dynamic textures from http://www.cwi.nl/projects/dyntex/index.html [23]. We modeled the textures to be lying on a plane in front of the camera perpendicular to the optical axis, and simulated a change in viewing angle from 0° to 20° in increments of 5° by means of a homography (0° corresponds to the frontal view). We took the images as the observations. Figure 3(a) shows how the Frobenius distance breaks-down as the viewing angle is changed. The plot also shows $d_{compensated}$. It can be seen that the proposed technique indeed works better. In figure 3(b), we plot normalized histograms of $(d_F - d_{compensated})$ for same textures as seen from different views and different textures as seen from different views. When comparing different textures, $d_{compensated}$ is not significantly lower than d_F, hence the peak at 0. But, for the same texture as seen from different views, we see that $d_{compensated}$ is significantly lower than d_F.

4.2. Invariance to Execution Rate of Activity

While building models for activities, one also needs to consider the effect of different execution rates of the activity [24]. In this section, we propose a technique to build invariance with respect to execution rate into the distance metric. In the most general case, one needs to consider general warping functions of the form $g(t) = f(w(t))$ such as in [25] where Dynamic time warping (DTW) is used to estimate $w(t)$. DTW requires a distance metric to be defined on the low-level features and hence falls under the class of algorithms that build invariance at the feature level. We propose a technique to build invariances not at the feature level, but at the distance metric level. We consider linear warping functions of the form $w(t) = qt$ for each action segment. Linear functions for each segment give rise to a piece-wise linear warping function for the entire activity, which accounts for variabilities in execution rate well. It can be shown that, under linear warps the stationary distribution of the Markov process in (2) does not change. Hence, a linear warp will affect only the state equation and not the measurement equation i.e. the A matrices and not the C matrices. Consider the state equation of a segment: $X_1(k) = A_1 X_1(k-1) + v(k)$. Ignoring the noise term for now, we can write $X_1(k) = A_1^k X(0)$. Now, consider another sequence that is related to X_1 by $X_2(k) = X_1(w(k)) = X_1(qk)$. In the discrete case, for non-integer q this is to be interpreted as a fractional sampling rate conversion as encountered in several areas of DSP. Then,

$$X_2(k) = X_1(qk) = A_1^{qk} X(0) \tag{12}$$

i.e. the transition matrix for the second system is related to the first by $A_2 = A_1^q$.

Estimating q: Given two transition matrices of the same activity but with different execution rates, we need a technique to estimate the warp factor q. Consider the eigen-decomposition of $A_1 = V_1 D_1 V_1^{-1}$, and $A_2 = V_2 D_2 V_2^{-1}$. Then, for rational q, $A_2 = A_1^q = V_1 D_1^q V_1^{-1}$. Thus, $D_2 = D_1^q$, i.e. if λ is an eigenvalue of A_1, then λ^q is an eigenvalue of A_2 and so forth. Thus, we can get an estimate of q from the eigenvalues of A_1 and A_2 as

$$\hat{q} = \frac{\sum_i log \left|\lambda_2^{(i)}\right|}{\sum_i log \left|\lambda_1^{(i)}\right|} \tag{13}$$

where $\lambda_2^{(i)}$ and $\lambda_1^{(i)}$ are the complex eigenvalues of A_2 and A_1 respectively. Thus, we compensate for different execution rates by computing $\hat{q}$. In the presence of noise, the above estimate of q may not be accurate, and can be taken as an initial guess in an optimization framework similar to the one proposed in section 4.1. But, in our experiments we obtained accurate estimates using the above procedure. Note that compensation for execution rate is done only for segments which have very similar $\hat{C}$ matrices.

5. Discussions and Experiments

Cascade of Dynamical Systems - A Grammar viewpoint: The cascade of dynamical systems can be viewed as a simplistic form of a regular expression grammar - each discovered activity being represented as a set of production rules leading from one state to the other. Note that this simple grammar is learnt in a totally unsupervised fashion. Given a training set of videos, we can segment the activity into prototypes and learn the relationships between them as a problem of grammatical inference instead of enforcing a linear structure. But in the completely unsupervised

Activity Type	Motif 1	Motif 2	Motif 3	Motif 4	Motif 5
Bending	10	1	0	2	1
Squatting	2	8	2	0	0
Throwing	0	0	7	0	1
Pick Phone	3	0	0	9	0
Batting	0	0	0	1	9

Table 1. Composition of the Discovered Clusters

scenario that we are interested in, the linear cascade is a reasonable and intuitive model.

Cascade of Dynamical Systems as a generative model: It is important for event models to possess both the ability to recognize actions and the ability to generate typical action sequences. There are currently very few unsupervised methods for event discovery and these methods typically rely on shot boundary detection, and therefore do not provide a generative model. In contrast, the cascade of linear dynamical systems model can be interpreted as a generative model. To generate an activity, we simply follow the linear cascade of its prototypes. The time spent in executing each individual prototype is sampled from the distribution of its duration.

Experiments: In the experiment described in section 3.1, five different complex activities – throw, bend, squat, bat and pick phone were discovered automatically. We were also able to learn the cascade of dynamical systems model in a completely unsupervised manner. We manually validated the segment boundaries and the corresponding discovered activities. We call each discovered repetitive pattern a *motif*. To counter oversegmentation effects, we merge very similar motifs. Since, a motif is a string of labels, we used the Levenshtein distance [26] as the metric to merge them. The classification of the activities into motifs is tabulated in Table 1. We see that the table has a strong diagonal structure indicating that each of the discovered motifs corresponds to one of the activities in the dataset. Figure 4 shows activity labels for the entire video sequence extracted manually and automatically. Matching of the colors in the figure indicate that the algorithm is able to discover and identify activities in an unsupervised manner. We found that the errors in labeling are typically near the transition between two activities, where the actual labeling of those frames is itself subject to confusion. To visualize the clusters and to see the *trajectories* of each activity, we embedded each segment into a six-dimensional Laplacian eigenspace. Dimensions 1-3 are shown in figure 5 and dimensions 4-6 in figure 6. We see that the trajectories of the same activity are closely clustered together in the Laplacian-space.

We show a few more recognition experiments based on our modified distance metric. In the next experiment, the setup is the same as described above. But, this time we have 10 activities – *Bend, Jog, Push, Squat, Wave, Kick, Batting, Throw, Turn Sideways, Pick Phone*. Each activity is executed at varying rates. For each activity, a model is

Figure 4. Color coded activity labeling for a 4000 frame video sequence. (a) Manual Labeling (b) Unsupervised Clustering result. Image best viewed in color.

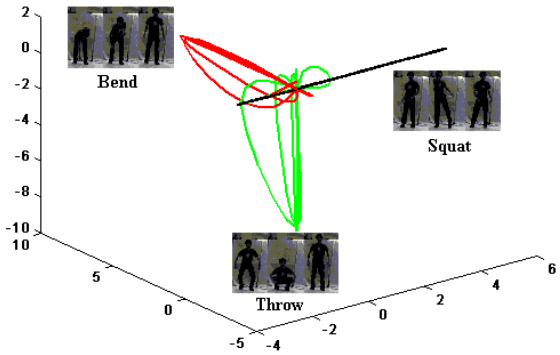


Figure 5. Trajectory Clusters in Laplacian Space dims 1-3. Best viewed in color.

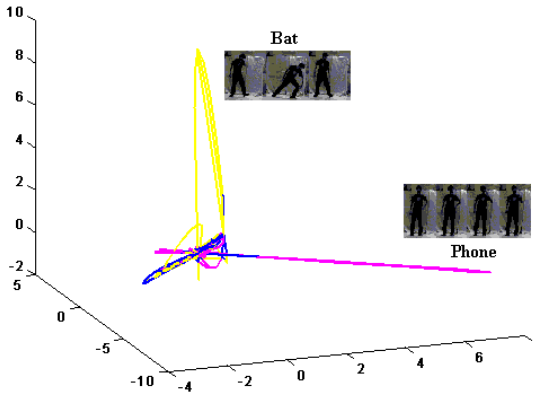


Figure 6. Trajectory Clusters in Laplacian Space dims 4-6. Best viewed in color.

learnt and stored as an exemplar. The features (flow-fields) are then translated and scaled to simulate a camera shift and zoom. Models were built on the new features, and tested using the stored exemplars. We also implemented a heuristic procedure in which affine transforms are compensated for by locating the center of mass of the features and building models around its neighborhood. We call it Center of Mass Heuristic – CMH. Recognition percentages are shown in table 2. The baseline column corresponds to direct application of the Frobenius distance. We see that our method performs better in almost all cases.

We also conducted a recognition experiment on a 10 minute video sequence obtained from a far-field surveillance camera. There were 4 different walking patterns (in

	Baseline		CMH		Our Method	
	Exemplars		Exemplars		Exemplars	
Activity	1	10	1	10	1	10
1	40	0	40	40	40	50
2	0	0	0	10	70	80
3	0	0	20	40	10	20
4	40	30	10	20	30	60
5	30	30	40	20	40	40
6	10	0	40	50	30	50
7	0	10	0	30	30	70
8	0	10	30	40	0	40
9	0	40	20	20	30	70
10	0	0	10	20	40	40
Average	12	12	21	29	32	52

Table 2. Recognition accuracies for three schemes

the location and direction of walk). A model for each of these activities was built and a recognition experiment was run over the entire video sequence and results were manually verified. There were 3 segments that were misclassified from a total of 24 meaningful segments. All of these 3 errors resulted because of a confusion between activities that are co-located but vary only in the local direction of motion.

6. Conclusions

In this paper, we proposed a vocabulary model for dynamic scenes and presented algorithms for unsupervised learning of the vocabulary from long video sequences. We showed the efficacy of the approach using both far-field surveillance and near-field videos. We also presented a technique for incorporating affine and view-invariance into the model. The results are promising and show that our technique can be used for unsupervised activity indexing as an initial filter for further processing.

References

[1] D. M. Gavrila, "The visual analysis of human movement," *CVIU*, 1998. 1

[2] C. Cedras and M. Shah, "Motion based recognition: A survey," *Image and Vision Computing*, 1995. 1

[3] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition." *CVPR*, 1997. 1

[4] J. Wright and R. Pless, "Analysis of persistent motion patterns using the 3d structure tensor," *IEEE Workshop on Motion and Video Computing*, pp. 14–19, 2005. 1

[5] Y. Rui, Z. Xiong, R. Radhakrishnan, A. Divakaran, and T. S. Huang, "A unified framework for video summarization, browsing and retrieval," *MERL Technical Report TR2004-115*, 2004. 1

[6] A. Kojima, T. Tamura, and K. Fukunaga, "Natural language description of human activities from video images based on concept hierarchy of actions," *IJCV*, vol. 50, no. 2, pp. 171–184, 2002. 1

[7] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activity in video," *CVPR*, 2004. 1

[8] L. Zelnik-Manor and M. Irani, "Event-based video analysis," *CVPR*, 2001. 1

[9] D. DelVecchio, R. M. Murray, and P. Perona, "Decomposition of human motion into dynamics based primitives with application to drawing tasks," *Automatica*, vol. 39, pp. 2085–2098, 2003. 1

[10] A. Ogale and Y. Aloimonos, "Shape and the stereo correspondence problem," *IJCV*, vol. 65 no:1, 2005. 2

[11] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic textures," *ICCV*, vol. 2, pp. 439–446, 2001. 2, 4

[12] A. Veeraraghavan, A. K. Roy-Chowdhury, and R. Chellappa, "Matching shape sequences in video with applications in human movement analysis," *IEEE PAMI*, Dec 2005. 2

[13] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto, "Recognition of human gaits," *IEEE CVPR*, vol. 2, pp. 52–57, 2001. 2

[14] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE Transactions on Image Processing*, 1996. 3, 5

[15] B. North, A. Blake, M. Isard, and J. Rittscher, "Learning and classification of complex dynamics," *IEEE PAMI*, vol. 22, no. 9, pp. 1016–1034, 2000. 3

[16] V. Pavlovic and J. Rehg, "Impact of dynamic model learning on classification of human motion," *IEEE CVPR*, pp. 788–795, 2000. 3

[17] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," *In Proc. of IEEE Conference on Decision and Control*, 2003. 3

[18] P. V. Overschee and B. D. Moor, "Subspace algorithms for the stochastic identification problem," *Automatica*, vol. 29, pp. 649–660, 1993. 4

[19] K. D. Cock and B. D. Moor, "Subspace angles and distances between ARMA models," *Proc. of the Intl. Symp. of Math. Theory of networks and systems*, 2000. 4

[20] J. Costeira and T. Kanade, "A multi-body factorization method for independently moving objects," *Technical Report - CMU Robotics Institute*, 1997. 4

[21] S. Mann and R. W. Picard, "Video orbits of the projective group: A simple approach to featureless estimation of parameters," *IEEE Transactions on Image Processing*, vol. 6, no. 9, 1997. 5

[22] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998. 6

[23] D. Chetverikov and R. Pteri, "A brief survey of dynamic texture description and recognition," *Proc. of the International Conference on Computer Recognition Systems*, 2005. 6

[24] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the space of a human action." *ICCV*, pp. 144–149, 2005. 6

[25] A. Veeraraghavan, R. Chellappa, and A. K. Roy-Chowdhury, "The function space of an activity," *IEEE CVPR*, 2006. 6

[26] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Doklady Akademii Nauk SSSR*, vol. 163(4), pp. 845–848, 1965. 7