Searching for Complex Human Activities with No Visual Examples

Nazli Ikizler & David A. Forsyth

Kan Ouivirach

Computer Science and Information Management
Asian Institute of Technology

June 9, 2009

Reference

- Ikizler, N. & Forsyth, D. A. (2008). Searching for Complex Human Activities with No Visual Examples. In *IJCV'08: International Journal of Computer Vision* (pp. 337-357).

Outline

Introduction

- Understanding what people doing is difficult.
- Application possibilities range from medical to security.
- Major difficulties:
  - Good kinematic tracking is hard;
  - Models have too many parameters to be learned;
  - There is no taxonomy for everyday behavior.

Introduction

Desirable activity recognition and retrieval system should:

- Handle different clothings and varying motion speeds of different actors;
- Accommodate the different timescales over which actions are sustained;
- Allow composition across time and across the body;
- Be a manageable number of parameters to estimate;
- Perform well in presence of limited quantities of training data;
- Be indifferent to viewpoint changes;
- Require no example video segment for querying.

Problems

- Training data is deficient because of a wide range of variations of behavior. Different subjects may perform the actions with different speeds in various outfits.
- Large model might solve it, but note that the more parameters we have, the more training data we have to provide. (Overfitting)
- Dealing with the composite nature of activities is difficult. Most of the current works deal with simple actions.
- Tracker responses are noisy, especially when the background is cluttered.

Proposed Solution

All these points suggest:

- Having a model of activity consisting of pieces which are relatively easily learned and are then combined together within a model of composition.
- To overcome the data shortage problem, they propose to make use of motion capture data. This data does not consist of everyday actions, but rather a limited set of American football movements. (Transfer learning problem)

Proposed Solution

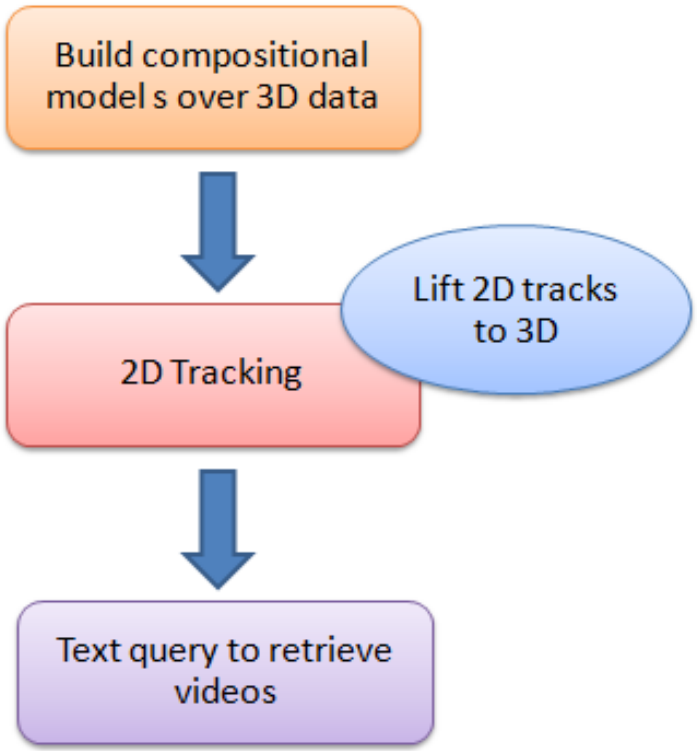

Figure: Processing flow

Related Work

There are 3 major threads:

1. Use motion clusters of the same type and explore the statistics or relative ordering of these clusters.
2. Use (typically, hidden Markov) models of dynamics or temporal logics to represent the crucial order relations between states that constrain activities.
3. Use discriminative methods, either with spatio-temporal templates or using 'bag-of-words'.

Motion Primitives

- A natural method for building models of motion on longer timescales is to identify clusters of motion of the same type.
- And then consider the statistics of how these motion primitives are strung together.
- Need to estimate fewer parameters and also can pool examples to do so.
- Feng and Perona (2002) described a method that first matches motor primitives at short timescales, then identifies the activity by temporal relations between primitives.

Methods with Explicit Dynamical Models

- Yamato et al. (1992) used Hidden Markov Models (HMMs) to recognize tennis strokes.
- Zhao and Nevatia (2004) used a finite-state model of walking, running and standing, built from motion capture.

Methods with Partial Dynamical Models

- Siskind (2003) described methods to infer activities related to objects – from an event logic formulated around a set of physical primitives.

Methods with Discriminative Models

- *Methods Based on Templates*: Ben-Arie et al. (2002) recognized actions by first finding and tracking body parts using a form of template matcher and voting on lifted tracks.
- *Bag-of-Words Approaches*: Laptev et al. (2003) introduced the notion of 'space-time interest points' and use SVMs to recognize actions.
- *Transfer Learning*: Farhadi et al. (2007) built an application involving transfering American Sign Language (ASL) words learned from a synthetic dictionary to real world data.

Representing Acts, Actions and Activities

They distinguish between:

- Short-timescale representations (acts), like a forward-step;
- Medium-timescale actions, like walking, running, jumping, whose temporal extent can be short and typically composites of multiple acts;
- Long-timescale activities, which are complex composites of actions.

Representing Acts, Actions and Activities

- Since they want complex, composite activities to share a vocabulary of base units, they use the kinetic configuration of the body as distinctive feature.
- Ignore limb velocity and accelerations because actions can be performed at varying speeds. (still a useful clue)

Acts in Short Timescales

- They form one snippet for each leg and one for each arm; omit the torso because torso motions appear not to be particularly informative in practice.
- Each limb in each frame is represented with the vector quantized value of the snippet centered on that frame.
- That is, they apply k-means to the 3D representation of snippets the limbs.

Limb Action Models

- Using a vague analogy with speech, they build a large dynamical model with the minimum of parameter estimation.
- In speech studies, in order to recognize words, phoneme models are built and joined together to form word models.
- By applying this idea, they first build a model of the action of each limb (arms, legs) for a range of actions using Hidden Markov Model (HMM).
- Select a set of 9 actions by hand, with the intention of modeling their motion capture data. (e.g. walking)
- Motion capture data was released by Electronic Arts in 2002. It consists of assorted football movements.

Limb Activity Models

- Having built atomic action models, they string the limb models into a larger HMM by linking states that have similar emission probabilities.
- Link between states m and n of the different action models A and B if the distance is minimal.

$$\mathrm{dist}(A_m, B_n) = \sum_{o_m=1}^{N} \sum_{o_{n=1}}^{N} p(o_m)p(o_n)C(o_m, o_n)$$

where o_m and o_n are the emissions, $p(o_m)$ and $p(o_n)$ are the emission probabilities of respective action model states A_m and B_n, N is the number of possible emissions and $C(o_m, o_n)$ is the Euclidean distance between the emissions centers.

Limb Activity Models

They assign uniform probability to transition between actions an transition to the same action.

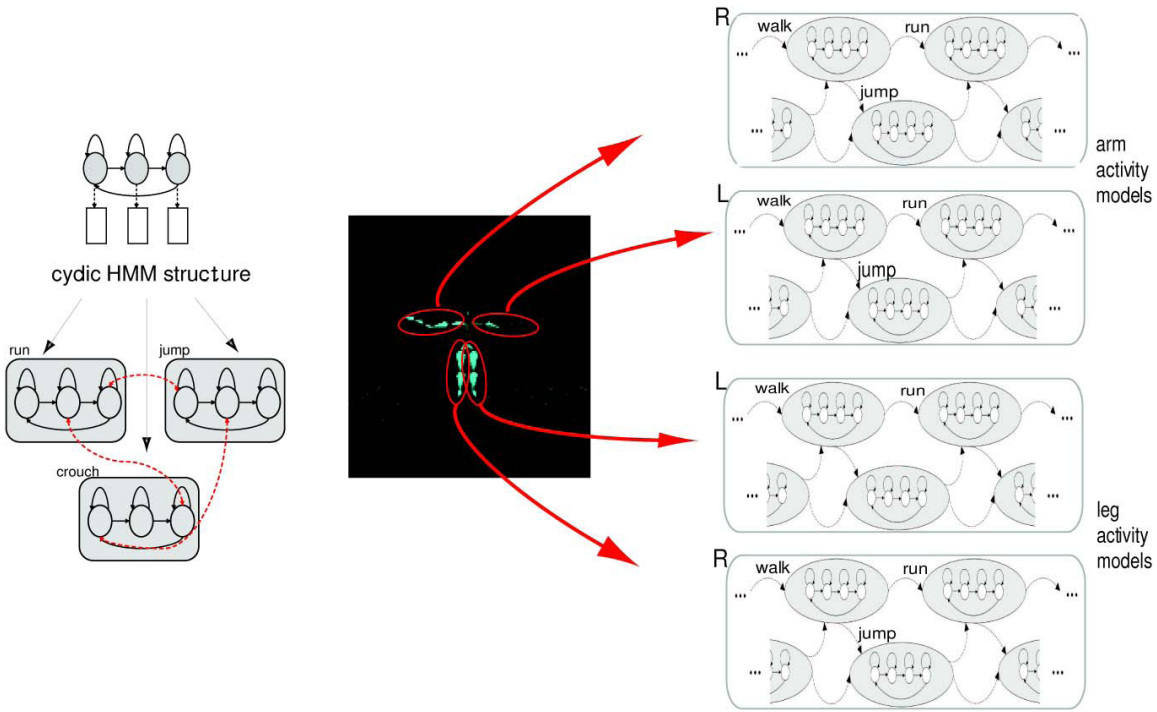

Figure: Limb Activity Models

Tracking

- Track motion sequences with the tracker of Ramanan et al. (2005).
- Obtain an appearance model by detecting a lateral walk pose, then detect instances in each frame using the pictorial structure method of Felzenszwalb and Huttenlocher (2005).
- No need for background modeling
- Capable of identifying the distinct limbs

Tracking

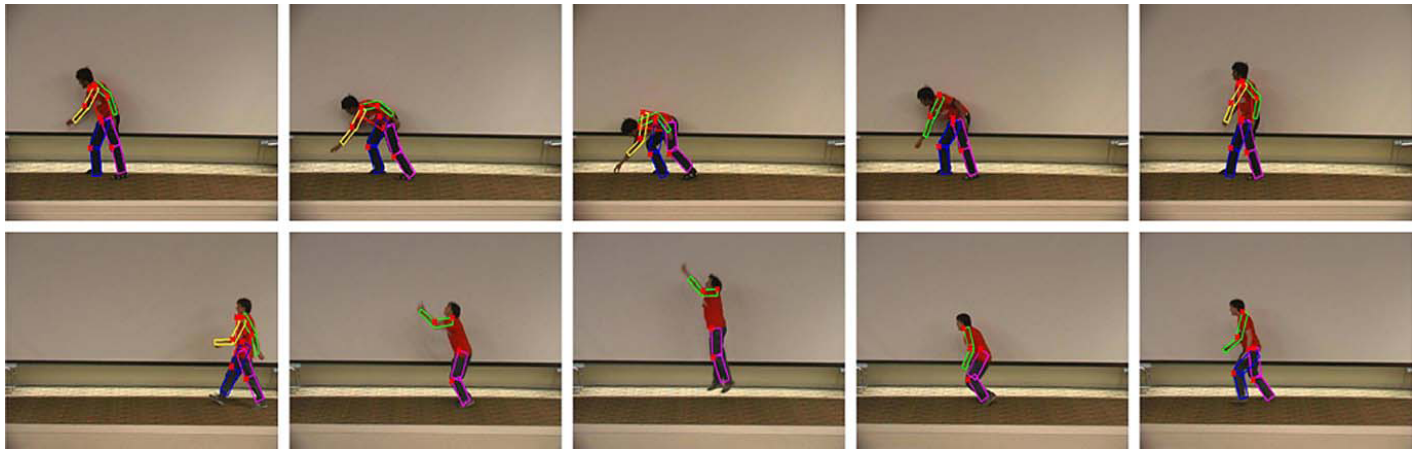

Figure: Example Tracks

Lifting 2D Tracks to 3D

- Obtain 3D reconstructions by matching projected motion capture data to image data by matching snippets of multiple motion frames.
- A complete sequence incurs a per-frame cost of matching the snippet centered at the frame, and a frame-frame transition cost which reflects:
  - the extent of the movement;
  - the extent of camera motion.
- Use dynamic programming to get the best sequence.
- Smoothing snippets – rather than frames – appears to significantly reduce reconstruction ambiguity (Forsyth et al. 2006).

Lifting 2D Tracks to 3D Algorithm

Algorithm 1 Lifting 2D Tracks to 3D

for each camera $c \in C$ **do**
  for all pose $p \in mocap$ **do**
    $\sigma_{pc} \leftarrow projection(p, c)$
  end for
  camera_transition_cost $\delta(c_i, c_j) \leftarrow (c_i - c_j) \times \alpha$
end for
for each $l_t \in L$ (leg segments in 2D) **do**
  for all $p \in mocap$ and $c \in C$ **do**
    $\lambda(l_t, \sigma_{pc}) \leftarrow match\_cost(\sigma_{pc}, l_t)$
    $\gamma(l_t, l_{t+w})$
    $\leftarrow transition\_cost(\lambda(l_t, \sigma_{pc}), \lambda(l_{t+w}, \sigma_{pc}))$
  end for
end for
do dynamic programming over δ, λ, γ for L
$c_{legs} \leftarrow$ (minimum cost camera sequence)
for each $a_t \in A$ (arm segments in 2D) **do**
  for $c_\epsilon \leftarrow$ neighborhood ϵ of c_{legs} and pose $p \in mocap$ **do**
    compute $\lambda(a_t, \sigma) \leftarrow match\_cost(\sigma_{pc_\epsilon}, a_t)$
    compute $\gamma(a_t, a_{t+w})$
    $\leftarrow transition\_cost(\lambda(a_t, \sigma_{pc_\epsilon}), \lambda(a_{t+w}, \sigma_{pc_\epsilon}))$
  end for
end for
do dynamic programming over δ, λ, γ for A

Lifting 2D Tracks to 3D

- Decompose the body into 4 quarters (2 arms, 2 legs).
- Match the legs using the snippet method, but allowing the left and right legs to come from different snippets of motion capture, making a search over 20 camera viewing directions.
- Choose arms in a similar way conditioned on the choice of legs, requiring the camera to be close to the camera of the legs.

Lifting 2D Tracks to 3D

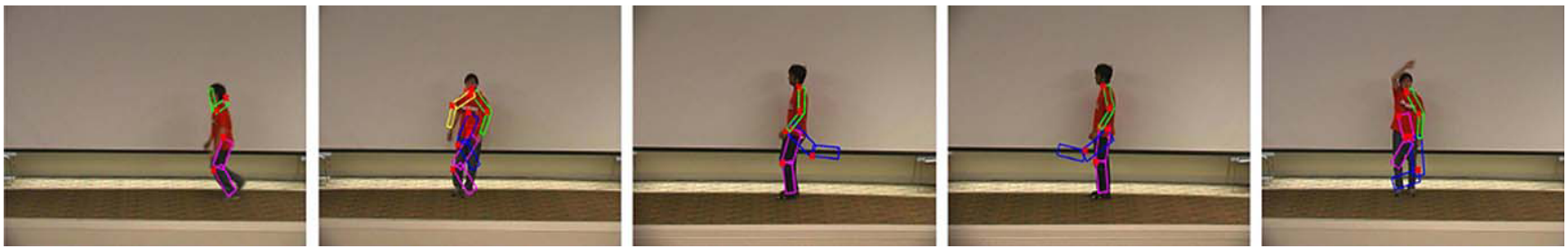

Figure: Bad tracks

Representing the Body

- They can now represent the body's behavior for any sequence of frames with $P(\text{limb activity model}|\text{frames})$.
- The model is built entirely on motion capture data.
- Computing a forward algorithm pass of the lifted sequences over the activity models gives us a posterior probability map representation for each video.
- This indicates the likelihood of each snippet to be in a particular state of the activity HMMs over the temporal domain.

Representing the Body

The posterior probability of a set of action states $\lambda = (s_1, \ldots, s_t)$ given a sequence of observations $\sigma_k = o_1, o_2, \ldots, o_t$ and model parameters θ can be computed from the joint.

$$P(\lambda|\sigma_k, \theta) \propto P(\lambda, \sigma_k|\theta) = P(s_1) \left(\prod_{j=1}^{t-1} P(o_j|s_j) P(s_{j+1}|s_j) \right) P(o_t|s_t)$$

where the constant of proportionality $P(\sigma_k)$ can be computed easily with the forward-backward algorithm.

Representing the Body: Forward-backward algorithm

Define the forward variable $\alpha_t(i) = P(q_t = i, o_1, o_2, \ldots, o_T|\theta)$, where q_t is the state of the HMM at time t and T is the total number of observations. Similarly, the backward variable $\beta_t(i) = P(o_{t+1}, \ldots, o_T|q_t = i, \theta)$ and so have the recursions:

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right]$$

$$\beta_t(j) = \sum_{i=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

where a_{ij} is the transition probability from state i to j, π_i and b_is are the initial state and observation probabilities, N is the number of states of the HMM.

Representing the Body: Forward-backward algorithm

$$\alpha_1(i) = \pi_i b_i(o_1)$$

$$\beta_T(i) = 1$$

This gives

$$P(\sigma_k) = \sum_{i=1}^{N} \alpha_T(i)$$

Representing the Body

- Wish to evaluate the posterior probability of a sequence of state groups $\lambda_g = (g_1, \ldots, g_t)$ conditioned on a sequence of observations $\sigma_k = (o_1, \ldots, o_t)$.
- We can regard a sequence of state groups as a set of string Λ_g, where a string $\lambda \in \Lambda_g$ if and only if $s_1 \in g_1, s_2 \in g_2, \ldots, s_t \in g_t$.
- Then we have:

$$P(\lambda_g, \sigma_k) = \sum_{\lambda \in \Lambda_g} P(\lambda, \sigma_k)$$

Representing the Body

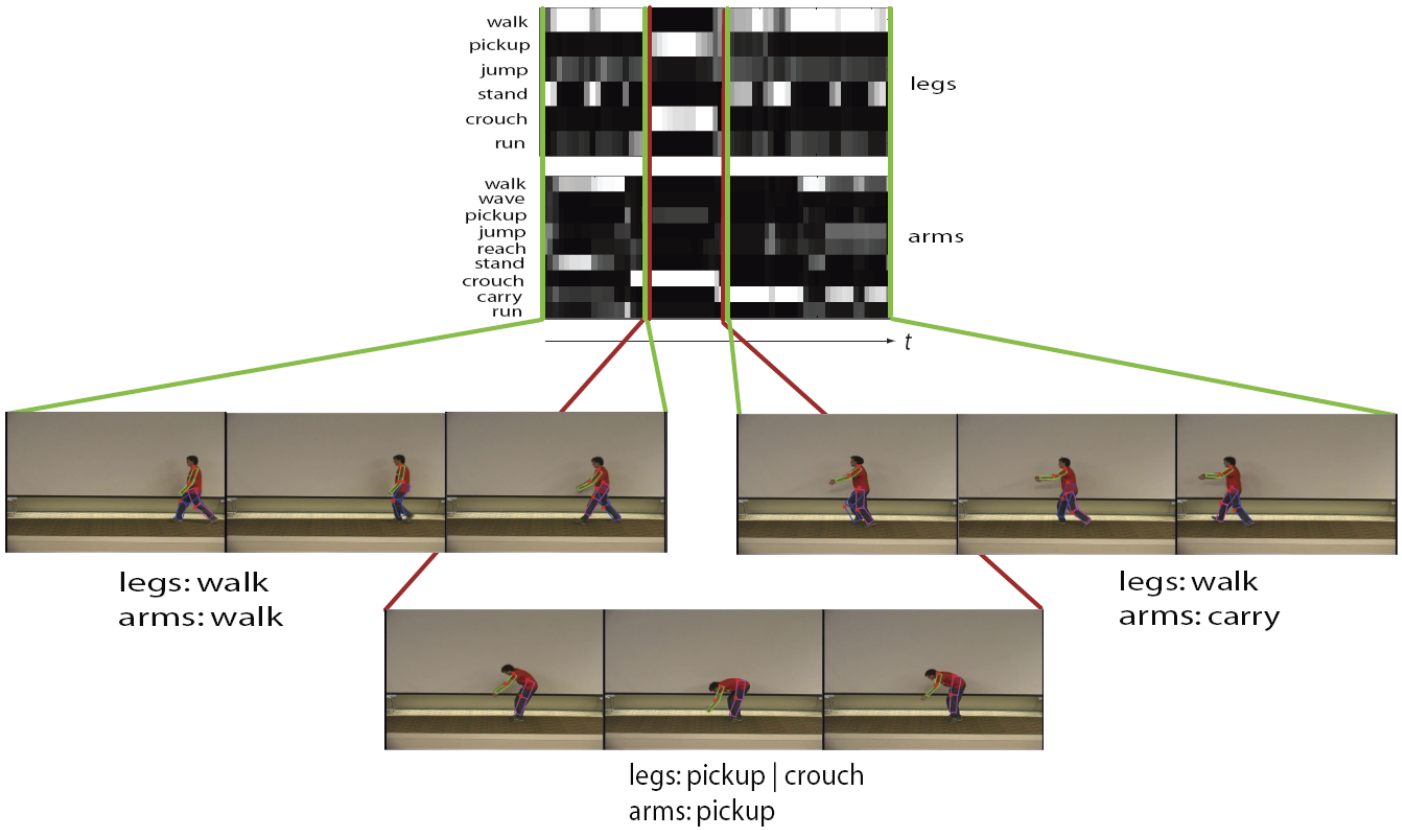

Figure: Example sequence

Querying for Activities

- They would like to be able to build complex queries of composite activities, such as carrying while walking.
- Do not wish to precise about the temporal location of particular activities, but would rather find sequences where there is strong evidence for one activity, followed by strong evidence for another, and wish a little noise scattered about.
- Start by using regular expressions for motion queries.
- An advantage of using RE is that it is easy to compute:

$$\sum_{\text{strings matching RE}} P(\text{string}|\text{frames})$$

Querying for Activities

- They reduce the regular expression to a Finite State Automaton (FSA).
- Then compute thee probability this automaton reaches its final state using a straightforward sum-product algorithm.
- Suppose we want to find videos where the subject is walking and waving his arms at the same time. For legs, we form a walk automaton. For arms, we form a wave automaton. Then simultaneously query both limbs with these automata.

Finite State Representation for Activity Queries

FSA is defined with the quintuple $(Q, \Sigma, \delta, s_0, F)$, where

- Q is the finite non-empty set of states of the FSA;
- Σ is the input alphabet;
- δ is the state transition function where $\delta : Q \times \Sigma \rightarrow Q$;
- s_0 is the (set) of initial states;
- F is the set of final states.

Finite State Representation for Activity Queries

- In the representation, each state $q_i \in Q$ corresponds to the case where the subject is inside a particular action.
- Transitions between states (δ) represent the actions taking place.
- Transitions of the form x^{u_x} means action x sustained for u_x length. (the actions shorter than the unit length do not cause the FSA to change its state.)

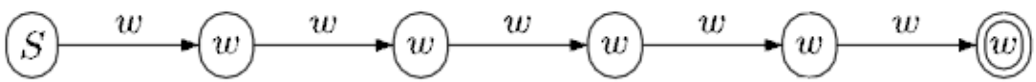

Figure: FSA for a single action

Finite State Representation for Activity Queries

- While forming the FSA, each action is considered to have a unit length u_x.
- A query string is converted to a regular expression, and then to an FSA based on these unit lengths of actions.
- Unit length is based on 2 factors:
  1. the fps of the video;
  2. the action's level of sustainability.

Finite State Representation for Activity Queries

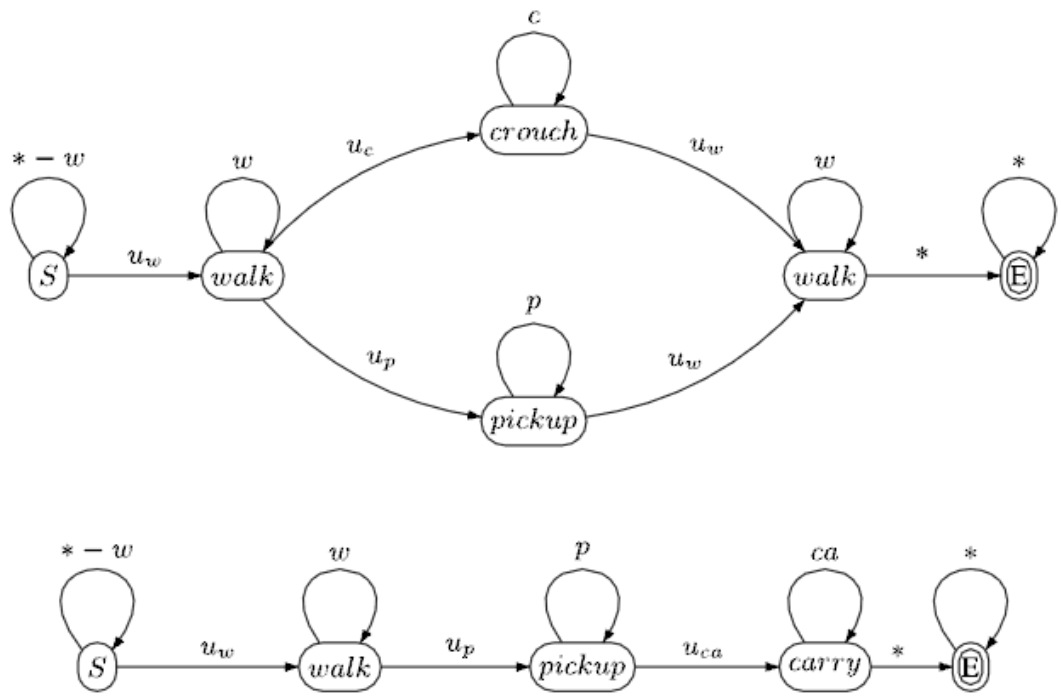

Figure: Query for a video where a person walks, pickups something and carries it.

Querying for Activities

- Now we have an FSA F, and wish to compute the posterior probability of any string accepted by this FSA, conditioned on the observations.
- Write Σ_F for the strings accepted by the FSA.
- Identify the alphabet of the FSA with states – or groups of states – of the model, and get:

$$P(F|o_1, \ldots, o_T, \theta) \propto \sum_{\sigma \in \Sigma_F} P(\sigma, o_1, \ldots, o_T|\theta)$$

where the constant of proportionality can be obtained from the forward-backward algorithm.

Querying for Activities

- Label the states in the FSA with indices $1, \ldots, Q$.
- Compute this sum with a recursion in a straightforward way:

$$\begin{aligned} Q_{ijs} &= P\{\text{a string of length } i \text{ that takes } F \text{ to state } j \text{ and} \\ &\quad \text{has last element } s, \text{ joint with } o_1, \ldots, o_i\} \\ &= \sum_{\substack{\sigma \in \text{ strings of length } i \text{ with} \\ \text{last character } s \text{ that take } F \text{ to } j}} P(\sigma, o_1, \ldots, o_i | \theta) \end{aligned}$$

Querying for Activities

- Write $\mathrm{Pa}(j)$ for the parents of state j in the FSA.
- Write $\delta_{i,s}(j) = 1$ if F will transition from state i to state j on receiving s and zero otherwise.

$$Q_{1js} = \sum_{u \in s_0} P(s, o_1|\theta)\delta_{u,s}(j)$$

and

$$Q_{ijs} = \sum_{u \in \mathrm{Pa}(j)} \delta_{k,s}(j) P(o_i|s,\theta) \left[\sum_{u \in \Sigma} P(s|u,\theta) Q_{(i-1)ku} \right]$$

Then

$$\sum_{\sigma \in \Sigma_F} P(\sigma, o_1, \ldots . o_T|\theta) = \sum_{u \in \Sigma, v \in s_e} Q_{Tvu}$$

Querying for Activities

- Note that nothing major changes if each item u of the FSA's alphabet represents a set of HMM states.
- We must modify each expression to sum states over the relevant group.
- For example, if we write s_u for the set of states represented by the alphabet term u, then we have:

$$Q_{1ju} = \sum_{u \in s_0} \sum_{v \in s_u} P(v, o_1|\theta)\delta_{u,s}(j)$$

and

$$Q_{iju} = \sum_{k \in \mathrm{Pa}(j), v \in s_u} \delta_{k,v}(j) P(o_i|v,\theta) \times \left[\sum_{u \in \Sigma, w \in s_u} P(v|w,\theta) Q_{(i-1)ku} \right]$$

Querying for Activities

- The alphabet from which queries are formed consists in principle of $6^2 \times 9^2$ terms.
- However, the tracker is not sufficiently reliable to give sensible representations of both legs (resp. arms).
- They do not attempt to distinguish between legs (resp. arms), reduce the alphabet to terms where either leg (resp. either arm) is performing an action.
- This gives an alphabet of 6×9 terms.

Experimental Results

- They collected their own set of motions, involving 3 subjects wearing a total of 5 different outfits in a total of 73 movies (15 Hz).
- For viewpoint evaluation, they collected videos of 5 actions; jog, jump, jumpjack, wave, and reach. Each action is performed in 8 different directions to the camera.

Experimental Results

Figure: Single activities with different views

Experimental Results

- Performance over a set of queries is evaluated using Mean Average Precision (MAP) of the queries.
- It is defined as the area under the precision-recall curve for that query.
- A higher average precision value means more relevant items are returned earlier.
- *AveP* over a set S is defined as:

$$AveP = \frac{\sum_{r=1}^{N} (P(r) \times rel(r))}{\text{number of relevant documents in S}}$$

where r is the rank of the item, N is the number of retrieved items, and $rel(r)$ is the binary relevance vector for each item in S and $P(r)$ precision at a given rank.

Experimental Results

For limb activity models:

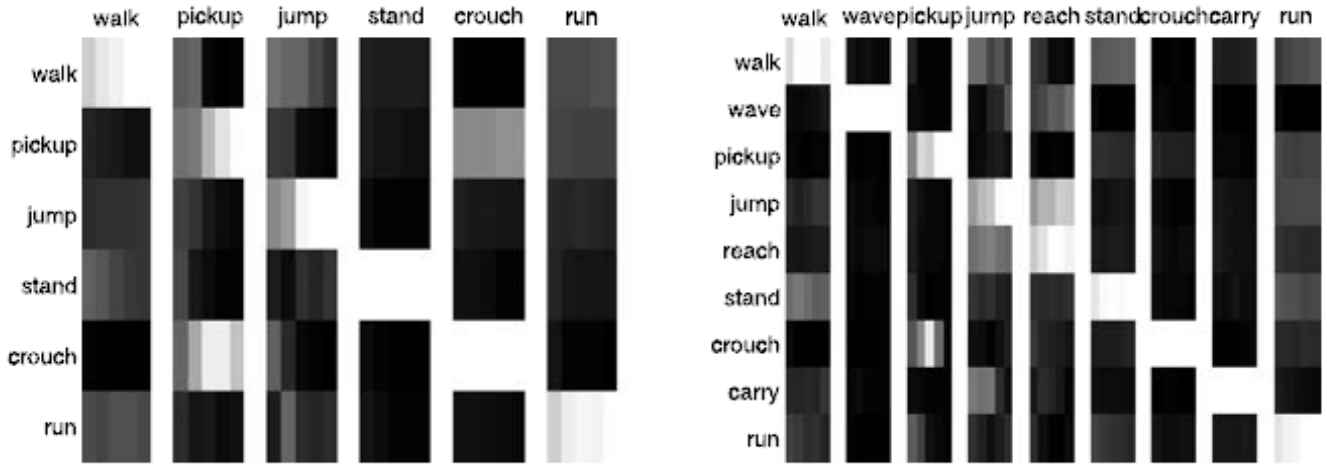

Figure: Posterior for each model

Searching

- Firstly, identify an activity to search for.
- Then mark relevant videos.
- Then write a regular expression.
- Finally, determine the precision and recall of the results ranked by $P(\text{FSA in end state}|\text{sequence})$.

Experimental Results

- Type I: single activities (*walk* for legs and *carry* for arms)
- Type II: 2 consecutive actions like *crouch* followed by a *run*
- Type III: activities consists of 3 consecutive actions (*walk-stand-walk* for legs and *walk-wave-walk* for arms)

Query type	MAP
Type I	0.5562
Type II	0.5377
Type III	0.5902

Figure: MAP values for different types of queries

Experimental Results

Compare with discriminative models:

- Control 1: SVM Classifier over 2D Tracks (MAP value: 0.3970)
- Control 2: SVM Classifier over 3D Lifts (MAP value: 0.3963)
- Control 3: SVM Classifier over 3D Motion Capture Set (MAP value: 0.3538)

Experimental Results

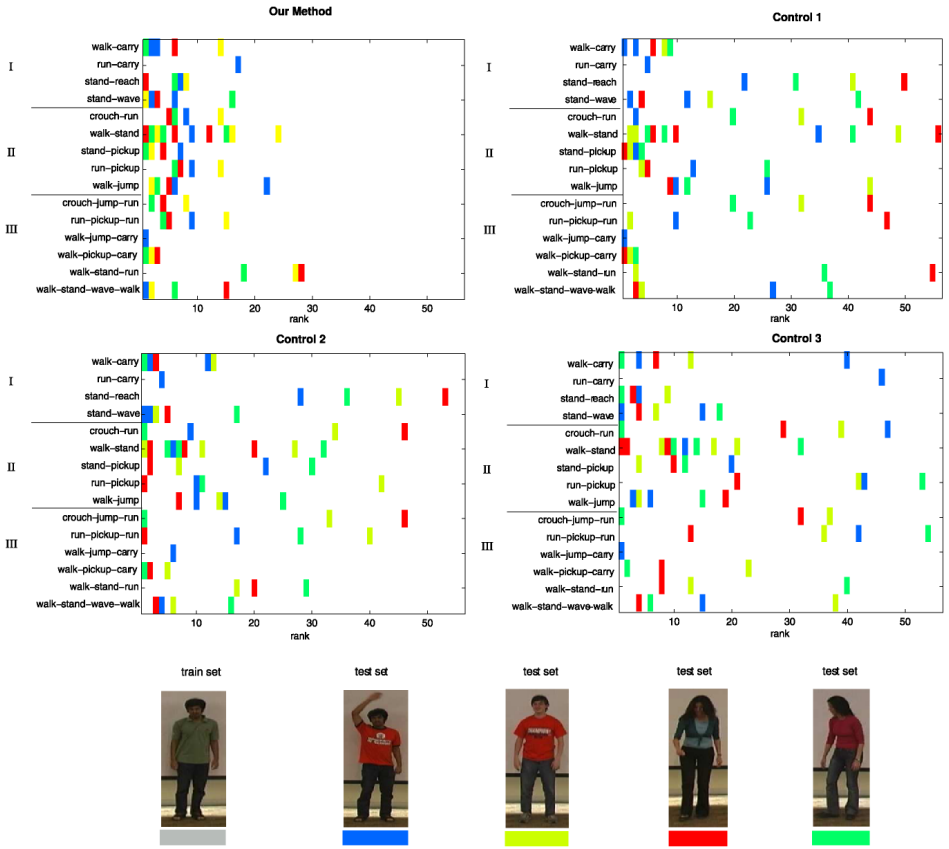

Figure: Results for complex queries

Experimental Results

More experiments I do not mention in this presentation. :)

- Viewpoint Evaluation
- Experiment with Complex Backgrounds
- Vector Quantization for Action Dynamics

Discussions and Conclusion

- What they have done:
  - Representation of human motion that can be used to query for complex activities;
  - Build their own queries using FSA and write separate queries for each limb;
- By joining models of atomic actions to form activity models, they perform minimum parameter estimation.
- It can be thought as an instance of transfer learning; they transfer the knowledge they gain from 3D motion capture data, to 2D everyday activity data.

Discussions and Conclusion

- No example activity is required to formulate a query.
- Dynamics transferred from motion capture domain to real world domain helps in retrieval of complex activities.
- Their representation is in 3D; so, they do not need to retrain the models separately for each viewing direction.
- Big difficulty is to track properly!