Action Points: A Representation for Low-latency Online Human Action Recognition

Sebastian Nowozin and Jamie Shotton
Microsoft Research Cambridge
7 JJ Thomson Ave, Cambridge, CB30FB, UK

Microsoft Research TechReport

MSR-TR-2012-68

July 9, 2012

Abstract

Applications of human action recognition in interactive systems such as games require the robust real-time recognition of human actions at low latencies from a stream of observations. The current paradigms of action recognition either treat the pre-segmented sequence as a whole unit to be classified, or classify a range of frames as action, evaluating the performance using a frame-by-frame measure. We argue that both paradigms are limited when addressing latency requirements. Instead, we propose the notion of "action points" to serve as natural temporal anchors of simple human actions. Action points enable latency-aware training and evaluation of online recognition systems. To demonstrate the usefulness of action points we show how two different systems, a Hidden Markov Model and a direct classification approach can be used with action point annotations. We evaluate our approach on two data sets with different input modalities and show that our abstraction of action points is useful in settings where human action recognition has to be performed online and at low latencies.

1 Introduction

Recognizing human actions from video data enables applications such as video understanding, semantic retrieval, surveillance, and human-computer interaction. Depending on the application, a recognition system may be constructed in different ways. If the video data is pre-segmented into coherent units and is processed offline—as is the case for retrieval applications—the models used typically incorporate information across the entire video sequence. For example, the recognition of human actions in Hollywood movies is an offline task [13, 15].

In contrast, *online recognition systems* need to run in real-time and process an incoming video stream without given temporal segmentation. However, such systems may have very different requirements in terms of latency. For example, a sign language recognition system should work in real-time and provide accurate recognition results, but may delay its recognition until an entire sentence or sequence of words is parsed [27]. This yields a beneficial tradeoff between delaying the recognition output for more accurate recognition results on the one side, and providing less accurate output with smaller delays on the other side.

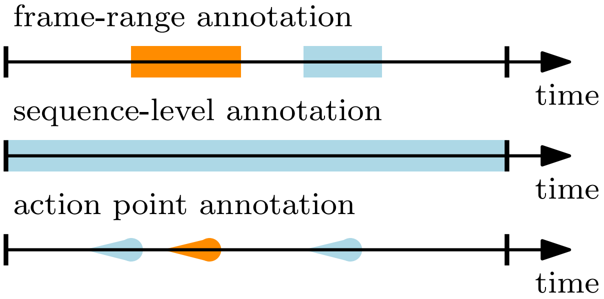


Figure 1: Annotation paradigms for action recognition: *frame-range annotation*, tagging a sequence of frames as being related to the action class; *sequence-level annotation*, marking the entire sequence as an instance of the action; and our proposed *action points*, reference points in each action instance, determined both by the pose and how-we-got-there.

In applications such as interactive gaming this is impossible: within the game, the player demands very low latency recognition. Moreover, not only the recognition of the instance is important, but precise temporal control over the activation of the in-game action relative to the bodily action performance is important as well. Imagine a jump-and-run game in which the jump action is triggered with a small but unpredictable offset; this may render the game unusable. Another application where latency is a concern are touch-based user interfaces. Here low latency is important for interactivity in order to maintain the feeling of control by the user. Therefore it is important that the action is "anchored" to a distinct point in time.

Our work proposes the notion of "action points" for precise temporal anchoring of human actions. Action points can be thought of as marking a specific pose conditioned on "how the user got into that pose". Action points make explicit the latency/accuracy tradeoff and allow accurate evaluation of human action recognition systems in contexts where latency is important. Using action points requires a change in annotation (as shown in Figure 1) and in the performance measure used. Existing human action recognition systems are not well suited to work with action points, and we therefore propose two novel extensions to deal with them. The first is a variant of a Hidden Markov Model (HMM) with an explicit "firing" state marking the action point; the second is a direct classification approach based on randomized decision trees. We evaluate our approaches on two datasets with different input modalities, and show that the latency/accuracy tradeoff can be measured precisely, making action point based systems useful as a component in an interactive system.

The use of a precise temporal anchor for marking action instances bears some similarity in the recent work on *poselets* [2], where sub-instance annotations about body part orientations are used to improve instance-level person detection. Likewise, action points provide sub-instance temporal anchoring to align all instances semantically, reducing the variance that the recognition model needs to cope with.

In summary our contributions are, (i) a simple abstraction for precise temporal anchoring of human actions, (ii) two models and a suitable error measure for low-latency action recognition, and (iii) demonstrating the usefulness of the action point abstraction in the low-latency regime.

1.1 Problem Definition

We now introduce the abstraction of an "action point" as a primitive of natural human actions.

Definition 1 (Action Point) *An* action point *of an action is a single time instance at which the presence of the action is clear and that can be uniquely identified for all instances of the action.*

This definition serves as a useful basis for annotation and recognition. For every suitable action class to be recognized a precise definition can be created and used to annotate a given data set in a consistent manner. For instance, consider the simple action of a "punch", where we could define the *action point* to be the time at which the hand is first maximally extended. At this time instance, the effect of the punch is irreversible, and moreover every straight punch has exactly one point in time that satisfies this definition. Apriori other definitions are possible (for example "point in time with maximum wrist velocity"), but once we agree on a definition we can objectively provide ground truth labels. Therefore action points are not about the semantics of a particular action (this depends on the definition), but about enabling reproducible, temporally-anchored, and low-latency recognition. We provide our definitions of action points used in the appendix.

The abstraction of an action point is a useful primitive for discrete events in interactive systems, such as interactive games, where an action point can be treated in a similar way as a "button press" of a handheld controller. This is in contrast to the mere recognition of an action where the presence of the action is confirmed. Action points confirm the presence of an action as well, but in addition are anchored in time, enabling interactive experiences that require precise temporal control.

For achieving low latency, recognizing action points is most useful if performed *online*, at each time step. In this case recognition is causal, in that it avoids to peek ahead of time, as this would incur additional latency. Formally, we define the task of online action recognition as follows.

Definition 2 (Online Action Recognition) *Given a finite vocabulary of actions $\mathcal{A}$ with action points, and given a sequence of observations $x_t \in \mathcal{X}$ over time $t = 1, 2, \ldots,$ decide at each time t and using only $x_1, x_2, \ldots, x_t$, whether or not an action is active at its action point.*

The online action recognition problem is the equivalent to *filtering* in a temporal state-space model: at each time step we are only allowed to incorporate information from the past. Moreover, we need to decide whether an action point is recognized or not. We now discuss the relationship of our proposed action point abstraction to prior work.

1.2 Related Work

Human action recognition has a long history and the recent surveys of Poppe [22], Weinland et al. [30], Turaga et al. [28], and Aggarwal and Ryoo [1] provide the background to our discussion. We address and discuss only the recognition of relatively simple human actions that could be performed in a few seconds as opposed to longer activities spanning minutes or hours. Most work in the existing approaches are instances of generative temporal state-space models or direct discriminative classifiers, which we discuss separately.

1.2.1 Temporal State-Space Models.

Hidden Markov Models (HMM) [23] and general Dynamic Bayesian Networks (DBN) [18] are flexible generative temporal state-space models with broad applicability to time series tasks. In their seminal work Starner and Pentland [27] successfully applied HMMs to the task of classifying sequences of sign-language gestures by training one HMM per gesture class and evaluating the marginal probability of the observed test sequence under each class model. More sophisticated DBN models as proposed by Brand et al. [3] better represent the internal structure of a gesture as well as correlations such as in left and right arm movements. As with almost all HMM-based recognition approaches, the recognition requires a temporal segmentation and exact inference is therefore restricted to either the offline setting or situations where a reliable temporal segmentation of the gesture within the sequence is available.

As an exception to this, the works [8, 19, 20] explicitly address the online HMM decoding problem under latency restrictions. Eickeler and Rigoll [8] build a continuous and online recognition system. To this end, a single HMM with multiple per-gesture submodels is trained and test-time recognition is performed by online filtering. Heuristics are used to reset the model state after each recognized gesture. Narasimhan et al. [19] show how approximate Viterbi decoding can be performed to trade off latency versus confidence, recovering a range of methods inbetween *filtering* on the one end (no latency, low accuracy) and *smoothing* on the other end (unbounded latency, highest accuracy). This method is applied by Natarajan and Nevatia [20] to online gesture recognition in a hierarchical HMM. Although these methods enable the use of HMMs and linear chain models in the online setting, the gesture or action, once recognized, is not temporally anchored with respect to a known reference.

1.2.2 Direct Classification Approaches.

Direct classification approaches treat the action recognition task as a classification problem on either an entire sequence or small blocks of frames, in a sliding window fashion. For short actions that last no more than a few seconds, Schindler and van Gool [25] have shown that short windows of measurements are sufficient to obtain state-of-the-art recognition performances on multiple data sets. Direct classification approaches can also be interpreted as generalizations of template matching techniques, but with learned features or weights as to which parts of the action are relevant.

Yao et al. [31] consider the recognition of activities by means of Hough-voting random forests [10], where the voting is learned to predict the center of a action from features using both silhouette appearance and skeletal joint features similar to the ones we will use. The final system is evaluated on the TUM kitchen data set using per-frame classification accuracies. As in [10] the system works as an offline recognition system, accumulating votes from both the past and future. Furthermore, in [10, 31] the instance-character of the activities is not accounted for and the evaluation only indirectly measure the achieved temporal accuracy. That said, if modified to only take features from the past, then in principle it seems well suited to our action point framework.

The influential work of Laptev et al. [13] and Marszałek et al. [15] addresses the action recognition problem in cinema movies and treat the task as a sequence-classification problem where the presence or absence of an action has to be determined for a presegmented sequence of up to 1000 frames.

Relationship to Key Poses. Distinct poses, so called *key poses*, often characterize the essence of an action; this has been used for recognition of action instances in [14] and [29]. Such key poses can be manually specified or learned from data, but in any case the goal is the recognition of the action instance.

In contrast action points are precise temporal anchorpoints relative to the action performance, taking both the temporal structure (e.g. clockwise movement versus anticlockwise movement) and the discriminative subparts (e.g. a throw is a throw independent of the foot position and whether it is an overhead or underhead throw). This is a harder problem than mere instance recognition.

For the recognition of action points we take into account temporal information before the action point. Key poses by themselves only quantify the pose, not how we got there, therefore work such as [12] add a postprocessing model to piece the matched poses together. Single frame keypose matching could be seen as a special case of our proposed methods when we limit the temporal window of features to just one frame.

We now introduce our latency-aware performance measure as well as our two recognition models.

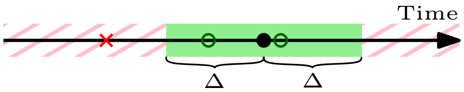


Figure 2: Latency-aware measure of predictive performance for a single action: a fixed time window of size 2Δ is centered around the ground truth action point annotation (marked •) and used to partition the three predicted firing events into correct (marked ○) and incorrect predictions (marked ×). If there is more than one firing event within a ground truth window only one prediction is counted, the remaining ones are ignored. All incorrect detections are counted. The number of correct and incorrect detections determines $\text{prec}_a(\Delta)$ and $\text{rec}_a(\Delta)$.

2 Methods

In the remainder we will denote by $x_t \in \mathcal{X}$ an observation at discrete time t, and by $x_{s:t}$ the sequence $(x_s, x_{s+1}, \dots, x_t)$ of observations. Likewise, we denote by y_t and $y_{s:t}$ an assignment to the predictive variable at time t or range s to t.

2.1 Performance Measure: F-score@Δ

Once we made predictions for the online action detection problem, we assess the quality using ground truth annotations. To this end, we define a performance measure that captures the characteristics of the system in an online setting. These are, its *precision*—how often is the action actually present when the system claims it is, its *recall*—how many true actions are recognized by the system, and its *latency*—how large is the delay between the true action point and the systems prediction.

For a specified amount of tolerated latency, say $\Delta = 100\text{ms}$ (3 frames), we measure the precision $\text{prec}_a(\Delta)$ and recall $\text{rec}_a(\Delta)$ for each action $a \in \mathcal{A}$ as shown in Figure 2. To combine precision and recall into a convenient scalar measure scaled between zero and one we use the balanced F-score [24],

$$\text{F-score}(a, \Delta) = 2\frac{\text{prec}_a(\Delta) \cdot \text{rec}_a(\Delta)}{\text{prec}_a(\Delta) + \text{rec}_a(\Delta)}.$$

For a system detecting multiple actions, we use the mean F-score over all actions, defined as

$$\text{F-score}(\mathcal{A}, \Delta) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \text{F-score}(a, \Delta).$$

Naturally, if we increase Δ, then $\text{F-score}(\mathcal{A}, \Delta)$ cannot decrease. In the experiments we will examine the performance measure as we vary Δ, characterizing the latency-accuracy tradeoff of action recognition systems.

We now propose two methods that are well suited for low-latency recognition, the firing Hidden Markov Model, and a random forest classification approach.

2.2 Firing Hidden Markov Model (F-HMM)

We use a continuous-observation HMM with discrete hidden states [18, 5]. The model is constructed as follows. At each time step t we have one random observation variable X_t with state x_t. Additionally we have an unobserved variable H_t taking values out of a finite set $\mathcal{H} = \left(\bigcup_{a \in \mathcal{A}} \mathcal{H}_a\right) \cup E$, where $\mathcal{H}_a$ is a set of states associated to an individual action a, and E is a set of states for an ergodic *background model* as shown in Figure 3a. The states within the set $\mathcal{H}_a$ are organized along a "fat chain", as shown in Figure 3b, and a subset $F_a \subset \mathcal{H}_a$ is designated as set of *firing states.* If H_t takes a state from F_a, the action point for action a is detected at time t. The fat chain is characterized by its length, its fatness, and the relative position of the set of firing states. Moreover, we allow forward arcs to point one, two, or three steps ahead. For the example shown in Figure 3b, we have a "(4,2,3,1) model": length 4, fatness 2, firing state at position 3, forward-arcs stepping 1.

The intuition motivating this construction is that an action is composed of a sequence of poses where the relative duration of each pose may vary. This variance is captured by allowing flexible forward transitions within the chain. Moreover, there may be two or more distinct ways to perform an action, for example an overarm or underarm throw, and by allowing a fat chain, we can learn the differences and overlaps as components of a mixture model of these implicit subclasses. The fatter the chain, the more potential components are in the mixture model. With this definitions, the full probability model is now specified as HMM,

$$p(H_{1:T}, X_{1:T}) = p(H_1)p(X_1|H_1) \prod_{t=2}^{T} p(X_t|H_t)p(H_t|H_{t-1}), \tag{1}$$

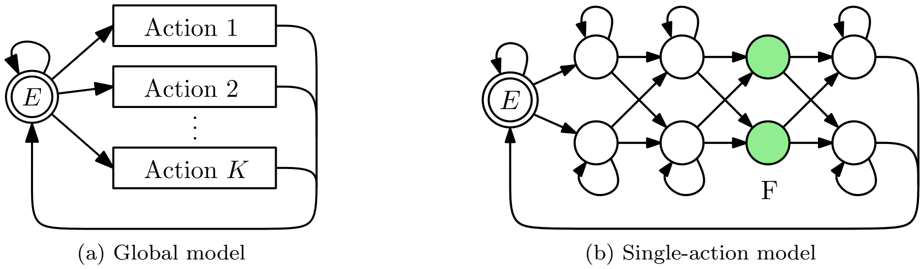
where $p(H_1)$ is the *prior* on the first hidden state, $p(X_t|H_t)$ is the *observation model*, and $p(H_t|H_{t-1})$ is the *transition dynamics model.*

For the observation model $p(X_t|H_t)$ we use one multivariate Gaussian distributions per hidden state with diagonal covariance matrix. The observation domain $\mathcal{X}$ depends on the modality and will be described in the experimental section. The prior model $p(H_1)$ is a multinomial distribution over the set E of background states. The transition dynamics model $p(H_t|H_{t-1})$ is a multinomial distribution over allowed transitions from the state of H_{t-1}. Every arrow in Figure 3b is an allowed transition and thus the model structure is fixed, but the probabilities along all feasible transition arcs are learned.

We learn the model parameters using expectation maximization (EM) [7, 5]. But whereas normal EM maximizes the marginal likelihood of observations,

$$p(x_{1:T}) = \sum_{h_{1:T} \in \mathcal{H}^T} p(x_{1:T}, h_{1:T}),$$

we instead also clamp the hidden variables at the sequence of annotated time-points A for which we observe the action point to the set of corresponding firing



(a) Global model

(b) Single-action model

Figure 3: State diagram of the F-HMM model for low-latency action recognition. **(a)** global model, showing a set of background states (E) and per-action recognition models, **(b)** per-action model: a "fat chain" of forward-linked states until a set of firing states is reached (marked (F), shaded in green) that marks the action point. One such graph and set of states is used per action, and the set of ergodic states (E) is shared among all individual action models.

states. In other words, we maximize the marginal likelihood

$$p(x_{1:T}, H_A \in F_A) = \sum_{h_{1:T\setminus A} \in \mathcal{H}^{(T-|A|)}} \sum_{h_A \in F_A} p(x_{1:T}, h_A, h_{1:T\setminus A}),$$

forcing the model to pass through a firing state of the correct action at precisely the time annotated in the ground truth. We can compute all quantities exactly because the clamping to a *subset of states* is technically no different to the clamping to a single state. This is similar to the sub-action state representation used by Morency et al. [16] for conditional random fields.

During EM learning we place prior distributions on all model parameters; for the multinomial distributions we use Dirichlet priors, for the mean parameters of the Gaussian distributions we use the improper flat prior, and for the diagonal covariance of the Gaussian distribution we use an inverse gamma distribution. Because the Dirichlet and inverse gamma priors are *conjugate* [5] to the multinomial and Normal covariance, we can still perform exact EM.

At test-time we can use normal offline *smoothing*, infering the hidden marginal distributions $p(H_t|X_{1:T} = x_{1:T})$, or use online *filtering* infering $p(H_t|X_{1:t} = x_{1:t})$ [5, 18]. For real-time applications only the latter is useful, but we compare both variants. From the inference results, we define the *probability of an action a* as $p(y_t = a|x_{1:t}) = \sum_{h \in F_a} p(H_t = h|X_{1:t} = x_{1:t})$. Once this exceeds a threshold δ_a we fire the action point at time t.

The F-HMM model aims to represent the temporal structure of multiple actions, at the cost of more complex learning and inference algorithms. We now propose a simpler direct classification approach to recognizing action points.

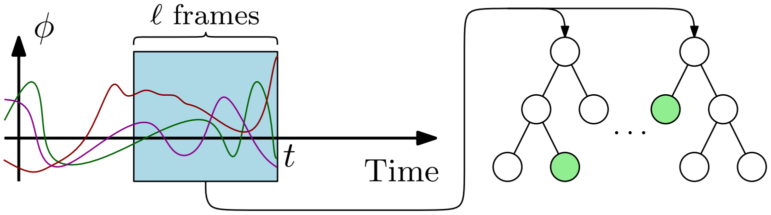


Figure 4: Direct classification approach using random forests at test time: we use a fixed size window of ℓ frames (typically 10 to 40) to extract a feature vector ϕ_t that is classified using multiple decision trees.

2.3 Random Forest Direct Classification

We now show how the direct classification approach can be straightforwardly adapted to the problem of recognizing action points. We will use random forest classifiers [4, 6], but any other efficient classifier could be used.

At test-time, for a time t, we derive a feature vector $\phi_t = \phi(x_t, x_{t-1}, \dots, x_{t-\ell+1}) \in \mathbb{R}^d$ from the last ℓ observations x_t to $x_{t-\ell+1}$, as shown in Figure 4. Depending on the input modality we can use different features such as silhouette masks, skeletal joint angles, and optical flow histograms. The feature vector ϕ_t is evaluated by a set of M decision trees, where simple tests $f_\omega : \mathbb{R}^d \to \{\text{left}, \text{right}\}$ are performed recursively at each node until a leaf node is reached. The parameters $\omega \in \Omega$ of each test are determined separately during the training phase, to be described below. Each tree $m = 1, \dots, M$ produces one class decision $y_t^{(m)} \in \mathcal{A}$ and the approximate *posterior class distribution*

$$p(y_t = a | x_t, \dots, x_{t-\ell+1}) := \frac{1}{M} \sum_{m=1}^{M} I(y_t^{(m)} = a) \tag{2}$$

over action point classes $\mathcal{A}$ and a background class "None" determines whether an action point is recognized. We evaluate $I(\text{pred})$ to 1 if the predicate evaluates to true, and to 0 otherwise. If $p(y_t = a | x_{t:(t-\ell+1)}) \geq \delta$ for an action point class $a \in \mathcal{A}$, we fire the action point as being detected at the current time t.

At training-time, we are given the full observations and action point annotations for a set of N sequences $\{(x_t^{(n)}, y_t^{(n)})_{t=1,\dots,T_n}\}_{n=1,\dots,N}$. Our goal is to learn a set of M decision trees that classify the action points in these sequences correctly by means of (2). We use simple decision stump tests with $\omega = (i, h)$, $1 \leq i \leq d$, $h \in \mathbb{R}$,

$$f_{(i,h)}(\phi_t) = \begin{cases} \text{left} & \text{if } [\phi_t]_i \leq h, \\ \text{right} & \text{otherwise.} \end{cases}$$

To find a good feature to use for splitting, we use the standard *information gain* criterion and training procedure [4]. Hence, we greedily select a split function $f_{(i,h)}$ for each node in each decision tree from a set of randomly generated proposal split functions. The tree is grown until a maximum allowed tree depth is reached or the node is pure, that is, all training samples assigned to that node have the same label. The majority label at each leaf is then used for prediction for all samples that reach this leaf. This choice of training criterion, splits, and prediction is the most commonly used for classification trees [6].

The window size used depends on the type of gestures, and we have experimented with sizes from five to 35 frames. To augment the training data we create additional virtual training examples by time warping the entire sequence with factors 0.9 and 1.1. All frames are taken as negative examples except for the frames marked with action points and a small window of ±3 frames.

3 Experiments

We evaluate our proposed approaches on two different datasets with different input modalities. For the first dataset, the Weizmann Repetitive Actions dataset, we provide action point annotations marking distinct points for each action.

The second dataset has been collected by ourselves and is a large dataset of skeletal joint data recorded with the Kinect SDK, that internally uses the real-time human pose recognition system developed by [26].

3.1 Weizmann Repetitive Actions

The *Weizmann human action dataset* [11] consists of 93 sequences of nine actors performing ten actions, where most actions are repeated multiple times per sequence and every sequence contains exactly one type of action. Although the data set is older and solved, we use this it as a basic verification of our approach.

The original task proposed in [11] is to classify in a leave-one-sequence-out setting the type of action performed in the sequence. In order to verify our instance recognition and the recognition latency we instead recognize each instance in every sequence. To enable the training and evaluation of both our F-HMM and random forest methods, we annotated all sequences with precise temporal action points. The action points are defined semantically as described in the appendix.

As features we use the aligned binary background subtraction masks provided by [11], and use one 61-by-41 matrix per frame. For the F-HMM this simple high-dimensional representation did not work, and we concatenate the binary masks of the 5 previous frames to one 5-by-61-by-41 array and reduce the entire data set using principal components analysis (PCA) to a per-frame observation vector of 20 dimensions. While this runs contrary to the conditional independence assumption encoded in the HMM, the preprocessing of multiple observations into a feature vector is common in speech recognition HMMs and known to improve performance, see e.g. [9, section 2.1].

For the F-HMM model we use an architecture of length 10 and fatness 3, with forward arcs to the next three states and a special firestate at position 6, for a "(10,3,6,3) model" as described in Section 2.2, with 20 additional background states. We set $\delta_a = 0.5$ for all F-HMM results.

For the random forest model we use 111 trees and 16k feature tests at each node to find a split function. The features are allowed to look up to seven frames into the past. From preliminary experiments, we determined $\delta_a = 0.1$ for all action classes; the lower threshold arises from imbalanced classes.

3.2 Kinect Actions Dataset

We collected 54 sequences from 17 actors of a length between 1,000 and 2,000 frames each, at 30 frames per second for a total of 65k frames. Each frame contains 20 joints described by 3D world coordinates, as shown in Figure 5. A total of ten actions (Jump, Kick, Punch, Throw, Shoot, Wave, Change weapon, Duck, Night goggles, Pick up) are performed. Most actions appear in each sequence, but 11 sequences are "background" sequences containing no action at all. Using a fixed definition of action point for each action class, we marked the action points manually using an annotation tool.

We use standard skeletal features derived from absolute, real-world joint coordinates, similar to the ones used in [17] and [31]. In particular, we use three different features: joint velocities (three scalars per joint), joint angles (35 angles defined by triples of joints), and joint angle velocities (35 velocities as change of joint angles over time). These features are illustrated together with the Kinect skeleton in Figure 5. We also define a special "zero-coordinate-joint" at the camera origin $(0, 0, 0)$ and 12 joint angles are defined using this special joint. Therefore 23 out of the 35 angles are invariant to rotation in pose, whereas 12 joint angles are depending on the relative pose to the camera. This allows to distinguish movements relative to the camera. Preliminary experiments indicated the usefulness of all these features to action recognition tasks. The 23 regular joint angles are composed of 12 angles at actual skeletal joints and 11 large-body angles, such as ankle-hip-ankle and wrist-hip-ankle angles.

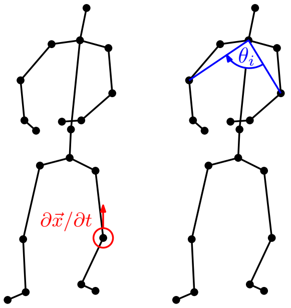


Figure 5: Skeletal features used: joint velocities (left), joint angles, joint angle velocities (right).

For the random forest model we use the same parameters as for the Weizmann dataset but allow feature tests of up to 35 frames in the past. For the F-HMM model we preprocess windows of the past 10 frames of the joint features using PCA to keep a 40-dimensional observation vector at each frame.[1] We use a "(10,3,6,1) model", with each action chain having length 10, fatness 3, firing state at the 6'th step, and one forward arc. There are 20 background states. Training is by means of Expectation Maximization and a maximum of 20 iterations.

[1] We also attempted to incorporate this feature into the random forest classifier but this did not improve the recognition performance.

4 Results

We now present quantitative results on the two datasets, illustrating the different behaviour of the models for recognizing action points.

4.1 Weizmann Action Recognition

The results on the Weizmann action point data are shown in Figure 6. In general the performance profiles look the same for all approaches, increasing monotonically as more latency is tolerated. Here, the direct random forest classification approach works the best, uniformly over all tolerated latencies, and the gains in performance flattens beyond five frames (200ms at 25fps). In contrast, for the F-HMM the performance still increases, which indicates that the F-HMM detects the action but does not localize it as precisely as the random forest approach. The offline smoothing results using the F-HMM outperforms the online filtering F-HMM, as is to be expected. The smoothing approach cannot be used for online recognition because it needs the full observation sequence.

One possible explanation for the poor performance of the F-HMM are the observation features used. To investigate this possibility we also tested a classic HMM model for the task of sequence classification using the same features and eight states, allowing for arbitrary state transitions. We use the same leave-person-out protocol, and for each action class train one HMM separately. The holdout sequences are then assigned to the class of the model with highest marginal probability of generating that sequence. This correctly classifies 84 out of 93 sequences, yielding an overall multiclass accuracy of 90%. This performance falls short of the typical near-perfect results reported for the Weizmann dataset [11] and makes it likely that the features used in the F-HMM are not

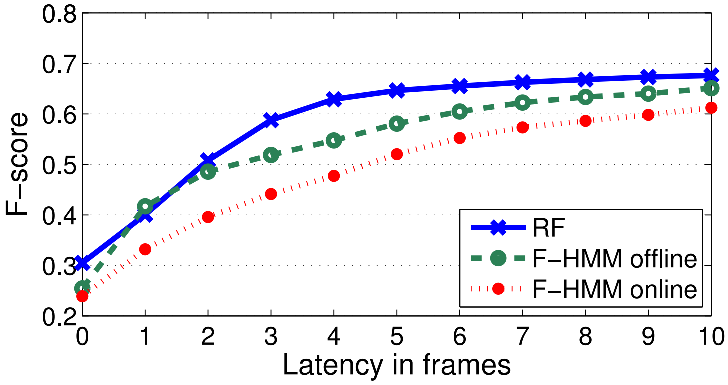


Figure 6: Latency profile of the Weizmann action point recognition task. We show F-scores as a function of the tolerated latency for the F-HMM and direct random forest classification approach. We show the average over ten actions and over nine leave-person-out holdout sets.

expressive enough. Whereas in the discriminative random forest approach additional features can be added easily, for the generative F-HMM approach this is harder because the joint distribution over features is modelled [21]. We also evaluated the sequence-level accuracy for our random forest approach, trained with action point annotations, and deciding for the class by taking the maximum probability over the entire test sequence. This correctly classifies 91 out of 93 sequences correctly, for 98% multiclass accuracy. The gap between the high accuracy achieved on the sequence-level classification task and the comparatively low F-score measures for the action point recognition task highlights that recognizing the presence of a (repeated) action is easier than precisely localizing an action instance in time.

4.2 Kinect Actions Dataset

The quantitative results for the skeletal dataset are shown in Figure 7. For the online recognition task with medium to high latencies (≥ 4 frames) the random forest approach is again superior to the F-HMM. For small latencies (1-3 frames) the F-HMM online approach has a slightly better performance.

The offline F-HMM is included only to understand the improvement that can be obtained if latency is not a concern. The difference between the F-HMM offline and F-HMM online results can be understood as the gain we obtain by taking future information into account when deciding about the presence of an action point.

To further understand the recognition performance across the different models, we visualize the predictions on a representative test sequence in Figure 8. The figures show that the F-HMM predictions are better calibrated in that a

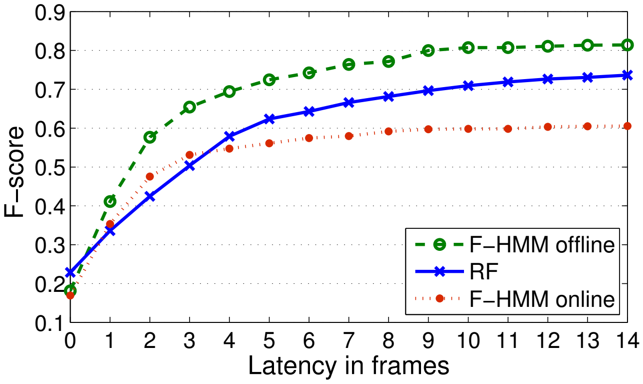


Figure 7: F-Score at tolerated latency for the F-HMM and direct random forest classification approaches. We show the average over ten actions and over the 17 leave-person-out holdout sets. Note that the offline F-HMM results obtained by exact smoothing cannot be used for online recognition; they serve only to show what is lost between offline and online inference. Each frame lasts 33ms.

threshold of $\delta_a = 0.5$ is a reliable detection threshold across all actions. The RF prediction is not as well calibrated, and a smaller threshold $\delta_a \approx 0.1$ has to be used to yield accurate predictions.

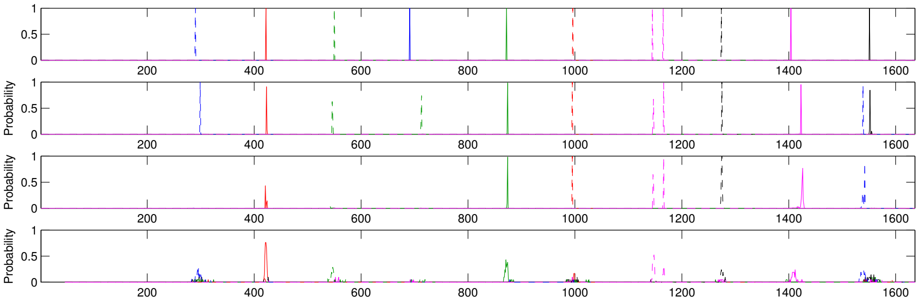


Figure 8: Probability of action for a test sequence of 1636 frames. From top to bottom: ground truth, F-HMM offline, F-HMM online, direct random forest classification. The individual actions (10 in total) are marked with combinations of five different colors and dashed/non-dashed lines. Both F-HMM predictions are confident, the offline F-HMM clearly outperforming the online filtering. Also, the offline F-HMM is able to detect actions more reliably (1st and 3rd ground truth action). The random forest produces less confident predictions but overall accurate recognition results.

5 Discussion and Conclusion

We have made explicit the problem of low-latency online recognition and presented *action points* as a suitable paradigm to provide precise temporal anchoring of actions. Further we have shown how action point annotations can be used in an integrated HMM-based model, and in a simpler direct classification model. Experiments have confirmed the usefulness of this simple abstraction.

However, our proposed method and the use of action points for annotation and detection of actions is limited to actions that are momentary, voluntarily performed and discrete in nature. Continuous activities such as walking or running, or a sequence of well-defined movements as in dancing do not fit our assumptions. In that case, the long-range temporal dependencies are likely better suited to classical sequence models such as hierarchical hidden Markov models or conditional random fields.

The random forest recognition system based on the skeletal joint data has been successfully integrated into a game title that is currently being sold in retail stores. The key advantages have been the runtime efficiency and temporal anchoring information provided by action points.

References

[1] J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 2011. To appear.

[2] L. D. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *ICCV*, 2009.

[3] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *CVPR*, pages 994–999. IEEE Computer Society, 1997.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

[5] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models.* Springer, 2005.

[6] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. Technical report, Microsoft Research, Oct. 2011. MSR-TR-2011-114.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[8] S. Eickeler and G. Rigoll. Continuous Online Gesture Recognition Based on Hidden Markov Models. Technical report, Faculty of Electrical Engineering - Computer Science, Gerhard-Mercator-University Duisburg, Aug. 1998.

[9] M. Gales and S. Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2007.

[10] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *Transactions on Pattern Analysis and Machine Intelligence.* (to appear).

[11] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[12] J. Kilner, J.-Y. Guillemaut, and A. Hilton. 3d action matching with key-pose detection. In *Search in 3D and Video (S3DV)*, 2009.

[13] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR.* IEEE Computer Society, 2008.

[14] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR.* IEEE Computer Society, 2007.

[15] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, pages 2929–2936. IEEE, 2009.

[16] L.-P. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR.* IEEE Computer Society, 2007.

[17] M. Müller, T. Röder, and M. Clausen. Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics*, 24(3):677–685, July 2005.

[18] K. P. Murphy. *Dynamic Bayesian Network: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

[19] M. Narasimhan, P. A. Viola, and M. Shilman. Online decoding of markov models under latency constraints. In *ICML*, 2006.

[20] P. Natarajan and R. Nevatia. Online, real-time tracking and recognition of human actions. In *Motion*, 2008.

[21] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, 2001.

[22] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[23] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–16, Jan. 1986.

[24] C. J. V. Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[25] K. Schindler and L. J. V. Gool. Action snippets: How many frames does human action recognition require? In *CVPR*. IEEE Computer Society, 2008.

[26] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011.

[27] T. E. Starner and A. P. Pentland. Real-time american sign language recognition from video using hidden markov models. In *IEEE Symposium on Computer Vision*, pages 265–270, 1995.

[28] P. K. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn*, 18(11):1473–1488, 2008.

[29] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3D exemplars. In *ICCV*. IEEE Computer Society, 2007.

[30] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. Technical report, INRIA, February 2010.

[31] A. Yao, J. Gall, G. Fanelli, and L. van Gool. Does human action recognition benefit from pose estimation? In *BMVC*, 2011.

Appendix: Kinect Dataset Action Points

We use the following semantic definitions to annotate our recordings.

Punch. Extend one hand with closed fist rapidly towards the camera. The other hand can either remain by the side of the body or like in a boxing-stance be lifted to the chin. *Action point:* when the hand would hit the opponent, i.e. the arm is straight.

Throw. Use one hand to throw an object upwards, towards the camera. Either as an overhead throw, or as a boccia-like throw starting from the side of the body. *Action point:* when the ball would leave the hand.

Kick. Extend one leg rapidly towards the camera, either frontally or from the body sideways. *Action point:* when the leg is straight and would hit the opponent.

Pick up. Grab with one or two hands some object on the ground floor, resuming a normal standing stance afterwards. *Action point:* when one hand touches the ground surface.

Change weapon. Use one arm to reach over the shoulder of that same arm to your backpack (Imagine a shotgun stuck in the top of the backpack), returning to a normal stance afterwards. *Action point:* when the hand is at the turning point, i.e. grabs the weapon and reverses its movement direction.

Shoot. Extend both arms to the front towards the camera, either pretending to hold a single handgun with both hands, or a riffle with both hands, slightly shifted. Remain in that position for a second, then pretend to "fire" the gun by rapidly moving both hands upwards. *Action point:* when the hand-elbow-shoulder angle reaches 135 degrees, i.e. a 45 upward movement over the straight arms have been made to signal the recoil.

Night goggles. Move both hands to the eyes, pretending to look through night goggles for a few seconds, then resume a normal stance. *Action point:* when hands reach the eyes.

Wave. Lift one arm to the side of your head and optionally slightly towards the camera. Wave the hand of that arm left-to-right for at least three times. Then resume a normal stance. *Action point:* each occurrence when the hand has been extended away, reversed direction and is then closest in distance to the subjects head.

Duck. Bend knees to reduce overall body height, while at the same time move both hands rapidly upwards, protecting your head (pretending something is going to fall onto you). Then resume a normal position. *Action point:* at the end of the downward movement.

Jump. Jump with both feet from the ground. *Action point:* at the apogee, when the upward motion reverses into a downward one.