Human Action Segmentation via Controlled Use of Missing Data in HMMs

Patrick Peursum† Hung H. Bui⋆ Svetha Venkatesh† Geoff West†
†Dept of Computing, Curtin University of Technology GPO Box U1987, Perth, Western Australia
⋆Artificial Intelligence Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, USA
†{peursump, svetha, geoff}@cs.curtin.edu.au, ⋆bui@ai.sri.com

Abstract

Segmentation of individual actions from a stream of human motion is an open problem in computer vision. This paper approaches the problem of segmenting higher-level activities into their component sub-actions using Hidden Markov Models modified to handle missing data in the observation vector. By controlling the use of missing data, action labels can be inferred from the observation vector during inferencing, thus performing segmentation and classification simultaneously. The approach is able to segment both prominent and subtle actions, even when subtle actions are grouped together. The advantage of this method over sliding windows and Viterbi state sequence interrogation is that segmentation is performed as a trainable task, and the temporal relationship between actions is encoded in the model and used as evidence for action labelling.

1 Introduction

From a machine understanding perspective, it can be useful to consider human body motion as a hierarchy of events ranked by complexity, where lower levels contain shorter motions (dubbed *actions* in this paper) that combine temporally to form higher level events (*activities*) which are more abstract. Most research has focused on developing techniques that can reliably classify an isolated event in its entirety given a set of possible events [1, 5, 6].

Only a few researchers have attempted to automatically segment an event into its component sub-events [9, 2, 3]. Segmentation is desirable since it allows the activity to be examined in finer detail, such as determining exactly when an actor manipulates an object in order to localise the position of that object. However, most of these attempts have relied on a sliding window or hidden state labelling, both of which can be noisy and lead to unreliable labelling. As an alternative approach, this paper proposes the use of Hidden Markov Models (HMMs) [10] modified to perform training and inferencing in the presence of missing data in the observation vector. During training, the multinomial observation vector contains two types of features: data extracted from a simple 3D pose estimation of the actor [4, 7] and data representing the labels for each sub-action (where each label is a boolean flag). When testing against an unseen sequence, the action labels are regarded as missing data and the most probable action label is inferenced by the modified HMM based on the actor's motions and position in the sequence.

The significance of this approach is that it is able to classify and segment the actions in an activity without resorting to heuristics such as sliding windows or labelling states in the Viterbi sequence. Furthermore, the temporal ordering of actions is encoded into the model, assisting in classification for events that are visually similar but temporally distinct. This temporal evidence also facilitates the classification of subtle actions, testing the limits of the approach.

2 Related Work

Pinhanez and Bobick [9] were among the first to propose decomposing an activity into its sub-actions and use the temporal relationships between actions to improve action classification. They use the concepts of *Past*, *Now*, *Future* and an associated logical formalism in a network topology to arrange actions temporally. Unfortunately, in lieu of a vision system sophisticated enough to detect sub-action motions, Pinhanez and Bobick were forced to generate synthetic data in order to test their models. Thus their 'sensors' provided perfect or near-perfect action classification.

Others have attempted automated segmentation using a window-driven approach [2, 8], where a small window is used to sample and classify part of the entire activity. The window is incrementally moved through the full sequence to label all frames. This can provide good results, but is highly sensitive to the window size and tends to be noisy.

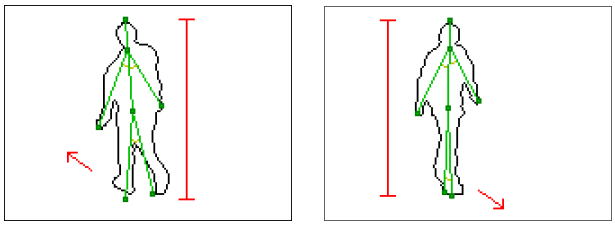
Another method using HMMs is to manually label the actions that the hidden states correspond to [3]. Then during inferencing, the Viterbi state sequence is used to determine which states occurred at what time. However, the hidden nature of HMM states means that they do not lend

Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)
1051-4651/04 $ 20.00 IEEE

themselves well to interrogation and labelling — each state could represent several actions at once or even none at all. Also, segmentation accuracy is limited by the number of HMM states, which is always less than the sequence length.

For this paper, HMMs were chosen as the classifier since they have been proven to be highly successful for human motion modelling and can be modified to handle missing data during both learning and classification [7].

3 Human Motion Capture



a b

Figure 1. "Star" Skeletonisation, showing views of two 2D projections of the same 3D skeleton.

This research extracts usable features from a human silhouette using a simple, fast skeletonisation process originally proposed by Fujiyoshi and Lipton [4] and extended to fuse multiple views into 3D [7]. It detects the gross extremities of the silhouette and assigns limb labels to each extremity (head, arm1, arm2, leg1, leg2). This assumes that each extremity corresponds to a limb of the actor, an assumption that is quite robust as long as the actor is not carrying anything bulky that significantly affects their silhouette.

The skeleton produced is essentially a 'stick-figure' simplification of the person's silhouette (see Figure 1). Features extracted from this skeleton include height, horizontal speed, torso length, torso angle to ground, arm lengths, arm angles to torso, leg lengths and angle between legs.

One unfortunate aspect of the skeleton is that limbs do not always have a corresponding extremity in the silhouette. For example, when the actor's legs come together and form a single extremity, only one 'leg' is detected. This produces an incomplete skeleton where some of the limbs may be missing. Thus to perform action recognition using the skeleton, it is necessary to extend the HMM to handle missing data during both learning and inferencing [7].

4 Action Segmentation and Classification

In order to perform segmentation and classification of actions in an activity, each action is associated with a particular action label flag in the observation vector. During training, the labels are fully observed (from the ground-truth) and the HMM learns an association between the action labels and the motions for that action. When it comes to classification, the labels are not observable and are marked as missing data to allow the HMM to generate its 'best guess' of the labels based on the motions being performed and the sequence of the activity. By taking the most probable action label at each frame, the sequence of actions and their start/finish times can be estimated, segmenting and classifying the actions within an activity at the same time.

The ground-truth segmentation for actions is constructed by hand, with action boundaries rounded to the nearest fifth frame. To some extent, this ground-truth is uncertain since many actions blend smoothly into the next, and some actions even partially overlap. Also, the pose estimation often cannot detect the start and end of a motion (compared to the ground-truth) due to the granularity of pose measurements.

5 Action Label Interrogation

During classification, all action labels are marked as missing. After inferencing the HMM is interrogated as to which missing action label is most probable at each time instant, indicating when each action is most likely occurring.

Missing data exists both for the incomplete pose skeleton and missing action labels. However, action labels are only missing during inferencing and thus this paper will only show the extension to the standard HMM for inferencing. See [7] for details on extending the HMM to learn with missing data.

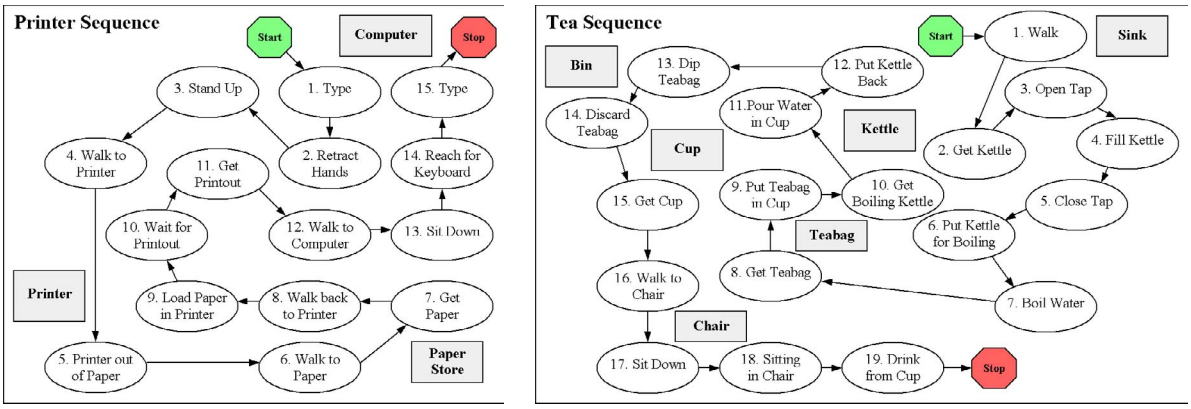
In order to classify a test case, the probability of seeing the test case's observations $y_{1\ldots T}$ is:

$$P(y_{1\ldots T}) = \sum_{i=1}^{Q} P(q_T = i, y_{1\ldots T}) \quad = \sum_{i=1}^{Q} \alpha_T(i)$$

Each y_t is a tuple (y_t^p, y_t^a), where y_t^p is the set of pose features and y_t^a is the set of boolean action labels for each action. Features are assumed independent. q_T is the hidden state q at the last time instant T and $\alpha_T(i)$ is the recursive 'forward variable' for HMMs. The HMM which best fits the observation vector is then chosen as the classification.

If there is missing data in the observation vector, the formulation of $\alpha_t(i)$ must be modified (denoted as $\alpha'_t(i)$ to indicate the distinction). If y_t is missing, $\alpha'_t(i)$ is:

$$\begin{aligned}
\alpha'_t(i) &= P(q_t = i, y_{1\ldots t}) \\
&= P(q_t = i, y_{1\ldots t-1}) \qquad \text{since } y_t \text{ is missing} \\
&= \sum_{j=1}^{Q} P(q_{t-1} = j, q_t = i, y_{1\ldots t-1}) \\
&= \sum_{j=1}^{Q} P(q_t = i | q_{t-1} = j) P(q_{t-1} = j, y_{1\ldots t-1}) \\
\alpha'_t(i) &= \sum_{j=1}^{Q} \left(A_{ji} \cdot \alpha'_{t-1}(j) \right)
\end{aligned}$$



(a) (b)

Figure 2. PRINTER (a) and TEA (b) Activities — action sequence breakdowns represented spatially and temporally. Ellipses indicate actions, with arrows showing the sequencing. Rectangles show the objects being interacted with.

where A_{ji} is the HMM state transition matrix that defines the probability of transitioning from state j to state i. Similarly, the HMM *backwards* recursive variable $\beta'_t(i)$ can be derived for the case where y_t is missing:

$$\begin{aligned}\beta'_{t-1}(i) &= P(y_{t\ldots T}|q_{t-1}=i)\\ &= P(y_{t+1\ldots T}|q_{t-1}=i) \qquad \text{since } y_t \text{ is missing}\\ &= \sum_{j=1}^{Q} P(q_t=j, y_{t+1\ldots T}|q_{t-1}=i)\\ &= \sum_{j=1}^{Q} P(y_{t+1\ldots T}|q_t=j)P(q_t=j|q_{t-1}=i)\\ \beta'_{t-1}(i) &= \sum_{j=1}^{Q}\left(\beta'_t(j)\cdot A_{ij}\right)\end{aligned}$$

For all other times t where y_t is observed, the standard HMM equations for $\alpha'_t(i)$ and $\beta'_t(i)$ must be used:

$$\alpha'_t(i) = B_{ik}\sum_{j=1}^{Q}\left(A_{ji}\cdot\alpha'_{t-1}(j)\right)$$
$$\beta'_{t-1}(i) = \sum_{j=1}^{Q}\left(\beta'_t(j)\cdot B_{jk}\cdot A_{ij}\right)$$

where B_{ik} is the HMM observation matrix that defines the probability of observing symbol $y_t=k$ when in state $q_t=i$.

To determine the most likely missing action label at each time t, it is necessary to calculate the probability that $y_t^a = true$ for each action label a (dubbed $\ell_t(a)$). The most probable label is then chosen as the action for time t.

$$\begin{aligned}\ell_t(a) &= P(y_t^a=true|y_{1\ldots T})\\ &= \sum_{i=1}^{Q} P(y_t^a=true|q_t=i)P(q_t=i|y_{1\ldots T})\\ \ell_t(a) &= \sum_{i=1}^{Q} B^a_{i,k=true}\cdot\gamma'_t(i)\end{aligned}$$

where $\gamma'_t(i)$ is the HMM *forwards-backwards* variable calculated by $\gamma'_t(i) = P(q_t=i|y_{1\ldots T}) \propto \alpha'_t(i)\cdot\beta'_t(i)$ and $\ell_t(a)$ is the probability that action label a is occurring at time t.

6 Results and Analysis

6.1 Experiments

Two types of activity were modelled for this research. The first type involves printing and retrieving a document before returning to the computer, consisting of 15 actions (Figure 2a). The second type is the act of making a cup of tea, which contains 19 actions (Figure 2b). All actions are executed one after the other, sometimes partially overlapping if the actions can be carried out simultaneously (eg: dipping teabag whilst putting the tea kettle down). Thus Bakis-1 strict left-right HMMs [10] are used to model the activities. Note that there are no gaps in the action sequence, hence even 'bridging' actions are modelled such as when the actor must reach out to the keyboard before typing.

To avoid any implicit learning of relative orientations, the scene was arranged in several different ways and training data was taken from all topographies for both activities.

The main difference between the PRINTER and TEA sequences is that the PRINTER sequence contains mostly

Action	Recall	Time (sec)	Mean Err. (frames)	Std Dev. (± frames)
Type	100%	17.6	N/A	N/A
TypeRetract	70%	0.6	3.5	4.80
StandUp	98%	1.3	5.2	5.18
WalkToPrinter	100%	3.2	4.9	5.94
PrinterOutOfPaper	92%	1.8	-2.3	8.99
WalkToPaper	84%	2.2	-2.4	5.73
GetPaper	86%	2.0	-0.4	9.71
WalkToPrinter	88%	2.4	-2.9	9.02
LoadPaper	80%	2.2	2.2	8,76
WaitForPrintout	86%	6.7	9.3	16.56
GetPrintout	84%	1.9	1.5	13.9
WalkToComputer	100%	3.0	-2.5	9.79
SitDown	100%	1.3	0.8	3.37
TypeReach	80%	0.8	2.9	5.67
Type	92%	8.7	-5.8	7.82

(a)

Action	Recall	Time (sec)	Mean Err. (frames)	Std Dev. (± frames)
Walk	100%	2.7	N/A	N/A
GetKettle	100%	2.6	0.4	10.14
OpenTap	94%	2.4	1.4	6.68
FillKettle	94%	7.8	-6.8	10.90
CloseTap	90%	2.1	3.5	6.74
PutKettleBoil	96%	3.1	-0.7	4.78
BoilWater	100%	15.6	-1.8	8.44
GetTeabag	98%	2.3	1.9	6.51
PutTeabagInCup	60%	1.2	-5.1	8.31
GetBoilingKettle	78%	2.0	-2.6	13.71
PourWaterInCup	86%	5.7	-4.0	18.62
PutKettleBack	80%	1.6	5.5	15.32
DipTeabag	96%	6.3	-0.8	5.74
DiscardTeabag	84%	1.9	6.0	7.54
GetCup	80%	1.4	-3.1	15.15
WalkToChair	98%	5.6	-3.8	16.47
SitDown	98%	1.8	-2.7	5.26
Sitting	74%	2.0	-7.7	3.82
Drink	100%	3.4	-16.3	32.30

(b)

Colour Legend: % Well-segmented Actions % Poorly-segmented Actions

Table 1. Segmentation accuracy. PRINTER sequence in (a), TEA sequence in (b). All actions have 50 instances.

prominent actions such as walking, typing and grabbing. In contrast, by including reasonably subtle actions such as putting the teabag in the cup, dipping the teabag and others, the TEA sequence was designed to push the limits of the system (whilst taking into account the shortcomings of the human pose estimation model). Note that some of these actions are also adjacent to other subtle actions. This forces the system to distinguish between subtle actions — if subtle actions were only ever adjacent to prominent actions, it would be impossible to tell if the system was merely detecting the prominent actions and simply allocating the labels for subtle actions in the gaps between.

6.2 Segmentation Analysis

Table 1 shows the accuracy in finding action boundaries when segmenting actions using the proposed technique. Results were generated using 10-fold cross-validation. Recall (true-positive rate) indicates the success in finding the action at all — misclassifications are defined as any segmentation where the centerpoint of the estimated action position does *not* fall within the time that the action actually occurred. Since a few actions are quite long (eg: *WaitForPrintout*), some segmentations were classified as 'correct' even though the boundary detection was significantly worse than the distribution of boundary errors across all 50 instances (distributions were found to be near-normal). Thus any instances with a boundary error more than 2.5 standard deviations from the mean are also considered misclassifications. Across all sequences there were only three events that were completely missed. Given these misclassification, most actions were detected reasonably well, with generally better recall rates for the longer actions. The main exception is *PutTeabagInCup* — its 60% recall can be explained by its very short duration (around one second, thus highly affected by segmentation inaccuracies) and the fact that the transition from the previous event (*GetTeabag*) to *PutTeabagInCup* is quite blurred.

Boundary detection evaluation was performed only on correctly-classified instances. In a few cases at the boundary between action transitions, segmentation would switch to the next action before producing a short burst (2-3 frames) of false positives for the previous action. These bursts were easily filtered out by requiring an action to endure for more than five frames. In general, the mean error in detecting event transitions is quite low, especially considering the ground-truth itself is only accurate to the nearest five frames. Uncertainty in these errors is also fairly good, with most standard deviations being less than 10 frames. The TEA sequence has larger uncertainties than the PRINTER sequence, a result that was expected since the TEA sequence has several subtle actions and the action transitions tend to be more indistinct than in the PRINTER sequence.

	Sequence-based (per-frame)					*Separate Models (pre-segmented)*				
Action	*T.P. Count*	*F.P. Count*	*Actual Count*	*Recall*	*Precision*	*T.P. Count*	*F.P. Count*	*Actual Count*	*Recall*	*Precision*
Type	32203	586	32950	97.7%	98.2%	99	15	100	99.0%	86.8%
TypeRetract	521	422	810	64.3%	55.2%	30	0	50	60.0%	100.0%
StandUp	1329	364	1630	81.5%	78.5%	49	0	50	98.0%	100.0%
Walk	12020	1942	13689	87.8%	86.1%	200	12	200	100.0%	94.3%
PrinterOutOfPaper	1898	257	2340	81.1%	88.1%	16	1	50	32.0%	94.1%
GetPaper	1863	427	2606	71.5%	81.4%	38	63	50	76.0%	37.6%
LoadPaper	2196	1276	2739	80.2%	63.2%	32	12	50	64.0%	72.7%
WaitForPrintout	6984	581	9371	83.4%	92.3%	33	0	50	66.0%	100.0%
GetPrintout	1819	726	2450	74.2%	71.5%	25	19	50	50.0%	56.8%
SitDown	1549	237	1735	89.3%	86.7%	50	0	50	100.0%	100.0%
TypeReach	537	633	1050	51.1%	45.9%	45	11	50	90.0%	80.4%

Colour Legend: % Good accuracy for action % Poor accuracy

Table 2. PRINTER sequence classification accuracy. Note that the four different Walk actions are analysed as one (same for Type)

Statistical hypothesis tests (99% confidence intervals for the true means) were used to identify error means that differed significantly from a zero error (ie: were significantly late or early, in a statistical sense). Most of these significant errors occur because the end of one event is often very similar to the beginning of the next, with the more prominent event usually annexing frames from the other event. For example, in the PRINTER sequence the second *Type* is detected early (-5.8 frames) since it tends to annex frames from the end of *TypeReach* (when the arms are outstretched). For the same reasons, *WaitForPrintout* and *TypeRetract* start late (+9.3 and +3.5). Similar failures occur in the TEA sequence, including *PutTeabagInCup* and *DiscardTeabag*.

Poor uncertainty (and thus poor segmentation) occurs in clusters, possibly because errors propagate from one event to the next until corrected by an event with a strong motion signature. Substantiating this is difficult, requiring more sequences and variety in activity types. However, the pattern does occur in both PRINTER and TEA (see Figure 1, *DipTeabag* and *SitDown* events).

A special case of error is found in the *Drinking* action, which is segmented very early (-16.3) and has an extremely poor uncertainty. This is due to the fact that the pose estimator only detects the arm limb when it has lifted significantly away from the actor's lap, but the ground-truth for the start of drinking is defined about half a second earlier, meaning that the segmentation does not have any evidence for the true start of the *Drinking* action.

6.3 Classification Analysis

For the purposes of evaluating the effect of temporal sequence evidence on classification accuracy, a separate set of HMMs was constructed, one for each individual subaction. This way, no temporal relationships between actions are encoded. The actions are pre-segmented according to the ground-truth for both training and testing of these models, and 10-fold cross-validation was performed. The results from this are compared to the results from Sequence-Based classification in Tables 2 and 3. Note that the Separate-Models classification has the substantial advantage of perfect segmentation, whilst sequence-based classification must perform both segmentation and classification.

Although the Sequence-Based method is on a per-frame basis and the Separate-Models method is on a per-instance basis, it is still possible to get a sense of whether the sequence aids in separating different (though visually similar) actions. As can be seen from Tables 2 and 3, the Separate-Models method has considerable confusion between actions that have similar motions. This includes *GetPaper*, *LoadPaper* and *GetPrintout*, all of which involve the actor reaching out with his hand. Note that *GetPaper* has a high recall but low precision since the other two events are mostly misclassified as *GetPaper*. Conversely, the Sequence-Based method is only outperformed on *TypeRetract* and *TypeReach*, and only because these are very short events, thus the accuracy is very sensitive to any failure in segmentation. Part of the reason for this success lies in the fact that the sequencing ensures that misclassifications can only occur between neighbouring actions.

A similar situation exists for the TEA sequence, with 'grabbing'-style actions such as *GetKettle*, *PutKettle*, *OpenTap*, *CloseTap* and *FillKettle* all confused with one another in the Separate-Models method. *PutTeabagInCup* also has a low accuracy, often incorrectly classified as *PourWaterInCup* since both involve the use of two hands. In comparison, the Sequence-Based method manages to detect *PutTe-*

	Sequence-based (per-frame)					*Separate Models (pre-segmented)*				
Action	*T.P. Count*	*F.P. Count*	*Actual Count*	*Recall*	*Precision*	*T.P. Count*	*F.P. Count*	*Actual Count*	*Recall*	*Precision*
Walk	9911	803	10475	94.6%	92.5%	100	34	100	100.0%	74.6%
GetKettle	4705	1641	5820	80.8%	74.1%	60	7	100	60.0%	89.6%
OpenTap	2489	581	3090	80.6%	81.1%	43	7	50	86.0%	86.0%
FillKettle	9017	781	9820	91.8%	92.0%	50	31	50	100.0%	61.7%
CloseTap	2202	414	2660	82.8%	84.2%	31	8	50	62.0%	79.5%
PutKettle	5075	1339	5981	84.9%	79.1%	66	6	100	66.0%	91.7%
BoilWater	19397	386	19609	98.9%	98.0%	45	2	50	90.0%	95.7%
GetTeabag	2387	409	2920	81.7%	85.4%	40	1	50	80.0%	97.6%
PutTeabagInCup	857	569	1510	56.8%	60.1%	20	19	50	40.0%	51.3%
PourWaterInCup	5446	845	7130	76.4%	86.6%	47	26	50	94.0%	64.4%
DipTeabag	7559	514	7940	95.2%	93.6%	50	29	50	100.0%	63.3%
DiscardTeabag	1506	451	2455	61.3%	77.0%	39	8	50	78.0%	83.0%
GetCup	1141	693	1805	63.2%	62.2%	24	8	50	48.0%	75.0%
SitDown	1823	356	2315	78.7%	83.7%	49	2	50	98.0%	96.1%
Sitting	1491	520	2600	57.3%	74.1%	46	0	50	92.0%	100.0%
Drink	4094	1068	4340	94.3%	79.3%	49	3	50	98.0%	94.2%

Colour Legend: % Good accuracy for action % Poor accuracy

Table 3. TEA sequence classification. Note that the four different Walk actions are analysed as one (same for GetKettle, PutKettle)

abagInCup reasonably well considering that its accuracy is fundamentally flawed by its short duration and difficulty in distinguishing it from the end of *GetTeabag*.

One caveat to note with the Sequence-Based method is that the first and last action of each activity (*Type* for PRINTER, *Walk*, *Drink* for TEA) have an artificially high accuracy since one of their boundaries is the first or last frame and thus that boundary's segmentation is perfect by default.

7 Conclusions

The controlled use of missing data in the observation vector of an HMM to facilitate action labelling presents an elegant, trainable approach to automatically segmenting an activity into its constituent actions without the need for heuristic methods such as sliding windows or interrogating the Viterbi state sequence. This has the added benefit of incorporating the sequence itself as a form of evidence, markedly improving classification of sub-actions that are visually similar but temporally distinct. Even though flat HMMs are not ideal for modelling the hierarchical relationship between activities and actions, the method is still able to segment action boundaries accurately.

References

[1] A. F. Bobick and J. W. Davis. An appearance-based representation of action. In *IEEE International Conference on Pattern Recognition*, 1996.

[2] A. F. Bobick and Y. A. Ivanov. Action recognition using probabilistic parsing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[3] M. Brand and V. Kettnaker. Discovery and segmentation of activites in video. *IEEE Pattern Analysis and Machine Intelligence*, 22(8):844–851, August 2000.

[4] H. Fujiyoshi and A. Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings of Workshop on Application of Computer Vision*, October 1999.

[5] S. Hongeng, F. Brémond, and R. Nevatia. Representation and optimal recognition of human activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–818–I–824, 2000.

[6] S. Park and J. K. Aggarwal. Recognition of two-person interactions using a hierarchical Bayesian network. In *Proceedings of the IEEE International Workshop on Visual Surveillance*, November 2003.

[7] P. Peursum, H. H. Bui, S. Venkatesh, and G. A. West. Technical report 2004/1: Human action recognition with an incomplete real-time pose skeleton. Technical report, Curtin University of Technology, WA Australia, May 2004. http://impca.cs.curtin.edu.au/publications/techreports.html.

[8] P. Peursum, S. Venkatesh, G. A. West, and H. H. Bui. Object labelling from human action recognition. In *IEEE Conference on Pervasive Computing*, pages 399–406, March 2003.

[9] C. Pinhanez and A. Bobick. Human action detection using PNF propagation of temporal constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[10] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.