EURASIP Journal on Applied Signal Processing 2005:13, 2110–2126

Robust Recognition and Segmentation of Human Actions Using HMMs with Missing Observations

Patrick Peursum
Department of Computing, Curtin University of Technology, GPO Box U1987, Perth, Western Australia 6845, Australia
Email: peursump@cs.curtin.edu.au

Hung H. Bui
Artificial Intelligence Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, USA
Email: bui@ai.sri.com

Svetha Venkatesh
Department of Computing, Curtin University of Technology, GPO Box U1987, Perth, Western Australia 6845, Australia
Email: svetha@cs.curtin.edu.au

Geoff West
Department of Computing, Curtin University of Technology, GPO Box U1987, Perth, Western Australia 6845, Australia
Email: geoff@cs.curtin.edu.au

Received 30 December 2003; Revised 17 August 2004

This paper describes the integration of missing observation data with hidden Markov models to create a framework that is able to segment and classify individual actions from a stream of human motion using an incomplete 3D human pose estimation. Based on this framework, a model is trained to automatically segment and classify an activity sequence into its constituent subactions during inferencing. This is achieved by introducing action labels into the observation vector and setting these labels as missing data during inferencing, thus forcing the system to infer the probability of each action label. Additionally, missing data provides recognition-level support for occlusions and imperfect silhouette segmentation, permitting the use of a fast (real-time) pose estimation that delegates the burden of handling undetected limbs onto the action recognition system. Findings show that the use of missing data to segment activities is an accurate and elegant approach. Furthermore, action recognition can be accurate even when almost half of the pose feature data is missing due to occlusions, since not all of the pose data is important all of the time.

Keywords and phrases: human motion analysis, action segmentation, HMMs.

1. INTRODUCTION

The goal of this work is to develop and analyse a method to automatically segment and classify subactions in a trainable manner within a typical indoors environment.[1] This paper considers motion arranged into a two-level hierarchy of events ranked by complexity, where the lower level contains shorter motions (dubbed *actions*) that chain together to form higher-level events (*activities*) which are longer and more abstract. The challenge is to segment a higher-level activity into its constituent subactions. This is desirable since it allows the activity to be examined in finer detail, such as determining exactly when an actor manipulates an object so that the position of the object can be localised [3].

In order to perform limb-level action recognition, it is necessary to extract features from an estimation of the human actor's pose. However, pose estimation in realistic environments will inevitably suffer from the problem of incomplete data. For example, a person's limbs will often be lost due to self-occlusions, occlusions by scene objects, imperfect silhouette segmentation, and other sources of error. In the past, researchers have worked around the problem of incomplete pose data by either using simple features that are always observable [4, 5, 6] or estimating the position of self-occluded limbs by using fully articulated human body models [7, 8, 9]. Fully articulated models can solve the problem of self-occlusions since the occlusions are being produced by a modelled object—the human body. Unfortunately, these techniques are computationally expensive and cannot address causes of incomplete data such as occlusion by scene

[1] Some ideas presented in this work have also appeared in [1, 2].

objects (which are unmodelled) or imperfect segmentation. Thus this paper proposes that missing data should be treated as an integral part of action recognition. An extended version of the EM algorithm for hidden Markov models (HMMs) [10] is used to handle missing data in the observation vector during both training and inferencing. This allows the use of a simple but fast (real-time) pose estimation by skeletonisation [11] that does not attempt to estimate the positions of limbs that it cannot immediately detect. The pose estimation is extended to fuse multiple views into 3D and the modified HMM is then used to demonstrate action recognition with the incomplete 3D pose for manually segmented actions.

The need to manually segment the actions before attempting recognition is itself a problem. It would be far more practical to automatically segment actions from a stream of human motion. This paper achieves such segmentation by inserting action labels into the observation vector and marking these labels as missing (or not) depending on the availability of data. Therefore, the multinomial observation vector contains two types of features: data extracted from the incomplete 3D pose estimation and data representing the labels for subactions (where each label is a boolean flag). During training, the action label data is available from the ground truth and so is fully observed (i.e., not missing). When testing against a new activity sequence, the action labels are not available and so are regarded as missing data. The most probable action label is then inferenced by the modified HMM based on the actor's motions and temporal position in the sequence.

The significance of this paper is in the use of missing data to facilitate action recognition and perform automated action segmentation. An extension to the EM algorithm to handle missing observation data in a discrete HMM is presented. This modified HMM is used to label the actions in an activity without employing a sliding window and without resorting to the less-than-ideal solution of approximating the actions from the Viterbi state sequence. Furthermore, the fact that subactions within an activity are often highly ordered is taken advantage of by encoding the temporal ordering of the actions into the model to assist in segmentating and classifying actions. In essence, temporal ordering is a context that provides two benefits: it assists in separating different actions whose motions are visually similar, and it facilitates the classification of subtle actions whose motion signatures are difficult to detect without guidance.

The rest of this paper is organised as follows. First, a brief review of related literature in this field is given. Then Section 3 describes the pose skeletonisation technique, followed by Section 4 discussing the extension of EM for HMMs with missing observation data. Section 5 details the investigation into the capability of using the incomplete pose for recognition tasks. Section 6 presents the results and analysis for the proposed trainable approach to automated action segmentation. The main limitations of the method are discussed in Section 7. Finally, conclusions are presented in Section 8.

2. RELATED WORK

Until recently, research in action recognition has been limited to using simple features, which limits motion analysis to simple activities or gestures. Examples include trajectory analysis [6, 12], bounding box statistics [3], position of flesh-coloured areas [5, 13], and fixed-orientation pose data [4, 14, 15]. However, more detailed models of human pose have also begun to be used in motion analysis in order to gain information on limb positioning. This has taken several forms, including fully articulated models from silhouettes [7, 9], exemplar matching [8], skeletonisation techniques [11, 16], and other limb-finding methods [17].

These feature sets have generally been *complete* at every frame, that is, all elements of the feature set are observed at every frame. If any incomplete data occurs (such as due to occlusion), it is normally assumed that the pose estimator resolves the missing information, either through filtering or interpolation. If simple, always-observable features are chosen, this assumption is automatically valid since these features are robust to noise or occlusions. For more complex body models, the fact that the recognition algorithms commonly in use do not handle missing data in their standard forms often means that complete feature sets must be provided by the pose estimator. In contrast, this paper shows that handling missing data in the recognition system can allow the use of a computationally efficient method of pose skeletonisation [11] for action recognition even though the skeleton is often incomplete.

For action recognition, hidden Markov models were chosen due to their success in modelling human motion [4, 5, 18] and the fact that they can be extended to handle missing data. HMMs have proven to be useful in action recognition since they are trained on exemplar data, are specifically built to deal with uncertainty, and are generally duration-independent. Crucially, HMMs can also be modified to allow for missing data during *both* training and classification, a facility that has been used successfully in speech recognition to handle short sections of inaudible speech [19]. It is thus possible to use HMMs in order to both model an activity and segment that activity into its constituent actions by using action labels in the observation vector. The probabilistic nature of HMMs means that they are well suited to finding action boundaries even when those boundaries are not well defined, as is the case with most real-world activities. Moreover, the temporal sequence of actions can be modelled in an HMM by defining a similar sequence in the transitions between the hidden states. These temporal relationships then become a context that assists in action recognition.

Among the first to investigate the concept of using sequence as a context to improve action classification were Pinhanez and Bobick [20]. They defined a logical formalism consisting of *Past*, *Now*, and *Future* to specify the temporal relationship between actions, and used this to show that temporal information can significantly assist action recognition. Unfortunately, they did not have a vision system that was sophisticated enough to actually detect the actions they modelled. Hence they were forced to generate synthetic data to

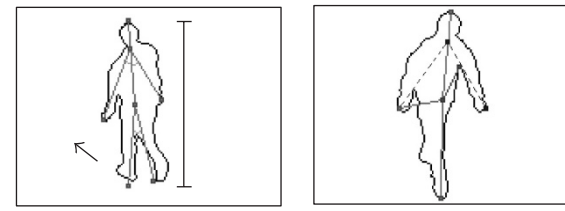
demonstrate the benefits of temporal knowledge, with actions being segmented perfectly or near-perfectly by their synthetic "sensors."

As a means of automatically segmenting an activity, one of the most popular methods is to use a sliding window [3, 5, 21]. This approach incrementally slides a small window across the entire activity, classifying each subset of frames that fall within the window. By moving the window one frame at a time, a profile of the action labels across the entire activity can be built up and used to determine action boundaries. Although this is a simple method that can provide good results, it has several disadvantages. The first is having to choose an optimal window size, which typically differs between each type of action and thus requires a normalising factor in order to compare the likelihood of different action models. Second, classification results tend to be highly sensitive to the exact choice of window size(s). Lastly, since each window of frames is classified independently of one another, labelling tends to be noisy and no temporal evidence is taken into account unless additional constraints are put on top of the sliding window.

Another segmentation method that uses HMMs is to heuristically determine which states generally relate to which actions [18] and then use the Viterbi algorithm to compute the most likely state sequence (and thus by implication, the most likely action sequence). Viterbi segmentation is very similar to the method proposed in this paper in that they both implicitly incorporate evidence of temporal sequence due to the HMM forward-backward algorithm for inferencing (which both filters and smooths). However, the Viterbi method interrogates the *states* to find segmentation boundaries. Using the states for segmentation is a less-than-ideal solution since HMM states are by definition hidden and so they are not guaranteed to relate to any particular observable phenomena. Additionally, the task of associating states with actions must be done manually after training, and involves some guesswork on the part of the researcher. Furthermore, HMM states are not very fine grained and can represent more than one action. For example, states on the boundary between two actions sometimes represent both actions, and the transition from one action to another can be blurred by these boundary states. This undermines the potential segmentation accuracy when using the Viterbi state sequence. In contrast, this paper is proposing the interrogation of the *observations* to find action boundaries, and thus can provide segmentation accuracy at the frame level.

3. POSE SKELETONISATION

This research employs a simple, real-time pose estimation via "star" skeletonisation originally proposed by Fujiyoshi and Lipton [11]. They generated the skeleton by extracting the human silhouette from the video using background segmentation [22] and finding the gross extremities of the silhouette's boundary, where extremities should correspond to the limbs and head. Extremities are found by taking the distance of each point on the boundary to the centroid and smoothing the resulting curve. Smoothing is done by taking the discrete fast Fourier transform of the curve, retaining only the lowest few frequency descriptors, and transforming back (although any method of smoothing could be used). Local maxima on this smoothed curve indicate where extremities exist. The "star" skeleton is formed by joining the extremities to the centroid.



(a) (b)

Figure 1: "Star" skeletonisation, with (a) showing a 2D projection of the 3D skeleton. Features are height, horizontal speed, torso length, torso angle to ground, arm lengths, arm angles to torso, leg lengths, and angle between legs. (b) A case where using the centroid as the anchor will incorrectly detect the armpit as an extremity (dashed lines show correct extremities as detected using the shoulder as the anchor).

A flaw in the skeletonisation is the use of the centroid as the "anchor" point for the distance to the boundary in finding limb extremities. In the case where the actor's arm is directed downwards, the algorithm will find that the *armpit* is an extremity rather than the hand (see Figure 1b). This problem occurs because the centroid-shoulder-hand points are not always "in line," often curling back such as when the arms are hanging downwards and creating a false limb extremity at the curl point (armpit). In contrast, the legs do not experience this problem since the centroid-hip-foot points are always "in line." Since Fujiyoshi and Lipton only analysed leg motions, they did not encounter the skeleton's arm issues. As a solution to the problem, this paper uses a "shoulder" point as the anchor point instead of the centroid. This is because the shoulders are nearly always "in line" with all four limb extremities (hands, feet) and their associated torso connection points (hip, shoulders) due to the structure of the human body. To find this "shoulder" point, the head extremity must first be found so that a torso vector can be defined which extends from the head to the centroid. The shoulder is then defined as a point one-third of the way down from the head along the torso vector. For this research, the head is always the topmost extremity, so finding the head is done by an initial pass using the centroid to find extremities, then choosing the top-most extremity as the head and discarding the rest (they will be superseded by extremities found using the shoulder as the anchor point).

To produce a pose that is independent of the orientation of the actor relative to the cameras, this paper combines the "star" skeletonisation obtained from multiple camera views into 3D world coordinates by finding correspondences

between views. Four cameras are mounted in the ceiling of the room, one in each corner. Correspondences in the skeleton are found via a "best-first" approach, where all possible combinations of correspondences are generated. Any combination with members greater than 25 cm (real-world distance) apart are removed to minimise the search space. From the remaining correspondences, the combination which minimises the average distance between all points is chosen and the points making up this combination are removed from all other combinations to eliminate them from further consideration in later iterations. The process is repeated until no more correspondences between views exist. Note that having one point in one view is not considered a correspondence since there is no evidence from other views that the extremity is an actual limb and not merely a product of noise from background segmentation.

Candidates for arm and leg extremities are then chosen heuristically depending on their length and height from the floor. As mentioned, the head is already allocated and is the extremity with the maximum height above the floor. The two lowest extremities that are both below 30 cm are considered the legs. From the remaining extremities, the two longest extremities are chosen as the arms. Note that this means that one or more of the arms and legs might not be detected, depending upon how many extremities are found (this is the missing data that the action classifier must handle). No attempt is made to determine the "left" from the "right" arm—they are simply "Arm 1" and "Arm 2" (and similarly for the legs). Finally, the skeleton is arranged such that the arms are attached to the shoulder and the legs are attached to the centroid in order to make a stick-figure skeleton, as shown in Figure 1a. These heuristics work well for the purposes of this research since the actor is never hunched over, lying down, or with arms raised above the head. Such postures would cause pose failures since they break the assumption that the head is the topmost extremity.

The number of Fourier descriptors retained affects how sensitive the system is in finding extremities. It is necessary to take into account the silhouette size (which varies depending on the distance of the person to the camera), since larger silhouettes have more detail and thus require fewer Fourier descriptors to find the same extremities. By visually inspecting the results, it was found that retaining 15 descriptors for a silhouette with 256 points struck a good balance between minimising false (nonlimb) extremities whilst still finding true limb extremities reasonably well. Similarly, 12 and 9 descriptors were found to be reasonable for silhouettes with 512 and 1024 points, respectively. Intervening numbers of points were not considered since the fast Fourier transform requires 2^n points in the silhouette (achieved by interpolation).

4. EXTENSION OF EM FOR DISCRETE HMMS WITH MISSING DATA

This section presents the extension of the expectation-maximisation (EM) algorithm for an HMM to allow missing data in the observation vector. For ease of presentation, it is assumed that each observation consists of a single scalar feature y_t which might be missing at time t. The set of all observations is thus represented by y_{obs}, where obs refers to the set of indices of all nonmissing observations. Although not shown here, the model can be easily generalised to the case where the observation y_t consists of many independent features y_t^f, any of which can be missing at a given time t.

A hidden Markov model [10] can be thought of as a finite-state machine whose discrete states are not directly observable (see Figure 2). Whilst the states are not directly observable, there are features that can be measured to give an indication of the current state (or in terms of the generative view of HMMs, the state produces observable, albeit noisy, features). These features are referred to as the observations, and the features across all time instants of a particular example sequence makes up the observation vector of that sequence. The states themselves transition from one state to another at each time instant, including self-transitions (meaning that the state remains unchanged). Both the observations and state transitions are assumed to be independent, and identically distributed (i.i.d.) values across time. That is, the probability distribution for transitivity from one state to the next is the same regardless of which time instant is occurring, and similarly for state observations. Thus an HMM that has discretely-valued observations can be defined by just three parameters:

(i) A_{ij}—probability of transitioning from state i to j,
(ii) B_{ik}—probability of observing symbol k in state i,
(iii) π_i—probability that the first state of the sequence is i.

The sufficient statistics for these parameters are

$$\begin{aligned} \text{SS}(A_{ij}) &= \sum_{t=2}^{T} \delta(q_{t-1} = i, q_t = j), \\ \text{SS}(B_{ik}) &= \sum_{t=1}^{T} \delta(q_t = i, y_t = k), \\ \text{SS}(\pi_i) &= \delta(q_1 = i), \end{aligned} \tag{1}$$

where $\delta(\cdot)$ is a "delta" or "selector" function, that is, 1 if and only if the argument condition is true, and 0 otherwise. These all have natural interpretations. If the states could be observed, A_{ij} is proportional to the number of times that the HMM has moved from state i at time $t-1$ to state j at time t. The other parameters are similarly interpretable.

Since the states are not directly observable in an HMM, these parameters must be estimated using expectation-maximisation (EM). Thus the expectation of these sufficient statistics (ESS) must be taken, given that it is only possible to observe the feature(s) y. Since this research must allow for the possibility of missing observation data, the standard HMM formulation is not sufficient. The ESS must be derived with consideration for missing data by denoting the

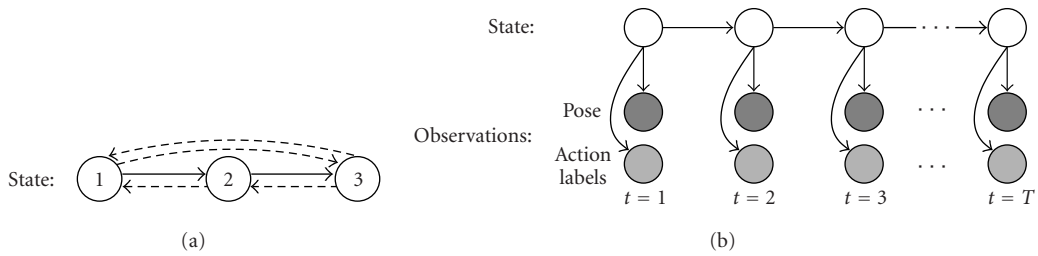


Figure 2: Two ways to view a hidden Markov model. The finite-state machine view (a) shows each state as a separate node and the state transitions between them, but cannot depict the observations. Solid lines indicate the strict left-right transitions that are used in this research—dashed transitions are disallowed (zero probability). The Bayesian network view (b) shows the HMM unrolled across time with the states "generating" observable features at each time instant. States are modelled with a single node whose *value* represents the currently active state, thus the state transitions cannot be explicitly shown.

observation vector as y_{obs}, where obs $\subset \{1, 2, \ldots, T\}$, the set of all observations available. Thus ESS(A_{ij}) is derived as

$$\begin{aligned}\text{ESS}\,(A_{ij}) &= \text{E}\left[\left(\sum_{t=2}^{T}\delta(q_{t-1}=i,q_t=j)\right)\middle|\, y_{\text{obs}}\right]\\ &= \sum_{t=2}^{T} P(q_{t-1}=i,q_t=j|y_{\text{obs}})\\ &= \sum_{t=2}^{T}\xi'_{t-1,t}(i,j).\end{aligned} \tag{2}$$

Note that $P(q_{t-1} = i, q_t = j|y_{\text{obs}})$ is defined as $\xi'_{t-1,t}(i, j)$ (xi-*prime*) since it differs from the standard HMM ξ variable in that the sequence is only *partially* observed (i.e., we only have y_{obs}, not the full $y_{1,\ldots,T}$). If y_t is missing, $\xi'_{t-1,t}(i, j)$ can be factorised in the following way:

$$\begin{aligned}\xi'_{t-1,t}(i,j) &= P(q_{t-1}=i,q_t=j|y_{\text{obs}})\\ &\propto P(q_{t-1}=i,q_t=j,y_{\text{obs}})\\ &\propto P(y_{\text{obs}\leqslant t-1},q_{t-1}=i)\\ &\quad\times P(y_{\text{obs}>t}|q_t=j)P(q_t=j|q_{t-1}=i)\\ &\propto \alpha'_{t-1}(i)\cdot\beta'_t(j)\cdot A_{ij},\\ \xi'_{t-1,t}(i,j) &= \frac{\alpha'_{t-1}(i)\cdot\beta'_t(j)\cdot A_{ij}}{\sum_{i=1}^{Q}\sum_{j=1}^{Q}\alpha'_{t-1}(i)\cdot\beta'_t(j)\cdot A_{ij}},\end{aligned} \tag{3}$$

where $\alpha'_t(i)$ and $\beta'_t(j)$ are the recursive HMM *forwards-backwards* variables that must also be modified to take into account missing data. $\alpha'_t(i)$ can be viewed as the filtered estimate for the probability of being in state i at time t given all of the observations up to and including time t. Similarly, $\beta'_t(i)$ can be considered the smoothed estimate for the probability of being in state i when taking into account all the observations after time t. Their combination yields the HMM $\gamma'_t(i)$ variable, which is essentially the filtered and smoothed estimate of the state at time t:

$$\begin{aligned}\gamma'_t(i) &= P(q_t=i|y_{\text{obs}})\\ &= \frac{P(q_t=i,y_{\text{obs}\leqslant t},y_{\text{obs}>t})}{P(y_{\text{obs}})}\\ &\propto P(q_t=i,y_{\text{obs}\leqslant t})P(y_{\text{obs}>t}|q_t=i)\\ &\propto \alpha'_t(i)\cdot\beta'_t(i),\\ \gamma'_t(i) &= \frac{\alpha'_t(i)\cdot\beta'_t(i)}{\sum_{i=1}^{Q}\alpha'_t(i)\cdot\beta'_t(i)}.\end{aligned} \tag{4}$$

If y_t is missing, $\alpha'_t(i)$ can be recursively calculated by

$$\begin{aligned}\alpha'_t(i) &= P(q_t=i,y_{\text{obs}\leqslant t})\\ &= P(q_t=i,y_{\text{obs}\leqslant t-1}) \quad \text{since } y_t \text{ is missing}\\ &= \sum_{j=1}^{Q}P(q_{t-1}=j,q_t=i,y_{\text{obs}\leqslant t-1})\\ &= \sum_{j=1}^{Q}P(q_t=i|q_{t-1}=j)P(q_{t-1}=j,y_{\text{obs}\leqslant t-1}),\\ \alpha'_t(i) &= \sum_{j=1}^{Q}(A_{ji}\cdot\alpha'_{t-1}(j)).\end{aligned} \tag{5}$$

Similarly, a recursive $\beta'_{t-1}(i)$ with a missing y_t is

$$\begin{aligned}\beta'_{t-1}(i) &= P(y_{\text{obs}>t-1}|q_{t-1}=i)\\ &= P(y_{\text{obs}>t}|q_{t-1}=i) \quad \text{since } y_t \text{ is missing}\\ &= \sum_{j=1}^{Q}P(q_t=j,y_{\text{obs}>t}|q_{t-1}=i)\\ &= \sum_{j=1}^{Q}P(y_{\text{obs}>t}|q_t=j)P(q_t=j|q_{t-1}=i),\\ \beta'_{t-1}(i) &= \sum_{j=1}^{Q}(\beta'_t(j)\cdot A_{ij}).\end{aligned} \tag{6}$$

For all other times t, where $y_t = k$ is observed, the standard HMM equations for $\xi'_{t-1,t}(i,j)$, $\alpha'_t(i)$, and $\beta'_t(i)$ are used:

$$\xi'_{t-1,t}(i,j) = \frac{\alpha'_{t-1}(i)\cdot B_{jk}\cdot \beta'_t(j)\cdot A_{ij}}{\sum_{i=1}^{Q}\sum_{j=1}^{Q}\alpha'_{t-1}(i)\cdot B_{jk}\cdot \beta'_t(j)\cdot A_{ij}},$$
$$\alpha'_t(i) = B_{ik}\sum_{j=1}^{Q}\left(A_{ji}\cdot \alpha'_{t-1}(j)\right), \tag{7}$$
$$\beta'_{t-1}(i) = \sum_{j=1}^{Q}\left(\beta'_t(j)\cdot B_{jk}\cdot A_{ij}\right).$$

The important difference between (3), (5), (6), and (7) is that (7) must take into account the fact that y_t is observed. This results in having to multiply by B_{ik} (which is the probability of observing the symbol k when in state i). In contrast, when y_t is missing, there is no multiplication by B_{ik}—see (3), (5), (6). This is because the feature is not observed, so there is no evidence to influence the current estimate of the state (represented by $\alpha'_{t-1}(i)$ and $\beta'_t(j)$). However, note that it is still necessary to calculate the effect of a state transition via multiplying by A_{ij}. If the first observation y_1 is missing, $\alpha'_1(i)$ is

$$\begin{aligned}\alpha'_1(i) &= P(q_1 = i, y_{\text{obs}\leqslant 1} = k)\\ &= P(q_1 = i) \quad \text{since } y_1 \text{ is missing,}\\ \alpha'_1(i) &= \pi_i.\end{aligned} \tag{8}$$

Otherwise, the standard HMM initialisation for $\alpha'_1(i)$ holds:

$$\begin{aligned}\alpha'_1(i) &= P(q_1 = i, y_{\text{obs}\leqslant 1} = k)\\ &= P(y_{\text{obs}\leqslant 1} = k | q_1 = i)P(q_1 = i),\\ \alpha'_1(i) &= B_{ik}\cdot\pi_i.\end{aligned} \tag{9}$$

In contrast, $\beta'_T(i)$ is initialised to 1 regardless of whether y_T is observed or not, equivalent to the standard HMM [10]. Note that termination probabilities [23] can also be introduced, and their usage is no different to the standard HMM.

Given (2)–(9), the reestimation equation for $\hat{A}_{ij}$ with missing data can be written in the same form as the standard HMM (albeit using the modified ξ'):

$$\begin{aligned}\hat{A}_{ij} &\propto \text{ESS}(A_{ij}),\\ \hat{A}_{ij} &= \frac{\sum_{t=2}^{T}\xi'_{t-1,t}(i,j)}{\sum_{j=1}^{Q}\sum_{t=2}^{T}\xi'_{t-1,t}(i,j)},\\ \hat{A}_{ij} &= \frac{\sum_{t=2}^{T}\xi'_{t-1,t}(i,j)}{\sum_{t=2}^{T}\gamma'_{t-1}(i)}.\end{aligned} \tag{10}$$

$\hat{B}_{ik}$ can likewise be derived with missing data. Note that the summation over t must be split into two summations: one over the observed data ($t \in \text{obs}$) and another over the unobserved data ($t \notin \text{obs}$):

$$\begin{aligned}\text{ESS}(B_{ik}) &= \text{E}\left[\left(\sum_{t=1}^{T}\delta(q_t = i, y_t = k)\right)\middle| y_{\text{obs}}\right]\\ &= \sum_{t=1}^{T}P(q_t = i, y_t = k | y_{\text{obs}})\\ &= \sum_{t\in\text{obs}}P(q_t = i, y_t = k | y_{\text{obs}})\\ &\quad + \sum_{u\notin\text{obs}}P(q_u = i, y_u = k | y_{\text{obs}})\\ &= \sum_{t\in\text{obs}}P(y_t = k | q_t = i, y_{\text{obs}})P(q_t = i | y_{\text{obs}})\\ &\quad + \sum_{u\notin\text{obs}}P(y_u = k | q_u = i, y_{\text{obs}})P(q_u = i | y_{\text{obs}})\\ &= \sum_{t\in\text{obs}}\delta(y_t = k)\cdot\gamma'_t(i) + \sum_{u\notin\text{obs}}B_{ik}\cdot\gamma'_u(i),\end{aligned} \tag{11}$$

$$\begin{aligned}\hat{B}_{ik} &\propto \text{ESS}(B_{ik}),\\ \hat{B}_{ik} &= \frac{\sum_{t\in\text{obs}}\left(\delta(y_t = k)\cdot\gamma'_t(i)\right) + \sum_{u\notin\text{obs}}\left(B_{ik}\cdot\gamma'_u(i)\right)}{\sum_{t=1}^{T}\gamma'_t(i)}.\end{aligned} \tag{12}$$

Finally, the reestimation for $\hat{\pi}_i$ is

$$\begin{aligned}\text{ESS}(\pi_i) &= \text{E}\left[\delta(q_1 = i)\,|\,y_{\text{obs}}\right]\\ &= P(q_1 | y_{\text{obs}})\\ &= \gamma'_1(i),\end{aligned} \tag{13}$$

$$\begin{aligned}\hat{\pi}_i &= \text{ESS}(\pi_i),\\ \hat{\pi}_i &= \gamma'_1(i).\end{aligned} \tag{14}$$

The HMM parameters are estimated via EM by initialising the three parameters with educated guesses based on the values in the training data and the model required (left-right for this paper). The parameters are then iteratively reestimated using the update equations (10), (12), (14) to improve the estimates of the parameters with respect to the training data. Reestimation ceases when the probability of observing the training data converges to a local maximum.

5. ACTION RECOGNITION WITH INCOMPLETE DATA

This section presents experimental results in classifying persegmented actions from incomplete skeleton data using the HMM with incomplete observation data.[2] It is thus assumed that the training data for each action is manually segmented and extracted. This assumption will be relaxed in section 6.

[2] Some of these results have also appeared as an extended abstract [1].

Table 1: Confusion matrix for classification of actions. Errors are highlighted.

Actions	Classification of Actions						
	Drink	Read	Type	Walk	SitDown	StandUp	Recall
Drink	90	1	0	0	0	0	98.9%
Read	5	55	0	0	0	0	91.7%
Type	0	0	40	0	0	0	100%
Walk	0	0	0	50	0	0	100%
SitDown	0	0	0	0	50	0	100%
StandUp	0	0	0	0	0	50	100%
Precision	94.7%	98.2%	100%	100%	100%	100%	—

5.1. Experiments

Experiments were performed to test whether action recognition can be robust under conditions of incomplete pose data. Actions took place within an indoor laboratory monitored by four cameras (one in each corner) capturing with a resolution of 320×240 at 25 fps. Ten features were extracted from the 3D pose: height, torso length, torso angle, leg lengths, angle between the legs, arm lengths, and angles between the arms and the torso (see Figure 1a). Since the pose estimation does not give the orientation of the person, angles are calculated along the plane spanned by the two limb vectors. Additionally, the horizontal speed of the actor was included. All feature measurements are continuous values, hence the data must also be discretised. These features form the observations of the HMM and are assumed to be independent given the hidden state.

Six HMMs were trained using these features, with one HMM per action performed in the scene: walking, sitting down into a chair, standing up from a chair, typing, reading, and drinking (the latter three all performed whilst sitting). *Reading* involves the actor picking up a book when seated, reading it for a short time, and placing the book back. *Drinking* differs from *Reading* in that the actor picks up a cup and drinks from it before placing it back. *Typing* is the act of typing at a computer keyboard, also whilst seated. All but *Walking* and *Typing* are modelled using Bakis-1 strict left-right HMMs [10] due to the fact that only *Walking* and *Typing* are cyclic in nature. Between 40 and 90 instances of each action were captured, with actions being manually segmented for both training and classification. *Drinking* and *Reading* are modelled with 20 states, *Walking* with 10 states, and all other actions with 15 states. These numbers were empirically found to provide the best classification results.

5.2. Classification results

Classification accuracy was evaluated by performing 5-fold cross-validation to produce the confusion matrix in Table 1. Classification accuracy is quite high with the only failures resulting from confusion between the drinking and reading actions. This confusion is explained by the fact that drinking and reading differ only slightly—drinking involves bringing an object (cup) to the actor's mouth whereas reading involves bringing an object (book) to the actor's body. Note that there are a higher number of misclassified reading instances than drinking instances. Closer inspection of these misclassified reading instances shows that the pose estimation has detected the shoulder of the person as a false limb, confusing the action recognition system into believing that the "limb" is a result of the act of drinking rather than simply being the actor's shoulder.

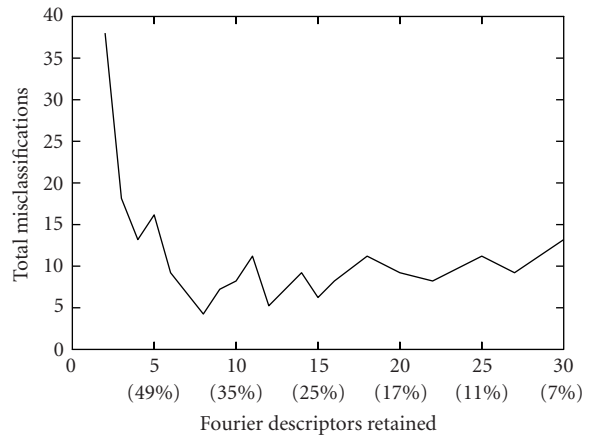
5.3. Effect of missing data

Further analysis was performed to determine how robust the action recognition is to various amounts of missing data. When retaining 15 Fourier descriptors, approximately 25% of the skeleton data is missing on average, due to undetected arms (42% of the time) and legs (26% of the time). This amount of missing data can be altered by changing the number of Fourier descriptors retained—having more descriptors results in less missing data and vice-versa. Note that more descriptors also result in an increase in false extremities that are not actually limbs due to the system's increased sensitivity. Thus a tradeoff exists between missing data and false-positive "limbs."

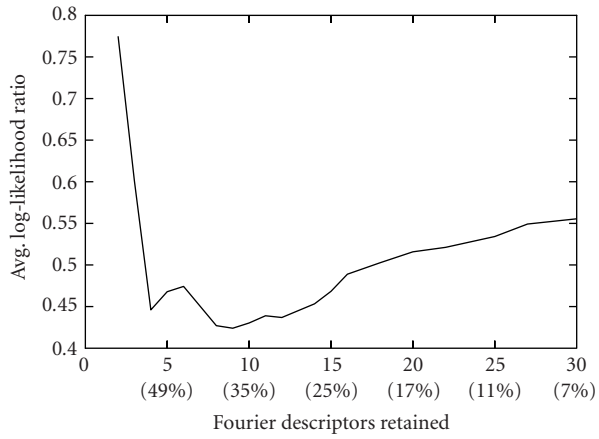
Figure 3 shows the effect on misclassifications when varying the amount of missing data. Note the sudden steep increase in the number of misclassifications that occurs when the number of Fourier descriptors drops to 6 or lower, equivalent to 45% or more of the data missing. Furthermore, since some features (height, speed, torso length, torso angle) are always observable, the proportion of missing data can be considered to be much higher—around 70% of the data is missing for those features that can contain missing values. Although this seems to be an excessively high tolerance of missing data, it can be explained by the fact that only a few movements are important in each action and these are often fairly prominent (e.g., reaching out an arm).

Conversely, when more descriptors are retained the decline in accuracy is not so pronounced. This is because the extra false-positive "limbs" only occur during times when the system does not have an extremity that fits the true limb (false limb extremities almost never take the place of a true limb extremity when both are available). Ideally, the system would indicate the limb is missing since the false limbs are basically extra noise rather than loss of crucial information.

Note that the original choice of retaining 15 Fourier descriptors is not optimal—the best log-likelihood ratios occur when retaining anywhere between 8 and 12 descriptors (40%



(a)



(b)

Figure 3: Effect of missing data on classification accuracy. Bracketed values indicate percentage of missing data. (a) Total number of misclassifications against descriptors kept, and is fairly noisy. (b) is similar except that the y-axis shows the confidence measure of the classification (ratio of the log-likelihood of the correct model and the best of the other models).

and 30% missing data, respectively). Slightly better results are obtained as less descriptors are kept since fewer spurious details are detected, although fewer than 8 descriptors means the misclassification rate increases markedly. This indicates that when around 40% of the data is missing, the number of false-positive limbs detected is minimal (having instead been defined as "missing data"). Any additional missing data then begin to adversely affect the detection of extremities that are actually limbs, hence misclassifications increase rapidly due to the loss of critical information.

6. ACTION SEGMENTATION VIA MISSING DATA

This section examines how controlled use of missing data can enable the segmentation of higher-level activities into their constituent actions, as well as providing benefits for classification of the individual actions.[3] In order to perform segmentation 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 motion features for that action. When it comes to classification, the labels are not observable (since no ground truth is available) and so are marked as missing data. This allows 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, which effectively segments and classifies the actions of an activity.

The ground-truth segmentation for actions is constructed by hand, with action boundaries rounded to the nearest 5th 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 exact start and end of a motion (compared to the ground truth) due to the granularity of pose measurements. For example, when a person reaches out to grasp a cup, the initial stage of the "reaching-out" motion goes undetected by the pose skeleton since the arm is still too close to the body. These issues make it difficult to estimate a suitable (and consistent) boundary point for actions, negatively affecting the accuracy of the system when automatically segmenting test-case activities. However, the statistical nature of HMMs is well suited to dealing with such fuzzy boundaries.

6.1. Experiments

Two types of higher-level activities were modelled for this research—printing a document and making a cup of tea (henceforth referred to Printer and Tea in this article)—see Figure 4 for a breakdown of each activity. The actions in both activities are carried out in a particular order and so are modelled with Bakis-1 strict left-right models [10], where states may only transition to themselves or the next state in the sequence. The Printer activity contains 15 actions and the Tea activity consists of 19 actions. There are no gaps in the sequence of actions, hence even "bridging" actions are labelled, such as when the person retracts their arms from the keyboard just before standing up in the Printer activity. Since each action can have several distinct stages in their progression, it is necessary to have about three or four states per action. Thus empirical tests found that Printer was best modelled with 50 states and Tea with 60 states. Note that the activity sequence (and thus the correct HMM) is assumed to be known before action segmentation occurs—classifying the activity is not performed since this paper concentrates on the problem of segmenting actions from a higher-level activity. Furthermore, the two modelled activities are so dissimilar to each other that an evaluation of activity classification would not provide any significant findings.

[3] Some of the results for segmentation have also appeared in [2].

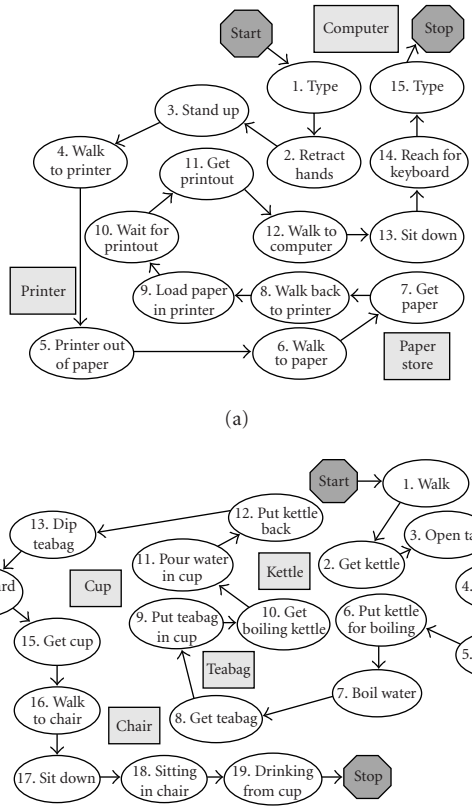


FIGURE 4: (a) Printer and (b) Tea activities—action sequence breakdowns represented spatially and temporally. Ellipses indicate actions, with arrows showing the sequencing. Rectangles show the objects being interacted with. Source: [2].

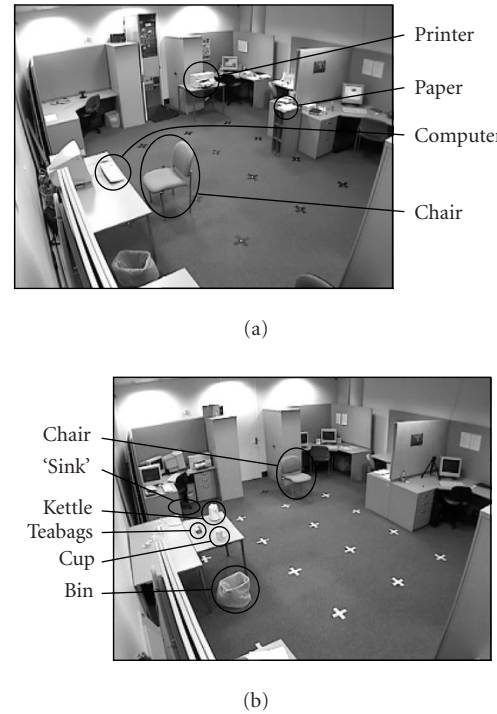


FIGURE 5: Example positions of relevant objects for (a) Printer and (b) Tea activities. In (b), the "sink" is located beside the table, but does not physically exist in the scene.

The Printer activity contains mostly prominent actions such as typing, standing up, and walking. Furthermore, the computer, printer, and paper store are all at different locations within the scene (see Figure 5a for an example). This means that the actions related to each object occur relatively distinct from each other ("connected" by walking actions), and so should be well segmented by the system. Conversely, much of the activity in the Tea sequences occurs in one location—the tea kettle, teabags, and cup are all located on the same table (see Figure 5b). Many of the actions at this location are consequently made up of short, subtle motions, and action boundaries tend to be "blurred" due to the smooth transition between actions. In effect, the Tea activity is designed to test whether the system can segment individual actions out of a cluster of subtle actions. This ensures that the system cannot achieve misleadingly good results simply by finding prominent actions and "filling in the gaps" for the remaining (subtle) actions.

To avoid any implicit learning of relative positions and orientations, the important objects for both Printer and Tea were arranged in various topologies and training data was taken from each of the configurations. This is to ensure that positional or directional information does not artificially provide evidence for segmentation. Note that object position data is *not* a feature for action recognition—only human motion information is used.

6.2. Inferring action labels from the HMM

In order to perform segmentation, it is necessary to query the HMM as to what the most probable action label is at every time instant. This is because during inferencing on an unseen test case, the action labels are unknown and thus marked missing. The probability that a missing action label $y_t^a = \text{true}$ can be calculated as follows:

$$\begin{aligned}\ell_t(a) &= P(y_t^a = \text{true}|y_{\text{obs}}) \\ &= \sum_{i=1}^{Q} P(q_t = i, y_t^a = \text{true}|y_{\text{obs}}) \\ &= \sum_{i=1}^{Q} P(y_t^a = \text{true}|q_t = i)P(q_t = i|y_{\text{obs}}), \\ \ell_t(a) &= \sum_{i=1}^{Q} B_{i,k=\text{true}}^a \cdot \gamma_t'(i),\end{aligned} \tag{15}$$

where $\ell_t(a)$ is the likelihood that the action a is occurring at time t. This probability is calculated for all action labels and the most likely label is used as the label for the time t. By taking the most likely labels across all time instants t, it is possible to build up a profile of the action sequence and segment the activity into its constituent actions.

Occassionally, the system temporarily labels frames incorrectly with the next action's label, before reverting back to the correct label two to three frames later. These short bursts of false positives can be filtered out by mandating that an action must be labelled for at least five consecutive frames before being accepted. An alternative solution to this would be to use a confidence measure to eliminate tentative labels (i.e., action labels that are only slightly more likely than the other actions). However, noise bursts occur very infrequently (less than 2% of all action transitions were noise bursts) and the described five-frame filtering proved to be adequate for solving the problem.

6.3. Evaluation of action detection

Segmentation results were obtained by performing tenfold cross-validation over all 50 sequences of each activity. Evaluation of the segmentation is mostly concerned with determining the accuracy of finding action boundaries. However, it was found that some border detections were significantly wrong, so much so that they are better considered as complete failures in finding the placement of the action within the sequence. Figure 6 shows the percentage of actions whose temporal position was correctly approximated—this percentage is typically known as the recall or true-positive rate. True positives were accepted according to two criteria.

(1) The centrepoint of a segmented action position must fall within the time that the action actually occurred (according to the ground truth).

(2) The error in detecting the start time of a segmented action must not be an outlier with respect to the distribution of errors across all instances satisfying criterion 1. Since the distribution of boundary errors are close to normal for all actions, an outlier was heuristically defined as being greater than 3 standard deviations from the mean.

The rationale for the first criterion is that if the system missed most of the frames that the action actually occurred in, the action has effectively been misclassified and should be portrayed as such. The second criterion ensures that significant errors are still considered misclassifications even when the action duration is very long (in which case, the first criterion is typically passed).

As can be seen from Figure 6, very short actions tend to be more difficult to correctly detect due to the first criterion. The worst performers are *TypeRetract* from the Printer activity and *PutTeabagInCup* from the Tea activity. Besides their short durations (less than one second), detection of these two actions is futher handicapped by the fact that the transitions from their immediate predecessors (*Type* and *GetTeabag*) are not clearly defined. A similar situation exists for other poor performers such as *TypeReach* and *Sitting*. Otherwise, recall rates are quite good at 80% or better.

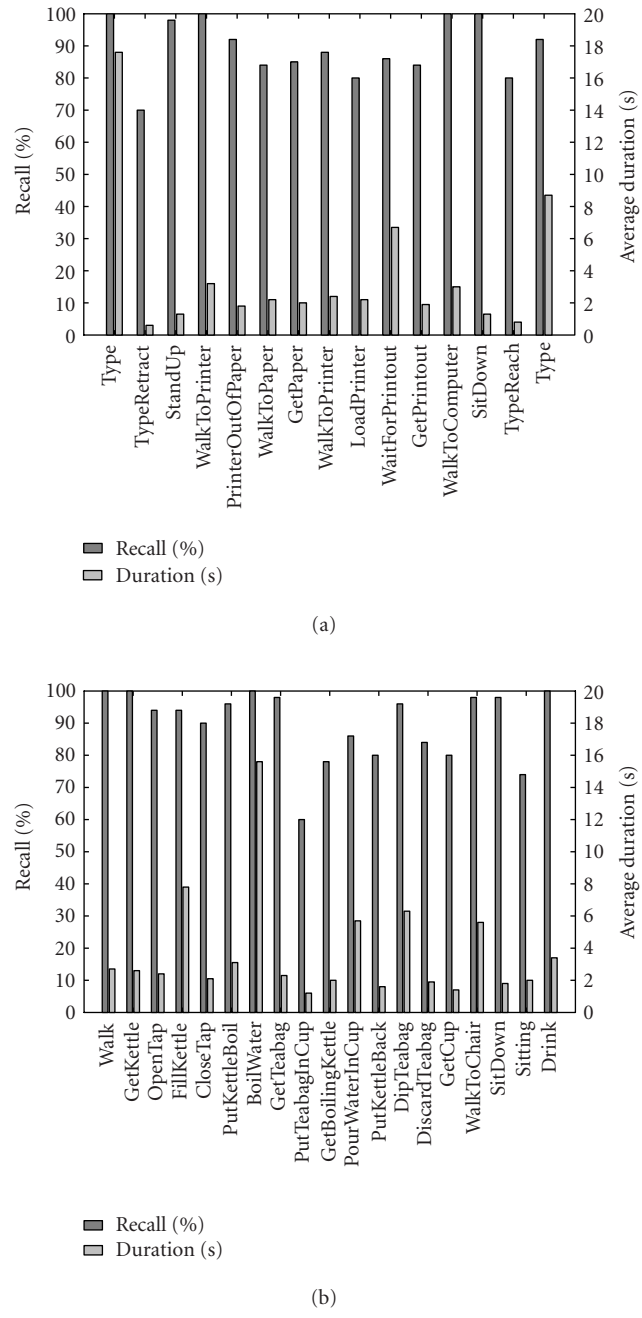
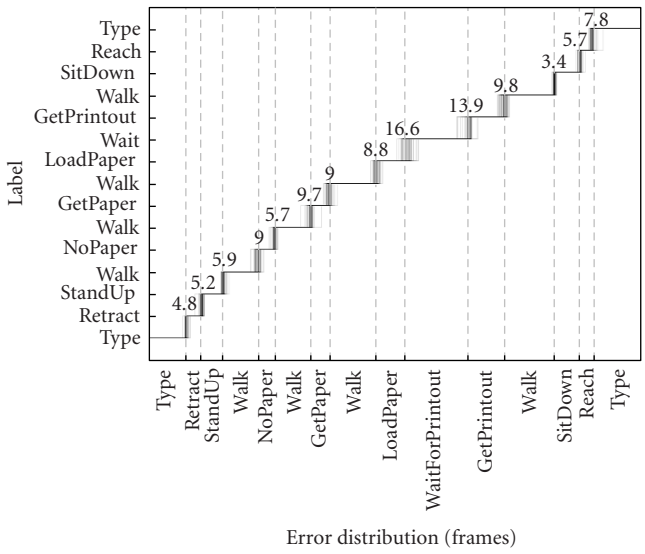


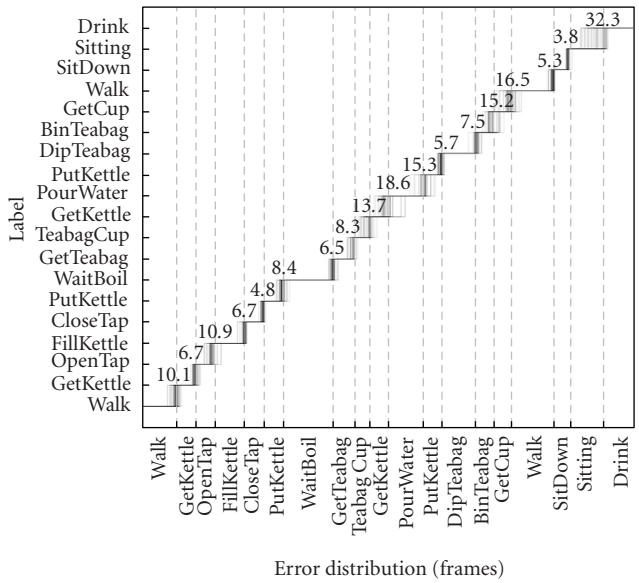
Figure 6: True-positive rates for correctly approximating the temporal position of (a) Printer and (b) Tea subactions. Errors are caused by very short actions (e.g., *TypeRetract*, *PutTeabagInCup*) and actions whose transitions are not clear-cut (e.g., *Sitting*).

6.4. Evaluation of border detection

Border detection was evaluated by considering the distribution of errors in correctly finding the beginning and end of each action (with respect to the ground truth). Figure 7 depicts step plots that show the distribution of errors for all actions in both the Printer and Tea activities. Misclassifications identified in the previous section (Section 6.3) were *not* part of this evaluation since they are considered to have missed detecting the action completely (i.e., they are outliers). Overall, the spread of the error distributions is reasonably small,



(a)



(b)

Figure 7: Step plots showing segmentation accuracy for (a) Printer and (b) Tea for detected actions from Figure 6. Y-axis plots the action labels detected, x-axis plots the distribution of errors versus ground truth for each action label (dashed lines). Each distribution has its standard deviation shown above it (in frames). Deviations from the ground truth are shown to scale in the figure, with 1 pixel = 2 frames, and intensity indicating the density of instances. Action durations are *not* on the same scale and so should not be compared to the error distributions.

given that the ground truth itself is uncertain to within ± 5 frames. As expected, the Printer sequences are generally more accurate than the Tea sequences because the Tea activity contains clusters of subtle actions.

For the Printer activity, uncertainty is quite low and the standard deviations are almost all less than 10 frames. The exception is the subsequence *LoadPaper-WaitForPrintout-GetPrintout-Walk*. The times of transition between these actions is not particularly distinct—the main difference between the actions is whether the actor's arm is extended or not. The precise time when the pose skeleton detects the actor's extended arm can vary depending on how well the silhouette is segmented. This uncertainty adds to the inaccuracy of border detection for these four actions.

In Tea, there is particular difficulty with the subsequence from *TeabagCup* to *PutKettle* since this range corresponds to a cluster of short, subtle hand motions that are taking place within an area of one or two square feet. Difficulties are also encountered at *BinTeabag-GetCup-Walk* due to the fact that these actions can be done in parallel and so tend to overlap. For example, at the time that the actor discards the teabag, he can turn to begin walking as his free hand reaches out to take the cup. The near-simultaneous nature of these three actions means that it is difficult to separate them into consecutive events (which the left-right HMM is forced to do). This same problem also means that the classification rate for these actions is adversely affected (see Figure 6b).

As a further analysis, statistical confidence interval tests for the true means (at 99% confidence) were conducted to identify actions that were consistently detected late or early (in a statistically significant sense). Given that the ground-truth uncertainty is ±5 frames, only mean errors that fell outside of this uncertainty were analysed further to see what went wrong. For the Printer activity, significant errors were found for *TypeRetract* (found late at +5.2 frames on average), *Type* (too early at −5.8 frames), and *WaitForPrintout* (+9.3). Similarly, the Tea activity had errors for *FillKettle* (−6.8), *PutTeabagInCup* (−5.1), *PutKettleBack* (5.5), *DiscardTeabag* (−5.1), *Sitting* (−7.7), and *Drink* (−16.3). Most of these errors occur because the end of one action can be very similar to the beginning of the next. This results in the more prominent event annexing frames from the other event. For example, the second *Type* event is found 5.8 frames too early since the system tends to allocate frames to *Type* from the end of the preceding *TypeReach* action (when the arms are fully outstretched).

A special case of error that cannot be explained with this annexing mechanism is the *Drink* action from Tea. This action is detected very early (−16.3) and has a very high uncertainty (standard deviation of 32.3). The problem here is that the ground truth for the start of the *Drink* motion is defined when the actor begins to lift the cup to his mouth. However, this motion is initially close to the body and so the pose estimation is unable to detect the motion until the actor's arm is considerably higher and away from the body (about half a second too late). In order to compensate, the system is forced to approximate the start of *Drink* as being a number of frames before the pose estimation first detects the motion. The end result is that the system tends to overestimate how much earlier the action starts, thus the chosen boundary is also highly uncertain.

6.5. Benefits of temporal sequence

In order to demonstrate the effectiveness of using temporal sequence as a context for action recognition, it is necessary to classify the actions with and without sequence evidence and compare the two. For comparison, there are two figures used for evaluating classification accuracy—recall and precision.

The recall figure for an action a is the percentage that is correctly labelled as a out of all possible ground-truth instances for a. In contrast, the precision is the percentage of instances that are correctly-labelled as a out of the total amount classified (correctly or not) as a. Thus recall represents how many true instances of an action were found and precision indicates how many classifications were wrong for that action. An extreme example is where all actions are classified as a. In this scenario, the recall would be 100% for action a but the precision would be very low (depending on how many instances of a there are versus the total number of actions). Additionally, the recall for all other (non-a) actions would be 0% and their precision would be undefined since there are no instances detected.

To generate classification results without taking into account sequence evidence, the sequences for each activity are split up into their constituent actions (according to the ground truth) and these isolated actions were then used to train a set of new HMMs, one for each action. Tenfold cross-validation is used for testing, where a test-case action is classified by running it against each HMM and choosing the most likely model as the action label for that test case. Classification is at a per-action-instance level, and so this method is referred to as the "separate models" method.

Classification results using sequence evidence are already provided by the experiments using this paper's proposed segmentation method. However, it is inappropriate to use the recall rates from Section 6.3 since they are based on a definition of "true positive" that uses heuristic criteria. These criteria were designed to help analyse border detection rather than for comparison against non-sequence-based approaches such as the separate models method. Therefore, this section uses a more traditional definition of "true positive" by analysing the classification at the per-frame level (the level at which labelling actually occurs). This is referred to as the "sequence-based" method. At each frame, correctness of classification is determined by checking the most likely label for that frame against the ground truth. By taking all the frames for an action across all sequences, it is possible to see what proportion of *frames* for each action was correctly classified. This then can give an indication of relative performance versus the separate models method, since the latter shows the proportion of *instances* for each action that was correctly classified.

Tables 2 and 3 show the classification results for sequence-based and separate models in both activities. Note that the first and last actions of the sequence-based method (*Type* for Printer and *Walk* and *Drink* for Tea) have a misleadingly good accuracy since the first and last frames are always perfectly "segmented" by virtue of being the endpoints of the sequence.

The sequence-based method is significantly handicapped by the fact that it must perform segmentation as well as classification on all actions except the outer boundaries (i.e., first and last frame of each sequence is always perfectly segmented by default). In contrast, the instances in the separate models method are manually presegmented according to the ground truth, removing the possibility of errors due to segmentation. Results show that the sequence-based method's handicap is more than compensated for by the inclusion of sequence evidence, especially for actions whose motions are visually similar. For example, *GetPaper*, *LoadPaper*, and *GetPrintout* from the Printer activity all involve the actor extending his arm. The separate models method confuses all three actions with a slight tendency to classify them as *GetPaper*, which explains why *GetPaper* has a reasonably good recall (76.0%) but poor precision (37.6%). A similar situation exists for the Tea actions *GetKettle*, *PutKettle*, *OpenTap*, *CloseTap*, *FillKettle*, and *GetCup*. These "grabbing"-style actions are all confused with one another under the separate models method.

For actions such as *Walk* or *SitDown*, the separate models method is slightly better than the sequence-based approach, but this is entirely due to the fact that separate models enjoys the advantage of perfect segmentation and the fact that these actions are not visually similar to any other action. In fact, the sequence-based method is only comprehensively outperformed on a few actions, including *TypeRetract*, *TypeReach* (Printer), *GetCup*, and *Sitting* (Tea). These actions are poorly classified by the sequence-based method mostly due to errors in segmentation—since they are so short in duration, any inaccuracies in segmentation have a significant effect on the per-frame precision and recall.

The benefits of temporal sequence stem from the fact that only neighbouring actions in the sequence are likely to be confused with the correct action. This is because an action that is several stages away from the correct action at time t is very unlikely to occur at time t, and so has almost no chance of being the most probable label. In the case of Printer and Tea, which have at most two adjacent actions due to their strict ordering, classification is really only a choice between the true action and its two immediate neighbours. This narrowing of the list of possible classes serves to improve both precision and recall, even when the actions must be segmented as well as classified.

6.6. Validation of segmentation results

One aspect of the system that has yet to be substantiated is whether the system is actually recognising actions based on their visual motion. It is altogether possible that the timing of the sequence is used in conjunction with prominent "landmark" actions to approximate the position of subtler actions that the system is not actually detecting. This would be a problem for the generality of the approach—if the ordering of actions is the major factor in finding subtle actions, the system could not be used for activities whose subactions are only weakly ordered.

In an attempt to verify that the system is not relying on indirect evidence to label subtle actions, a new "activity"

Table 2: Printer classification accuracy. The four different *Walk* actions are analysed as one (the same for *Type*). Good accuracy for action is denoted by % while % refers to poor accuracy. Source: [2].

Model	Sequence-based (per frame)					Separate models (presegmented)				
Action	TP count	FP count	Actual count	Recall	Precision	TP count	FP 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%

Table 3: Tea classification. The four different *Walk* actions are analysed as one (the same for *GetKettle, PutKettle*). Good accuracy for action is denoted by % while % refers to poor accuracy. Source: [2].

Model	Sequence-based (per frame)					Separate models (presegmented)				
Action	TP count	FP count	Actual count	Recall	Precision	TP count	FP 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%

was designed. This activity contains six actions that cannot be detected due to the limitations of the pose estimation, such as *CrossingArms*, *HandsInPockets*, *HandsAtSides*, and so forth. These six undetectable actions are grouped into two sets of three actions, with the two groups separated by *Walk*, a prominent action which should always be detected. If the system is truly performing recognition based on visual motions, the accuracy of segmentation for the undetectable actions should be very poor. The *Walk* action is included as a control action and hence it should be segmented highly accurately.

Since all actions involve either standing in place or walking about, the new activity is referred to as StandAround. Fifteen sequences were taken of this activity type for testing and a leave-one-out cross-validation was performed to generate the results in Table 4. As can be seen from the results, the recall and precision for almost all actions is very poor—in many cases less than 20%. In addition, 14 event instances were completely missed, which is proportionally far higher than the three missed in each of the Printer and Tea sequences, considering that there are only 15 sequences (in comparison to the 50 sequences for Printer and Tea).

Table 4: StandAround classification accuracy, evaluated on a per-frame basis. Good accuracy is represented by %, while % denotes poor accuracy.

Action	TP count	FP count	Actual count	Recall	Precision
Walk	5014	510	5750	87.2%	90.8%
ArmsAtSides	551	218	1855	29.7%	71.7%
HandsToPocket	614	1931	1260	48.7%	24.1%
HandsInPocket	556	587	3810	14.6%	48.6%
ArmsToSides	246	997	940	26.2%	19.8%
CrossingArms	93	681	405	23.0%	12.0%
ArmsCrossed	965	1583	1955	49.4%	37.9%
HandsToBack	126	173	345	36.5%	42.1%
HandsAtBack	1384	336	1960	70.6%	80.5%
HandsToFront	74	220	395	18.7%	25.2%
HandsAtFront	1062	588	1565	67.9%	64.4%

There are three notable exceptions to this generally poor accuracy: *Walk*, *HandsAtBack*, and (to a lesser extent) *HandsAtFront*. Since *Walk* was specifically included because it is a prominent action, its high accuracy is not unusual. Conversely, *HandsAtBack* and *HandsAtFront* were not expected to be reliably detected by the pose estimation. Inspection of the pose skeleton reveals that this assumption is false—the skeleton *is* sufficiently sensitive and stable enough to detect the slight leaning forward of the actor during *HandsAtBack* and slight lean backwards during *HandsAtFront*. Together with the ordering of the sequence, these motions are distinct enough to allow the system to separate the two actions from the rest of the activity. This is true only because of the contrived nature of the StandAround activity—there is not much scope for variation between instances of each action since each action (apart from *Walk*) was intended to produce very little detectable motion. More realistic activities such as the Printer and Tea sequences contain more complex motions and thus exhibit more noise.

These exceptions notwithstanding the results conclusively show that the success of segmenting the Printer and Tea sequences cannot be explained away as an over-reliance on the temporal ordering. Rather, a significant contribution is being made by the detection and recognition of the visual motions that characterise each subaction.

6.7. Variant sequences

So far, this paper has only considered test cases that closely follow the strict left-right sequence imposed by the HMM model. This section explores the ability of the system to correctly segment sequences which differ from the strict model. Specifically, variations were performed where some actions were left out. For the Printer activity, this involved a scenario where the printer actually had paper already loaded, thus the actions relating to retrieval of spare paper and loading the printer are omitted (dubbed Printer-HasPaper). Similarly for the Tea activity, the kettle was assumed to be full and so does not require filling at the sink (referred to as Tea-KettleFull).

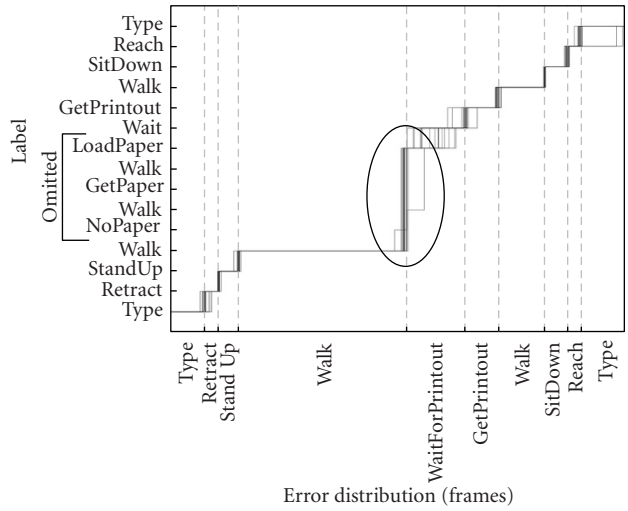
The 50 sequences from the full activities previously analysed (Printer and Tea) were used to create two sets of training data, one for each of Printer-HasPaper and Tea-KettleFull, respectively. Fifteen sequence instances for each variant were recorded and tested with the appropriate training data set. Variant sequences were not included in the training set. The success of segmentation is shown in Figure 8. Note that most of the omitted actions are skipped in almost all cases, with only one Printer-HasPaper sequence and three Tea-KettleFull sequences incorrectly producing labels for these omitted actions.

Of most concern is that the true beginning of *WaitForPrintout* is universally (and mistakenly) labelled as the start of the omitted *LoadPaper* action. This failure also means that the labelling for *WaitForPrintout* is spread out between *LoadPaper* and *GetPrintout* since the system has no evidence for the start of *WaitForPrintout* (because it was allocated to *LoadPaper*). In spite of this, segmentation is mostly corrected at the next action (*GetPrintout*). A similar situation exists for Tea-KettleFull—the start of *PutKettle* is always incorrectly labelled as *CloseTap* and consequently the labels for *PutKettle* are pushed towards the end of the true action. Segmentation of the subsequent action (*WaitBoil*) is not adversely affected. More investigation is required to explain exactly why the system consistently mislabels the start of the *WaitForPrintout* and *PutKettle* actions with the preceding (omitted) action labels (*LoadPaper* and *CloseTap*), especially since the exact error occurs in both activities even though the two scenarios are different. More experiments with different activity variants would need to be performed to determine which situations produce mislabelling and which situations do not suffer from the problem (if any).

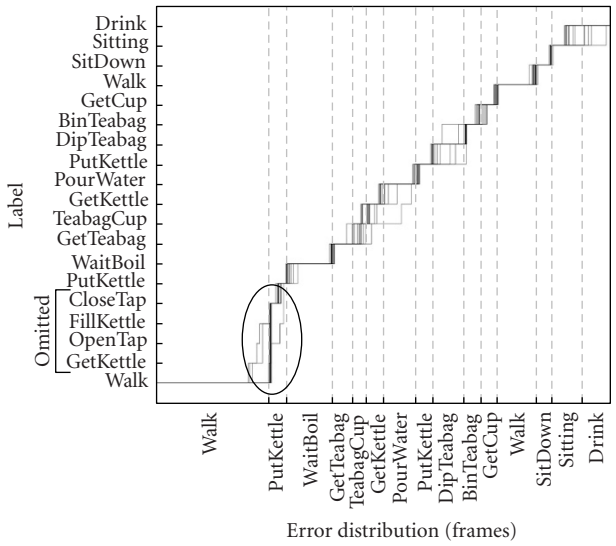
Not shown in the step plots is the amount of noise that is filtered out by the requirement that an action must occur for at least five frames. Almost every sequence produces short burts of labels for the omitted actions. This is to be expected since the HMM models have no special transition to skip the states relating to the omitted actions. Therefore, transitions through these states cannot be avoided, but the system does minimise the time spent in the states.

Apart from the actions noted, segmentation accuracy is similar to the accuracy of the full Printer and Tea activities. For example, although it can be seen that the *Drink* action in Tea-KettleFull is segmented far too early and with a high uncertainty, this is consistent with the results in Section 6.4 and is not caused by the omission of actions.

These results demonstrate that the system is fairly robust to the omission of actions from the sequence, indicating that the association between motions and labels is strong enough to override the model's inflexible left-right sequence. This has positive implications for more complex variants, such as handling variations in the ordering of subactions. However, weak ordering is very difficult to model using a standard "flat" HMM. It is impractical to design a standard HMM that allows out-of-order transitions at the correct state since the HMM has no inherent structure to facilitate this (since the states are—by definition—hidden). A better alternative would be to use a two-level, hierarchically organized



(a)



(b)

Figure 8: Step plots showing labelling versus ground truth for (a) Printer-HasPaper and (b) Tea-KettleFull for all 15 runs. In most cases, the system correctly avoids generating labels for actions that were not performed in the ground truth (circled area shows labels are skipped for those actions marked as "omitted"). The exceptions are the last omitted actions—*LoadPaper* for Printer-HasPaper and *CloseTap* for Tea-KettleFull.

HMM [24, 25], where the lower level contains one sub-HMM for each action and transitions between sub-HMMs are controlled by the higher level. These sub-HMMs could be trained separately and their parameters fixed. It would then be a relatively straightforward task to design the higher-level transitions between sub-HMMs to reflect the required variation in sequencing since the sub-HMMs map directly onto the actions.

7. LIMITATIONS

The major weakness of the proposed segmentation method is the assumption that the activity type is known in advance, with the only challenge being to segment the actions out of the given activity. In practice, the high-level activity must first be classified before subaction segmentation and classification can proceed. This presents the issue that the activity itself could be misclassified, possibly due to the existence of other activities that appear to be similar but are contextually very different. This would have serious detrimental effects on action classification since the resultant action labels will all be generated from the wrong activity model. However, the inclusion of action labels in the observation vector also provides a potential solution: objects in the activity could be instrumented with sensors that indicate when they are used. This data could allow some of the action labels to be set during inferencing, substantially improving the ability to separate between activity types that involve different objects or have different times when the same object is used. For instance, if the computer in the Printer activity could notify the system whenever keyboard or mouse input occurs, the *Type* label could be included in the observation stream (instead of being marked as missing), immediately narrowing down the possible activities to those that have interactions with the computer at the particular time(s) when *Typing* occurred.

Then there is the problem of handling activities whose subactions can be executed in different orders. For example, the Printer activity could easily have the actor retrieve spare paper even before moving towards the printer if they knew that the printer is out of paper. As mentioned at the end of Section 6.7, hierarchically structured HMMs would be more suited to modelling such variants since they can model the hierarchical relationship between an activity and its subactions. An added benefit of hierarchies is that the sub-HMMs could be better tailored to the action being modelled. The current flat structure is left-right across the entire model, and although the activity itself is left-right, some subactions are better modelled with cycles in the HMM (such as *Walk* and *Type*).

8. CONCLUSIONS

Missing data in HMMs has been shown to be an elegant and flexible way to model features that are either unavailable or computationally difficult to obtain. It is a natural fit to the problem of dealing with features that are sometimes undetectable, such as in the case for this paper where the measured pose skeleton is often incomplete due to self-occlusions or occlusions by scene objects in real-world scenes. Using missing data, action recognition under conditions of incomplete pose information is highly robust to occlusions—accuracy is still good even when almost half of the feature data is missing. However, there is a limit as to how much missing data the system can cope with before classification accuracy drops off sharply—in the case of the actions explored, the limit occurs when around 45% of the data is missing.

Missing data can also be employed as a means of inferencing on unknown data at a fine-grained scale. By setting the ground truth for action labels during training and later forcing these ground truth labels to be missing (unobserved) during inferencing, it is possible to accurately segment an activity into its constituent actions. An additional benefit is that the temporal relationships between actions can be explicitly encoded into the model of the HMM. This provides a context for action classification which can markedly improve the recognition of actions that are visually similar but occur at different times in the sequence. Furthermore, the use of missing data facilitates a trainable approach to segmentation, thus avoiding the need for heuristic segmentation methods such as sliding windows or manually labelling and interrogating the Viterbi sequence of hidden states.

Finally, it has been shown that the system is not unduly reliant on the temporal sequence. Visual motion is still the dominant factor in action recognition and can even override the ordering defined by the sequence (within the limits imposed by the HMM model). HMM structures that are more flexible than the left-right models used in this research would facilitate the segmentation of less strictly-ordered activities, but the hierarchical relationship between activities and actions is not easily modelled with standard (flat) HMMs—hierarchical HMMs would be much better suited for the task.

REFERENCES

[1] P. Peursum, H. H. Bui, S. Venkatesh, and G. West, "Classifying human actions using an incomplete real-time pose skeleton," in *Proc. 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI '04)*, vol. 3157 of *Lecture Notes in Artificial Intelligence*, pp. 971–972, Springer-Verlag, Auckland, New Zealand, August 2004.

[2] P. Peursum, H. H. Bui, S. Venkatesh, and G. West, "Human action segmentation via controlled use of missing data in HMMs," in *Proc. IEEE International Conference on Pattern Recognition (ICPR '04)*, vol. 4, pp. 440–445, Springer-Verlag, Cambridge, UK, August 2004.

[3] P. Peursum, S. Venkatesh, G. West, and H. H. Bui, "Object labelling from human action recognition," in *Proc. IEEE Conference on Pervasive Computing and Communications*, pp. 399–406, IEEE CS Press, Dallas-Fort Worth, Tex, USA, March 2003.

[4] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 379–385, IEEE CS Press, Champaign, Ill, USA, June 1992.

[5] D. J. Moore, I. A. Essa, and M. H. Hayes, "Exploiting human actions and object context for recognition tasks," in *Proc. IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 80–86, IEEE CS Press, Corfu, Greece, March 1999.

[6] A. K. R. Chowdhury and R. Chellappa, "A factorization approach for activity recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 4, IEEE CS Press, Madison, Wis, USA, 2003.

[7] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman, "Articulated body posture estimation from multi-camera voxel data," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. I, pp. 455–460, IEEE CS Press, Kauai, Hawaii, USA, December 2001.

[8] G. Mori and J. Malik, "Estimation human body configurations using shape context matching," in *Proc. 7th European Conference on Computer Vision (ECCV '02)*, pp. 666–680, Springer-Verlag, Copenhagen, Denmark, May–June 2002.

[9] M. W. Lee, I. Cohen, and S. K. Jung, "Particle filter with analytical inference for human body tracking," in *Proc. IEEE Workshop on Motion and Video Computing (MOTION '02)*, pp. 159–165, IEEE CS Press, Orlando, Fla, USA, December 2002.

[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[11] H. Fujiyoshi and A. J. Lipton, "Real-time human motion analysis by image skeletonization," in *Proc. 4th IEEE Workshop on Applications of Computer Vision (WACV '98)*, pp. 15–21, IEEE CS Press, Princeton, NJ, USA, October 1998.

[12] N. Sumpter and A. J. Bulpitt, "Learning spatio-temporal patterns for predicting object behaviour," in *Proc. 9th British Machine Vision Conference (BMVC '98)*, vol. 2, pp. 649–658, BMVA Press, Southampton, UK, September 1998.

[13] R. Hamid, Y. Huang, and I. Essa, "ARGMode—activity recognition using graphical models," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 4, IEEE CS Press, Madison, Wis, USA, June 2003.

[14] A. F. Bobick and J. W. Davis, "An appearance-based representation of action," in *Proc. 13th IEEE International Conference on Pattern Recognition (ICPR '96)*, vol. 1, pp. 307–312, IEEE CS Press, Vienna, Austria, August 1996.

[15] X. Sun, C.-W. Chen, and B. Manjunath, "Probabilistic motion parameter models for human activity recognition," in *Proc. IEEE International Conference on Pattern Recognition (ICPR '02)*, vol. 1, pp. 443–446, IEEE CS Press, Quebec, Canada, August 2002.

[16] C.-W. Chu, O. C. Jenkins, and M. J. Matarić, "Markerless kinematic model and motion capture from volume sequences," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 2, pp. 475–482, IEEE CS Press, Madison, Wis, USA, June 2003.

[17] S. Park and J. K. Aggarwal, "Recognition of two-person interactions using a hierarchical Bayesian network," in *Proc. ACM International Workshop on Video Surveillance*, pp. 65–76, ACM Press, Berkeley, Calif, USA, November 2003.

[18] M. Brand and V. Kettnaker, "Discovery and segmentation of activites in video," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 844–851, 2000.

[19] M. Cooke, P. Green, and M. Crawford, "Handling missing data in speech recognition," in *Proc. International Conference on Spoken Language Processing*, pp. 1555–1558, IEEE CS Press, Yokohama, Japan, 1994.

[20] C. S. Pinhanez and A. F. Bobick, "Human action detection using PNF propagation of temporal constraints," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pp. 898–904, IEEE CS Press, Santa Barbara, Calif, USA, June 1998.

[21] A. F. Bobick and Y. A. Ivanov, "Action recognition using probabilistic parsing," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pp. 196–202, IEEE CS Press, Santa Barbara, Calif, USA, June 1998.

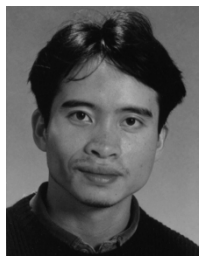
[22] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[23] Y. Al-Ohali, M. Cheriet, and C. Suen, "Introducing termination probabilities to HMM," in *Proc. 16th IEEE International Conference on Pattern Recognition (ICPR '02)*, vol. 3, pp. 319–322, IEEE CS Press, Quebec, Canada, 2002.

[24] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden Markov model: analysis and applications," *Machine Learning*, vol. 32, no. 1, pp. 41–62, 1998.

[25] H. H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden Markov model," *Journal of Artificial Intelligence Research*, vol. 17, pp. 451–499, 2002.

Patrick Peursum is a Ph.D. student in the Department of Computing, Curtin University of Technology. His research interests include computer vision, human-motion analysis, and machine learning. He received his B.S. degree (with honours) in computer science from Curtin University of Technology.



Hung H. Bui is a researcher at Artificial Intelligence Center, SRI International. Previously, he was a lecturer at the Department of Computer Science, Curtin University of Technology. His research interests are in activity recognition, probabilistic reasoning, and machine learning. He received his Ph.D. degree in computer science from Curtin University of Technology.

Svetha Venkatesh is a Professor at the Department of Computing, Curtin University of Technology. Her research interests are in large-scale pattern recognition, image understanding, and applications of computer vision to image and video indexing and retrieval. She codirects the university's Centre of Excellence in Intelligent Operations Management and directs the university's Institute of Multi-sensor Processing and Content Analysis.

Geoff West is a Professor of computer science in the Department of Computing, Curtin University of Technology. His research interests include 3D object recognition, feature extraction, surveillance, smart homes, pervasive computing, small-scale systems, automatic visual inspection, data mining, telemedicine, and related applications. He received a Ph.D. degree in systems engineering from City University, London. He is a senior member of the IEEE, a member of the IEE, UK, a Fellow of the IEAust, and a Charted Engineer.