

Learning 3D Action Models from a few 2D videos for View Invariant Action Recognition

Pradeep Natarajan
Raytheon BBN Technologies
Cambridge, MA
pradeepn@bbn.com

Vivek Kumar Singh
University of Southern California
Los Angeles, CA
{viveksin|nevatia}@usc.edu

Ram Nevatia

Abstract

Most existing approaches for learning action models work by extracting suitable low-level features and then training appropriate classifiers. Such approaches require large amounts of training data and do not generalize well to variations in viewpoint, scale and across datasets. Some work has been done recently to learn multi-view action models from Mocap data, but obtaining such data is time consuming and requires costly infrastructure. We present a method that addresses both these issues by learning action models from just a few video training samples. We model each action as a sequence of primitive actions, represented as functions which transform the actor's state. We formulate model learning as a curve-fitting problem, and present a novel algorithm for learning human actions by lifting 2D annotations of a few keyposes to 3D and interpolating between them. Actions are inferred by sampling the models and accumulating the feature weights learned discriminatively using a latent state Perceptron algorithm. We show results comparable to state-of-art on the standard Weizmann dataset, with a much smaller train:test ratio, and also in datasets for visual gesture recognition and cluttered grocery store environments.

1. Introduction

Action recognition from videos has been a subject of active research in recent years, since effective solutions to this problem can have a wide range of applications including human-computer interaction (HCI), visual surveillance, search and retrieval among others. While several approaches have been explored, at their core they involve development of suitable modeling frameworks and low-level features. The different modeling approaches considered in previous work include SVM[12], HMMs[17][4], CRFs[18][15] and finite state machines[8]. In terms of feature extraction, shape based templates were used in [17], while a combination of shape and flow based features were

used in [10][16][5], and features around a sparse set of spatio-temporal interest points (STIPs) were used in [12]. Foreground based features still continue to be popular (e.g. [22][18][14]), but these approaches require fairly accurate foreground extraction for good results.

A key limitation of a majority of these approaches is that the models are learned from 2D features, and hence they are not generalizable across datasets and are also typically not robust to view and scale variations. This would either require collecting a prohibitively large training set, or re-training the models for each new dataset. Viewpoint variations are addressed in [14][16] by rendering 3D *Motion Capture (Mocap)* data of actions from multiple viewpoints, and matching these rendered models to the 2D features. [8] on the other hand, first tracks limbs using a pose tracker, and then lifts the tracks to 3D for matching with *Mocap* models. However, collecting such *Mocap* data is time consuming and requires expensive equipment.

We address both these limitations by learning effective 3D action models from just a few video samples. We reduce training requirements by representing complex actions as sequences of a small number of primitive actions represented by parametric functions. The parameters of the functions are learned from lifting 2D pose annotations in a few key frames in sample videos, to 3D and interpolating between them; this eliminates the need for *Mocap* data. The event models are mapped to a *Dynamic Bayesian Network (DBN)* which has associated potentials for transition between the actions, and a set of feature functions that measure the similarity between the states and observations. The relative weights of the different features are learned from training data discriminatively, using a novel *Latent State Perceptron Algorithm*, to train with only partially observed states. This two-step training process which first learns an action model, and then learns feature weights significantly reduces training requirements. We also introduce a novel inference algorithm to compute the action sequence by sampling the action models and accumulating the feature weights.

Our approach uses projections of 3D action models like [14][16], but we learn our models from 2D videos and hence do not require *Mocap* data. Also, we track poses by sampling from the action models in the continuous domain, while [14] and [16] sample from a discrete set of poses for each action. Our approach also differs from [8] since we lift only a few key pose annotations to 3D during learning and recognition is done by projecting the 2D models, while [8] lifts 2D pose tracks during recognition. Also [3] presents an approach for lifting and interpolating between keyposes using 3D kinematic constraints, similar to our model learning approach. However, unlike [3] which requires manual pose annotations for pose tracking, our tracking and recognition is completely automated. A similar approach is considered in [19], which first recognizes actions by keypose matching and then tracks joints by point transfer from the representative key frames. We on the other hand use action dynamics, which model temporal constraints on pose transitions, and can also recognize more complex actions and viewpoints. Thus our approach builds on several earlier works, but expands the capability possible with any of them.

We show results on three datasets: the Weizmann set [6] that has been used by several other researchers, a gesture dataset that we have constructed with a variety of arm gestures common in HCI applications and finally actions in a complex, real environment of a small grocery store. We obtain good results in all cases as described later in the paper.

In the rest of the paper, we first discuss action representation, inference algorithm and features in section 2. Next, we present the learning algorithm in section 3, followed by the results in section 4 and conclusion in section 5.

2. Action Representation and Recognition

Our action representation is based on the concept that a *composite* action can be decomposed into a sequence of *primitive* actions. Each primitive action *pe* modifies the *state* *s* of the actor to give a new state *s'*. This can be expressed in a functional form as $f_{pe}(s, s', N)$, which maps the current state *s* to the next state *s'* given a set of *parameters* *N*. This representation of actions is inspired by *Temporal Logic of Actions (TLA)*[11], which combines first-order logic and temporal logic and represents actions with logical predicates but doesn't deal with uncertainty and errors.

We assume that there is a known finite set of possible functions *f* to represent the primitives. This is common in many domains of interest; in particular, in human motion analysis, [7] shows that there are only 3 possible movements for any limb - *Rotate*, *Flex* and *Pause* although there is infinite variability in the possible rotation axes and angles. Further, the *Flex* action can be represented as the simultaneous rotation of two parts in opposite directions about the same axis (like thigh and knee when flexing the leg, or lower and upper arms when flexing the hand).

2.1. Graphical Model Representation

Given the action models in the form of parametric functions *f*, we embed them into a *Dynamic Bayesian Network (DBN)* which we call the *Dynamic Bayesian Action Network (DBAN)* illustrated in Figure 1.

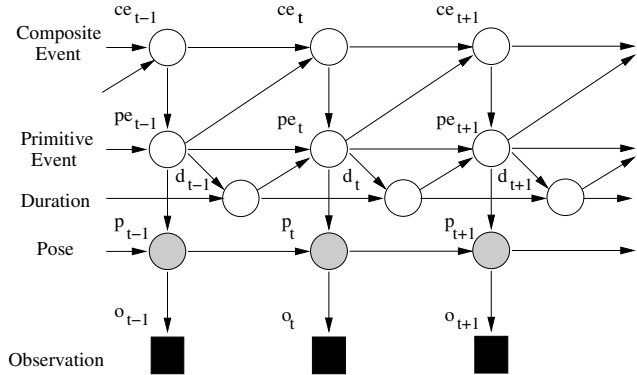


Figure 1. Dynamic Bayesian Action Network

The nodes in the topmost layer in the DBAN correspond to the composite actions like *walk*, *pickup*, etc. The middle layer corresponds to the primitives and the lowest layer corresponds to the pose. In addition, we also include primitive event durations in the model. Thus, the *state* s_t of the DBAN at time *t* is denoted by the tuple (ce_t, pe_t, d_t, p_t) .

The optimal state sequence $s_{[1:T]}^*$ for an observation sequence of length *T* is computed by maximizing the weighted sum of potentials similar to [2].

$$s_{[1:T]}^* = arg \max_{\forall s_{[1:T]}} \sum_{t=1}^T \sum_f w_f \phi_f(s_{t-1}, s_t, I_t) \quad (1)$$

where, $\phi_i(s_{t-1}, s_t, I_t)$ are observation and transition potentials and w_i is the weight vector that models the relative importance of the potential functions. Also note that we call our model a DBN, since we generalize the HMM in [2] by using a multi-variable state representation.

2.2. Pose Tracking and Recognition

During recognition, we first detect the person in the video using a state-of-the-art pedestrian detector. We then use a combined shape and foreground blob tracker to track and localize the person in each frame, through changing pose. The position and scale information available from the tracker is then used to adjust the predicted 3D pose obtained from the event model. The likelihood of the pose is then computed by matching the adjusted pose with low-level image features.

Since the pose p_t in the state s_t is continuous, an exhaustive search of the entire state space to recognize the state sequence is not possible. Instead we use a sampling based approach to recognize the best state sequence, by

storing the *top-K* states s_t at each frame t . This is similar to sampling from probability distributions in particle filters, except that the potential functions are un-normalized. For each state s_t , we first sample the *Action Transition Potential* $\phi_a(s_t, ce_{t+1}, pe_{t+1})$ to choose the next actions (ce_{t+1}, pe_{t+1}) . Next, we sample from the *Pose Transition Potential* $\phi_p(p_t, pe_{t+1}, p_{t+1})$ to choose the next pose p_{t+1} . Here, $\phi_p(p_t, pe_{t+1}, p_{t+1})$ represents a distribution over the parameters N in the function $f_{pe_{t+1}}(p, p', N)$ corresponding to primitive pe_{t+1} . Further, since the orientation of the actor w.r.t the camera can change in actions like *walk*, we scan a window within 10° of the previous orientation to choose the current orientation. Next, we compute the observation potential $\phi_{obs}(s_{t+1}, o_{t+1})$, and recognize the best state sequence by accumulating the weighted sum of potentials. Algorithm 1 presents pseudocode for the algorithm.

Algorithm 1 Inference Algorithm

- Sample initial distribution $\phi_{init}(s)$ to get initial states $S_0 = \{(s_0^{(i)}, \alpha_0^{(i)}) | i = 1..K\}$
 - for** $t = 1$ to T **do**
 - for** $i = 1$ to K **do**
 - Action Prediction:
 - Sample $\langle ce_{t+1}^{(i)}, pe_{t+1}^{(i)} \rangle \sim \phi_a(s_t, ce_{t+1}, pe_{t+1})$
 - Pose Prediction:
 - Sample $p_{t+1}^{(i)} \sim \phi_p(p_t, pe_{t+1}, p_{t+1})$
 - Weight Estimation:
 - $\alpha_{t+1}^{(i)} = \alpha_t^{(i)} + \sum_f w_f \phi_f(s_t, s_{t+1}, o_{t+1})$
 - end for**
 - end for**
-

The sum in the weight estimation step above, computes the weighted sum of the transition and observation potentials. This is similar to the inference used in CRF-Filters [13], but differs in two crucial aspects - 1) We use a two-stage prediction, first for actions and then for poses. 2) We accumulate the weights over time for each sample, instead of resampling based on the instantaneous weights, so that our recognition algorithm does not drift due to local errors.

The representation and inference framework described so far is general and can be applied in many domains. Now, we will describe the specific observation and transition potentials we used in our implementation.

2.3. Transition Potential

In our implementation, we allow all possible transitions between the composite events ce . We model the primitive transitions by using the primitive event durations in the *log*

of *signum* function as follows:

$$\phi_a(s_t, ce_{t+1}, pe_{t+1}) = \begin{cases} -\ln\left(1 + e^{\frac{d_t - \mu(pe_t) - \sigma(pe_t)}{\sigma(pe_t)}}\right) & pe_{t+1} = pe_t \\ -\ln\left(1 + e^{-\frac{d_t - \mu(pe_t) + \sigma(pe_t)}{\sigma(pe_t)}}\right) & pe_{t+1} \neq pe_t \end{cases} \quad (2)$$

Here, $\mu(pe_t)$ and $\sigma(pe_t)$ are the mean and variance of primitive event pe_t durations learned from the training data. With this definition, the probability of staying the same primitive pe_t decreases near the mean duration $\mu(pe_t)$ and the probability of transition to a new primitive increases.

We model the pose transition potential with the *log* of normal distribution as:

$$\phi_p(p_t, pe_{t+1}, p_{t+1}) = -\frac{(p_{t+1} - p_t - \theta_{mean})^2}{2\theta_{var}^2} \quad (3)$$

Here, $(p_{t+1} - p_t)$ denotes the amount by which pose p_{t+1} has changed from p_t . θ_{mean} and θ_{var} are the mean and variance of the amount by which primitive pe_{t+1} changes the pose at each frame, and is also learned during training.

2.4. Observation Potential

We compute the observation potentials of a state $\phi_{obs}(s_t, o_t)$ using features we extract from the video. For robustness, we use multiple features to compute the likelihood, such as foreground overlap, difference image match, and a novel *grid-of-centroids* based approach to match foreground distribution. Figure 2 illustrates the observation potentials we extract.

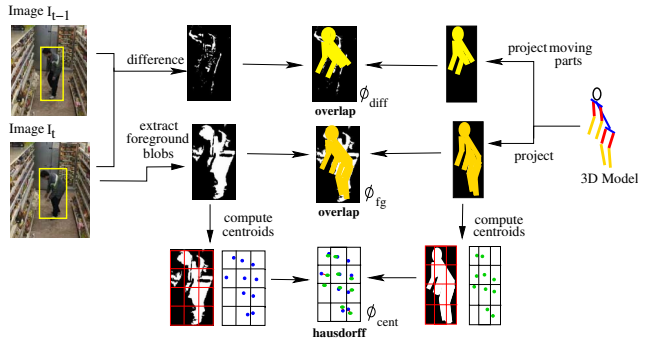


Figure 2. Computation of Observation Potentials

Foreground Overlap: The foreground overlap score of the pose p is computed by accumulating the foreground pixels overlapping with orthographic projection of pose p .

$$\phi_{fg}(p, I_{fg}) = |I_{fg} \cap Proj(p, I)|$$

where, $Proj(p, I)$ is the projection of pose p on image I and I_{fg} is the foreground image.

Difference Image Matching: We use differences between the frames I_t and I_{t-1} within the person detection window as an estimate of instantaneous changes in the observed

pose. The observation potential of s for change in pose is modelled as the overlap between the difference image and projection of moving body parts on the image.

$$\phi_{diff}(s, I_t, I_{t-1}) = |Diff(I_t, I_{t-1}) \cap Proj(p_{change}, I)|$$

where, $Diff(I_i, I_j)$ is the difference image between the frame i and frame j . This measures the similarity between the instantaneous motion observed in the video, and the pose change.

Foreground matching using Bag of centroids: In practice, extracted foreground blobs are often fairly noisy and the foreground overlap measure is very sensitive to pose misalignments. So in addition to the overlap measure, we propose *grid-of-centroids* to match the foreground blob distribution with the pose. To compute the grid-of-centroids, we place an $m \times n$ grid on the image I and compute the centroids of the foreground pixels within each cell. We represent this grid-of-centroids by the set of the non-zero centroids $\nu_{[1, n_c]}(I)$, where n_c is the number of non-zero centroids. To compute the potential of pose p using the grid-of-centroids, we compare the grid-of-centroids computed over the foreground blobs within the person detection window, and those computed from the projection of p on the image $Proj(p, I)$. We use Hausdorff measure to compute the similarity score between the grid-of-centroids.

$$\phi_{cent}(p, I_{fg}) = \max_{i \in [1, n_c]} \min_{j \in [1, n_{c'}]} \|\nu_i(Proj(p, I)) - \nu_j(I_{fg})\|^2$$

where, $\|\cdot\|$ is the euclidean norm.

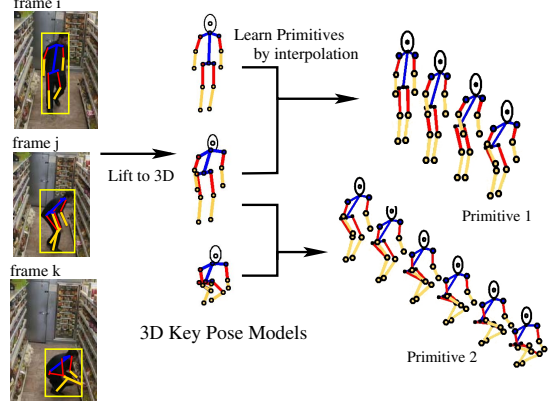
Position change matching: In addition to the observation potentials described, we also include a motion weight since an event can be performed while the actor is either standing or while he is in motion, say walking or running. We compute this by matching expected change in position with the position change observed using the person tracker. Given the observed change in position δ_{pos} , we define the potential function of a state s with primitive event pe as

$$\phi_{pos}(s, \delta_{pos}) = -\frac{(\delta_{pos} - \mu_{pos}^{pe})^2}{2\sigma_{pos}^{pe}} \quad (4)$$

where, μ_{pos}^{pe} , σ_{pos}^{pe} are mean and variance of the change in person position for the primitive event pe . For stationary events, $\mu_{pos}^{pe} = 0$.

3. Action Learning

Learning action models in our framework involves two problems - (1) Learning the parameters N in the primitive event definitions $f_{pe}(p, p', N)$. (2) Learning the weights w_k of the different potentials. We will now describe our algorithms for learning these parameters in detail.



Annotated Key Poses

Figure 3. Model Learning Illustration for *Crouch* action with 3 keyposes and 2 primitives

3.1. Model Learning

Since the functional form of f_{pe} is known from prior knowledge, learning N involves *curve-fitting* from a set of (p, p') pairs. In particular, human limb motions involve rotations of the form $Rotate(part, axis, degree)$, where the parameter $axis$ is the axis of rotation and $degree$ is amount by which the part is rotated. Now, if we know the center of rotation, $axis$ and $degree$ can be computed from the start and end locations of $part$ using simple geometry. We take such an approach as illustrated in Figure 3.

3.1.1 KeyPose Annotation and 3D Lifting

Given an action, we first choose a set of *keyposes* that best represent the action. In our work, we do this manually but we can choose them automatically from a sample video by computing a *motion energy* function similar to [14]. Next we annotate the 2D joint locations of the frames containing the key poses in a sample action video. In addition, we also annotate for the relative depth ordering between the joints, the height H of the person in pixels, and also the approximate pan and tilt of the camera. Our approach to lift 3D pose from 2D annotations is similar to [21].

Given the 2D (x, y) annotations, we lift the pose to 3D (x, y, z) by comparing them to part lengths of an idealized human model as follows:

$$z_{j2} - z_{j1} = \begin{cases} \sqrt{l_{j1, j2}^2 - \Delta x_{j1, j2}^2 - \Delta y_{j1, j2}^2} & z_{j2} > z_{j1} \\ -\sqrt{l_{j1, j2}^2 - \Delta x_{j1, j2}^2 - \Delta y_{j1, j2}^2} & z_{j2} < z_{j1} \\ 0 & z_{j1} = z_{j2} \end{cases} \quad (5)$$

where,

- $j1, j2$ are possible adjacent joints.
- $\Delta x_{j1, j2} = |x_{j2} - x_{j1}|$, $\Delta y_{j1, j2} = |y_{j2} - y_{j1}|$.
- $l_{j1, j2}$ is the length of part between joints $j1, j2$.
- $z_{j1} < z_{j2}$ indicates that joint $j1$ is in front of joint $j2$.

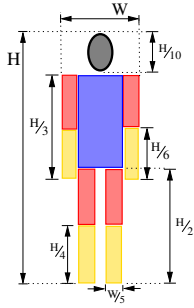


Figure 4. Part lengths of ideal human model

The set of possible $(j1, j2)$ are *(Shoulder, Elbow)*, *(Elbow, Wrist)*, *(Hip, Knee)*, *(Knee, Ankle)* on both the left and right sides, and *(Center Hip, Neck)*. We set the part lengths $l_{j1, j2}$ as fractions of the person’s image height H as illustrated in figure 4. Since $\Delta x_{j1, j2}$ and $\Delta y_{j1, j2}$ are known from the annotations, we only need to compute the z_j to estimate the 3D joint positions. We then infer the relationship between z_{j1} and z_{j2} for every connected pair of joints from the relative depth information. Given this information, equation (5) forms a set of linear equations in z_j . We set $z_{CenterHip}=0$ and solve for z_j to get the 3D keyposes. Next, we *normalize* the 3D poses to a standard height and orientation and also let the hip center be at $(0, 0, 0)$, since different videos of an action can be shot at different viewpoints and scale.

3.1.2 Pose Interpolation and Model Learning

We now describe our approach to learning the model, from the key pose annotations of a single action video. After we have lifted the key pose annotations to 3D and normalized them, we define the primitive actions to be the motions that interpolate between each of the key poses.

Let pe denote the primitive event corresponding to the transition from keypose k_s to k_e . Also, let t_s and t_e be the frames in the action video when keypose k_s and k_e occur respectively. Since the keyposes are normalized to the same height, orientation and location we only need to compute the transformations of the body parts between the keyposes. We do this by computing the rotations of the torso, the left and right upper arms, lower arms, upper legs and lower legs in that order.

As we discussed earlier, all limb motions can be expressed in terms of rotations [7] as $Rotate(part, axis, \theta)$, where $axis$ is the axis rotation and θ is the angle of rotation. Since $Rotate$ has a fixed functional form, to learn the primitive pe we need to compute $axis$ and θ from k_s and k_e for every $part$. Let $(j1, j2)$ denote the start and end joints of $part$. For example if $part=left\ upper\ arm$, then $j1=left\ shoulder$, $j2=left\ elbow$. Also, let $k(j, x)$ denote the x -coordinate of joint j in keypose k . Also let:

$$\begin{aligned} \Delta k_{s,x} &= k_s(j2, x) - k_s(j1, x), \Delta k_{s,y} = k_s(j2, y) - k_s(j1, y) \\ \Delta k_{s,z} &= k_s(j2, z) - k_s(j1, z), \Delta k_{e,x} = k_e(j2, x) - k_e(j1, x) \\ \Delta k_{e,y} &= k_e(j2, y) - k_e(j1, y), \Delta k_{e,z} = k_e(j2, z) - k_e(j1, z) \end{aligned}$$

and let,

$$V_s = (\Delta k_{s,x}, \Delta k_{s,y}, \Delta k_{s,z}), V_e = (\Delta k_{e,x}, \Delta k_{e,y}, \Delta k_{e,z})$$

Now, we compute $axis$ and θ as:

$$axis = \frac{V_s \times V_e}{|V_s \times V_e|}; \theta = \frac{\cos^{-1}\left(\frac{V_s \cdot V_e}{|V_s||V_e|}\right)}{t_e - t_s + 1} \quad (6)$$

The approach described so far computes the model by lifting 2D annotations of a few key poses to 3D and interpolating between them, from a *single* action video.

When we have annotations from multiple videos, we first collect the annotations corresponding to each keypose. We lift the annotations to 3D, normalize them, and then compute the mean and variance of the joint locations. We then compute the primitive action parameter $axis$ from the mean keyposes as in equation (6). We compute θ for each action sample, and then compute θ_{mean} and θ_{var} as the mean and variance of the θ ’s. Further we also compute mean duration $\mu(pe)$ and variance $\sigma(pe)$ from $d = t_e - t_s + 1$ for the duration models.

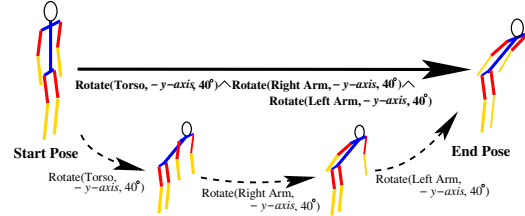


Figure 5. Primitive Learning by Pose Interpolation

3.2. Feature Weight Learning

We now describe our approach for training the feature weights $\bar{w} = \{w_f\}$ associated with the feature potentials $\phi_f(\cdot)$. We define the likelihood of a state sequence $s_{[1:n]}$ for a feature f in image sequence $I_{[1:n]}$ as:

$$\Phi_f(I_{[1:n]}, s_{[1:n]}) = \sum_{i=1}^n \phi_f(h_i, s_i) \quad (7)$$

where, $h_i = \langle I_{[1:n]}, s_{i-1}, i \rangle$. We formulate feature weight estimation as minimization of the log likelihood error function $\mathcal{E}(\bar{w})$ over the entire training set \mathcal{T} . Thus, the log likelihood error function for the given set of weights \bar{w} is:

$$\sum_{i \in \mathcal{T}} \delta(ce_i^o \neq ce_i^{gt}) \sum_f w_f \left(\Phi_f(I_{[1:n]}, s_{[1:n]}^{gt}) - \Phi_f(I_{[1:n]}, s_{[1:n]}^o) \right)$$

where, $s_{[1:n]}^{gt}$ is the ground truth label sequence and $s_{[1:n]}^o$ is the estimated state sequence obtained using the inference algorithm described in Algorithm 1.

Due to this log linear formulation of the likelihood error function, we can learn the weight vector using the *Voted Perceptron algorithm* [2]. However, this uses completely labeled training data to estimate the model parameters, while

in our case the ground truth label sequence only contains event annotations. Hence, we extend the Voted Perceptron algorithm to deal with partially labeled data. In particular, we add an additional step of estimating the latent variables based on the current parameters and use it to compute the training error.

Algorithm 2 Discriminative Feature Weight Learning

Randomly set the initial weight vector \bar{w}

for $m = 1$ to M **do**

for $i = 1$ to N **do**

 • Set $\Delta_f = 0$

 • Use algorithm 1 to compute the most likely state sequence $s_{[1:n_i]}^o$ on the i^{th} training sequence \mathcal{T}^i using the current weight vector \bar{w} .

if $ce_{[1:n_i]}^o \neq ce_{[1:n_i]}^{gt}$ **then**

 • Given the labeled event sequence, estimate the most likely state sequence using algorithm 1 (without the action prediction step)

$\tilde{s}_{[1:n_i]} = \arg \max_{\forall p} \sum_f w_f \Phi_f(\mathcal{T}^i, \langle ce^{gt}, pe^{gt}, d^{gt}, p \rangle_{[1:n_i]})$

 • Collect the feature errors

$\Delta_f = \Delta_f + \Phi_f(\mathcal{T}^i, \tilde{s}_{[1:n_i]}) - \Phi_f(\mathcal{T}^i, s_{[1:n_i]}^o)$

end if

end for

 • Update the weight vector

$w_f = w_f + \frac{\Delta_f}{\|\Delta_f\|_{L^1}}$

end for

The training algorithm summarized in Algorithm 2, takes M passes over the training set. For each training sequence, the most likely state sequence with the current weight vector is computed using algorithm 1. If the estimated composite event is not correct, ground truth state sequence is estimated from the labeled event sequence using algorithm 1 without the action prediction step (since the action is known). The feature errors between the observed and the ground truth sequence are collected over the entire training set and is used to update the weight vector. In our experiments, we learnt multiple randomly initialized feature weights for each action model and use the weights that achieve the highest training accuracy.

4. Experiments

We now present our results on three datasets.

Weizmann Dataset [6]: The Weizmann human action recognition set contains 93 video sequences with nine different actors performing 10 different full body actions. Each video contains multiple, consecutive instances of one action. In our experiments, we automatically segment these instances as part of the recognition process. However, we

do not allow transitions between different composite actions since other approaches make a similar assumption. Figure 7 shows sample results obtained by our algorithm on this dataset. We performed rigorous experiments with various train:test ratios as illustrated in figure 6(a). Our method achieved 97% accuracy from models obtained from just 1:8 train:test ratio, and 99.5% with a 3:6 train:test ratio. Table 1 presents a comparison of our results with other methods, and shows that our method produces results similar to state-of-art methods but with much smaller train:test ratio.

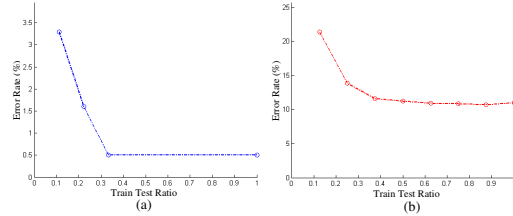


Figure 6. Error rate vs *train:test* ratio (a) Weizmann Dataset; (b) Gesture Dataset

	Train:Test	Recognition Accuracy
Jhuang et al [9]	6:3	98.8
Space-Time Shapes [6]	8:1	100
Fathi et al [5]	8:1	100
Sun et al [20]	3:6	87.3
Proposed Method	1:8	96.7
	3:6	99.5
	8:1	99.5

Table 1. Evaluation Results on the Weizmann Dataset

Next, we evaluate our approach on the Weizmann robustness test dataset. This set includes 11 instances of walk action in presence of background clutter and occlusions (*robust deform*), and 9 instances of walk action at different pan angles from 0° to 81° (*robust view*). The recognition task requires correctly detecting the walk action against remaining 9 actions in the original dataset. Without any re-training, our algorithm correctly recognizes the walk action on all 20 instances. For robust view set, we apply our algorithm to search over different pan angles from 0 to 90 degrees. Figure 8 shows samples results obtained by our algorithm and the corresponding foreground masks used as input to our algorithm. [6] also correctly recognizes all the action instances but requires accurate foreground extraction. [1] uses clouds of spatio temporal features to correctly recognize 19 instances but fails on the video where there is background motion around the body part salient to the action such as walking with dog (see figure 8(b)). Use of explicit human body model makes our algorithm robust to such background distractions.

Gesture Dataset: We collected 5-6 instances of 12 actions from 8 different actors in an indoor lab setting. The dataset contains a total of about 500 action sequences across all actions. The videos are 852×480 resolution, and the person’s height is 200-250 *pixels*. Figure 9 shows sample results on

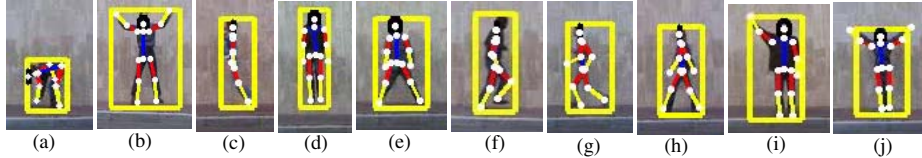


Figure 7. Results on the Weizmann Dataset: inferred pose is overlaid on top of the image and illustrated by limb axes and joints

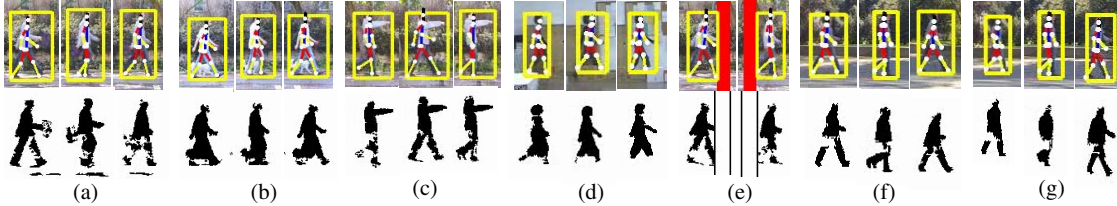


Figure 8. Results on the Weizmann Robustness Dataset: walking with (a) swinging bag (b) dog (c) moonwalk (d) occluded feet (e) Occlusion by pole. (f) and (g) with viewpoint variations

this dataset. Figure 6(b) illustrates *error rate Vs train:test* curve. These results show that the models learnt even with a few samples are sufficient to distinguish among actions with relatively small differences.

Grocery Store Dataset: This dataset is collected in a more cluttered setting of a grocery store. The camera is static and has a downward tilt of $\approx 20^\circ$. In each video, an actor enters the scene, picks up an item and leaves. The action set includes 3 full body actions - *walking*, *pickup from shelf*, *crouch and pickup*, with 16 videos from 8 different actors and the observed size of the actor varies from 200 to 375 pixels in a 852×480 frame. The main challenges here are poor foreground extraction, highly articulated and ambiguous poses and large changes in the orientation and scale of the actor. The poor foreground extraction is due to the shadows, changes in lighting due to reflection from outside traffic and color similarities between actor’s clothing and the background (see figure 10). Also, the actions are not segmented *a priori*; temporal segmentation is part of the recognition process. Sample results are shown in figure 10, which also shows the complexity of the environment and pose articulations.

Dataset	Train:Test ratio	Recognition (% accuracy)	2D Tracking (% accuracy)	Speed (fps)
Gesture	3 : 5	90.18	94.75	8
Grocery	1 : 7	100	85.80	1.6

Table 2. Performance on Gesture and Grocery Store dataset

Table 2 summarizes the quantitative results on gesture and grocery store datasets, again illustrating the low training requirements of our approach. For the Grocery store sequence, the recognition sequence labeling results are perfect with overlap between detected and annotated event intervals being about 90%; note that it is difficult to mark the boundaries between actions precisely to construct the ground truth. The results on Grocery dataset are better than those on Gesture dataset, in spite of the higher apparent complexity and ambiguity in a single frame, because

of the smaller number of actions which are quite distinct from each other and the temporal models embedded in the DBAN.

We also evaluated the pose tracking errors. As we do not have 3-D ground truth, we measure the accuracy of estimated 2D parts. A part estimate is considered correct if its segments lies within 50% of the length of the ground-truth segment. We only include the parts that are involved in the action (e.g. only hands in the gesture sequence) and only sequences where the correct action is recognized (in analogy with object detection where spatial accuracy of false alarms is not measured). Third column of the Table 2 provides the results. Our pose tracking accuracy is sufficient for inferring actions and their relationships to objects in the surroundings but possibly not for motion capture, which is not our objective.

The last column of Table 2 gives the computational speeds on a 3 GHz Xeon CPU running Windows/C++ programs. The numbers are for the entire system including person detection, foreground extraction and recognition. Most of the computation arises from low-level feature extraction and our inference algorithm itself runs close to real time (20-30 fps).

5. Conclusion

To summarize, we have presented a general framework for simultaneous tracking and action recognition using probabilistic graphical models, that has minimal training requirements. We have made several specific contributions in feature extraction, learning and inference of actions. We have demonstrated our learning and inference algorithms in the standard Weizmann dataset [6], for hand gesture recognition as well as for action recognition in realistic videos collected in a grocery store. Our results indicate that we can learn very effective action models with just a few training samples, and only partial state annotations.

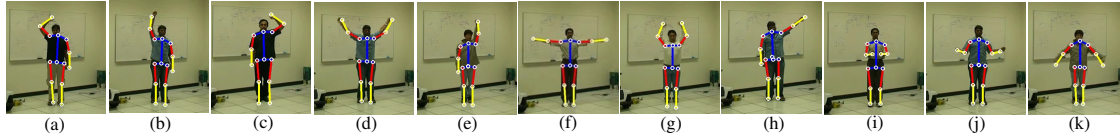


Figure 9. Results on the Gesture Dataset: inferred pose is overlaid on top of the image and illustrated by limb axes and joints

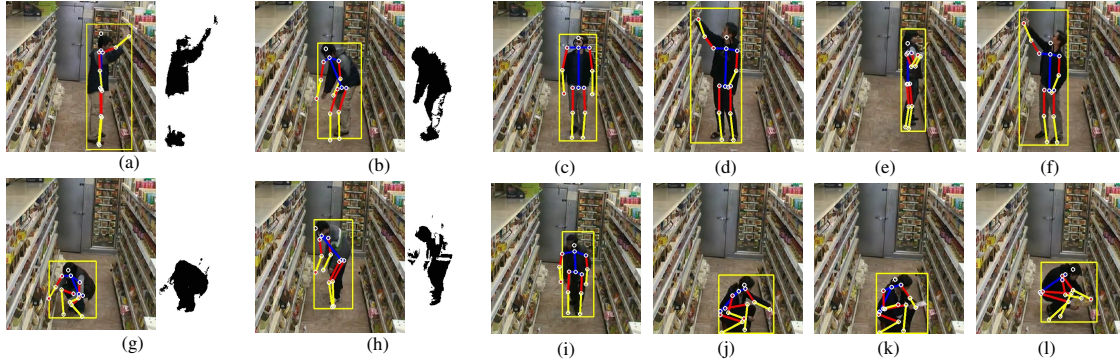


Figure 10. Results on the Grocery Store Dataset: The bounding box shows the actor's position and the inferred pose is overlaid on top of the image, illustrated by limb axes and joints. (a-b), (g-h) show extracted foreground blobs (within the detection window) used as input.

Acknowledgements

The authors thank Fei Sha for useful discussions on the Voted Perceptron algorithm. Authors would also like to thank Prithviraj Banerjee and Pramod Sharma for help with the annotations. This research was supported, in part, by the Office of Naval Research under Contract #N00014-06-1-0470.

References

- [1] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In *CVPR*, 2009. 6
- [2] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, 2002. 2, 5
- [3] D. DiFranco, T. Cham, and J. Rehg. Reconstruction of 3d figure motion from 2d correspondences. In *CVPR*, pages 1:307–314, 2001. 2
- [4] T. Duong, H. Bui, D. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. *CVPR*, 1:838–845, 2005. 1
- [5] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *CVPR*, 2008. 1, 6
- [6] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *T-PAMI*, 29(12):2247–2253, December 2007. 2, 6, 7
- [7] A. Hutchinson and G. Balanchine. *Labanotation: The System of Analyzing and Recording Movement*. ISBN: 0878305270, 1987. 2, 5
- [8] N. Iqbal and D. A. Forsyth. Searching video for complex activities with finite state models. In *CVPR*, 2007. 1, 2
- [9] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007. 6
- [10] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, 2007. 1
- [11] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994. 2
- [12] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005. 1
- [13] B. Limketkai, D. Fox, and L. Liao. Crf-filters: Discriminative particle filters for sequential state estimation. In *ICRA*, pages 3142–3147, 2007. 3
- [14] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, 2007. 1, 2, 4
- [15] L.-P. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*, 2007. 1
- [16] P. Natarajan and R. Nevatia. View and scale invariant action recognition using multiview shape-flow models. In *CVPR*, 2008. 1, 2
- [17] V. Shet, S. N. Prasad, A. Elgammal, Y. Yacoob, and L. Davis. Multi-cue exemplar-based nonparametric model for gesture recognition. In *ICVGIP*, 2004. 1
- [18] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional random fields for contextual human motion recognition. In *ICCV*, pages 1808–1815, 2005. 1
- [19] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *ECCV (I)*, pages 629–644, 2002. 2
- [20] X. Sun, M. Chen, and A. Hauptmann. Action recognition via local descriptors and holistic features. In *CVRPW*, pages 58–65, 2009. 6
- [21] C. J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. In *CVIU*, volume 80, pages 349–363, 2000. 4
- [22] D. Weinland, R. Ronfard, and E. Boyer. Automatic discovery of action taxonomies from multiple views. In *CVPR*, pages II: 1639–1645, 2006. 1