Int J Comput Vis
DOI 10.1007/s11263-012-0550-7

Exploring the Trade-off Between Accuracy and Observational Latency in Action Recognition

Syed Zain Masood · Chris Ellis · Marshall F. Tappen · Joseph J. LaViola Jr. · Rahul Sukthankar

Received: 5 December 2011 / Accepted: 19 July 2012

Abstract An important aspect in designing interactive, action-based interfaces is reliably recognizing actions with minimal latency. High latency causes the system's feedback to lag behind user actions and thus significantly degrades the interactivity of the user experience. This paper presents algorithms for reducing latency when recognizing actions. We use a latency-aware learning formulation to train a logistic regression-based classifier that automatically determines distinctive canonical poses from data and uses these to robustly recognize actions in the presence of ambiguous poses. We introduce a novel (publicly released) dataset for the purpose of our experiments. Comparisons of our method against both a Bag of Words and a Conditional Random Field (CRF) classifier show improved recognition performance for both pre-segmented and online classification tasks. Additionally, we employ GentleBoost to reduce our feature set and further improve our results. We then present experiments that explore the accuracy/latency trade-off over a varying number of actions. Finally, we evaluate our algorithm on two existing datasets.

Keywords Action recognition · Observational latency · Computational latency · Microsoft Kinect · Multiple instance learning · Conditional random field · Bag of words

1 Introduction

With the introduction of the Nintendo Wii, Playstation Move, and Microsoft Kinect controllers, human motion is becoming an increasingly important part of interactive entertainment. Beyond gaming, these technologies also have the potential to revolutionize how humans interact with computers.

A key component to the success of these technologies is the ability to recognize users' actions. A successful interactive system that is intuitive and pleasant to use should embody two fundamental characteristics:

1. High Accuracy—The system must be accurate at recognizing actions.
2. Low Latency—Latency is a key issue for interactive experiences. Systems that lag behind user actions feel cumbersome. This is particularly important for entertainment applications, where problems with lag have resulted in very negative reviews for some motion-based games (Metacritic 2011).

Traditionally, accuracy has driven the design of recognition systems. This paper takes a different path by also focusing on the latency in recognition. We pay particular attention to a type of latency that we refer to as *observational latency*, which is caused when the recognition system must wait for the human to move or pose in a fashion that is clearly recognizable. This is in contrast to *computational latency*, which

S.Z. Masood and C. Ellis contributed equally towards this paper.

S.Z. Masood ( ) · C. Ellis · M.F. Tappen · J.J. LaViola Jr.
Department of Computer Science, University of Central Florida, Orlando, FL 32826, USA
e-mail: smasood@eecs.ucf.edu

C. Ellis
e-mail: chris@eecs.ucf.edu

M.F. Tappen
e-mail: mtappen@eecs.ucf.edu

J.J. LaViola Jr.
e-mail: jjl@eecs.ucf.edu

R. Sukthankar
Google Research, Google Inc, Mountain View, CA 94043, USA
e-mail: rahuls@cs.cmu.edu

Published online: 18 August 2012

is caused by the recognition system itself. The focus of our work is to develop a thorough understanding of the accuracy/latency trade-off that can be used to better design activity recognizers for interactive applications.

The contributions of this paper lie in both novel algorithms and data. To make the measurement of observational latency possible, we introduce a novel action dataset. This set is unique in that it measures how quickly a recognition system can overcome the ambiguity in initial poses when performing an action.

Rather than manually selecting key poses for each action as in Carlsson and Sullivan (2001), we present a novel Logistic Regression learning framework that *automatically* finds the most discriminative canonical body pose representation of each action and then performs classification using these extracted poses. It should be noted that we do not assume pre-defined prototype key poses for each action, but instead choose the key pose through automated learning. For reduced latency, we introduce an additional parameter-controlled cost that forces the system to find a discriminative action pose by observing as few frames of the video sequence as possible. This learning strategy makes it possible to rigorously explore the trade-off between accuracy and latency when spotting actions in an input stream. Section 6 shows how this classifier can significantly outperform the baseline Bag of Words and Conditional Random Field classifiers.

Additionally, we study the effects of reducing the feature count using the GentleBoost algorithm. We find that we can achieve similar classification accuracy by using a small subset of our initial features. Furthermore, we analyze the impact of reducing the number of actions in the classification task on both the latency and the accuracy of the classifier. We find that as actions are eliminated, the best achievable accuracy improves at each latency range.

We also evaluate the performance of our algorithm against both the MSRC-12 (Fothergill et al. 2012) and MSR Action3D (Li et al. 2010) datasets. We classify actions in MSRC-12 with high accuracy, along with most of the actions in the MSR Action 3D set as well. The failure cases in the actions in MSR Action3D set are analyzed in Sect. 10.

2 Basic Approach and Assumptions

With the release of the Microsoft Kinect sensor, reasonably accurate joint positions can be recovered in real-time. Since we use Kinect sensor data, we assume that the user faces the sensor and stands within its field of view. Our method could be extended to non-depth video, but that would require some method of estimating pose, such as Guan et al. (2009) and Yang et al. (2010).

Table 1 The list of actions used in constructing the dataset

Balance	Kick
Climb Up	Punch
Climb Ladder	Twist Left
Duck	Twist Right
Hop	Step Forward
Vault	Step Back
Leap	Step Left
Run	Step Right

Each video in the dataset consists of one person performing one action, from a set of 16 actions, a single time.[1] Figure 1 lists the set of actions used. These actions are chosen based on experiments in Norton et al. (2010), which used the game *Mirror's Edge* to identify a set of actions which would be natural for an interactive gaming experience. Section 6.2 reports results for simultaneously distinguishing between all 16 actions. Although the set of actions is not as extensive as Raptis et al. (2011), it is still substantially larger than in other previous work such as Li et al. (2010).

Each action is performed starting from a rest state, making it possible to measure how quickly the action is recognized from this rest state. Gathering the actions in fashion is reasonable because the vast majority of these actions, which have been chosen through user studies in (Norton et al. 2010), require returning to a rest pose before repeating the actions. In the set of 16 actions in Table 1, the only exceptions to this are “balance” and “run” actions. In addition, beginning each action from a rest pose makes it possible to produce a more realistic estimate of latency in a system with a variety of gestures. While modifications for special cases, such as repeated combinations of punches, may be able to reduce latency for special situations, our goal is to examine latency as it would occur over a wide variety of gestures.

We gathered a new dataset, rather than using an existing one such as the HumanEVA dataset (Sigal et al. 2010), because, at the time we began, previous datasets had not been gathered in a fashion that makes it possible to measure the latency in recognition from the moment the human begins performing the actions. However, recently new datasets have become available, and we present our results in Sect. 10.

2.1 Latency and Action Recognition

We define the latency of an action as the difference between the time a user begins the action and the time the classifier classifies the action. This total time has several different components. At a high level, the latency can be broken down into two parts:

[1] See Sect. 5 for more details on the data gathering process.

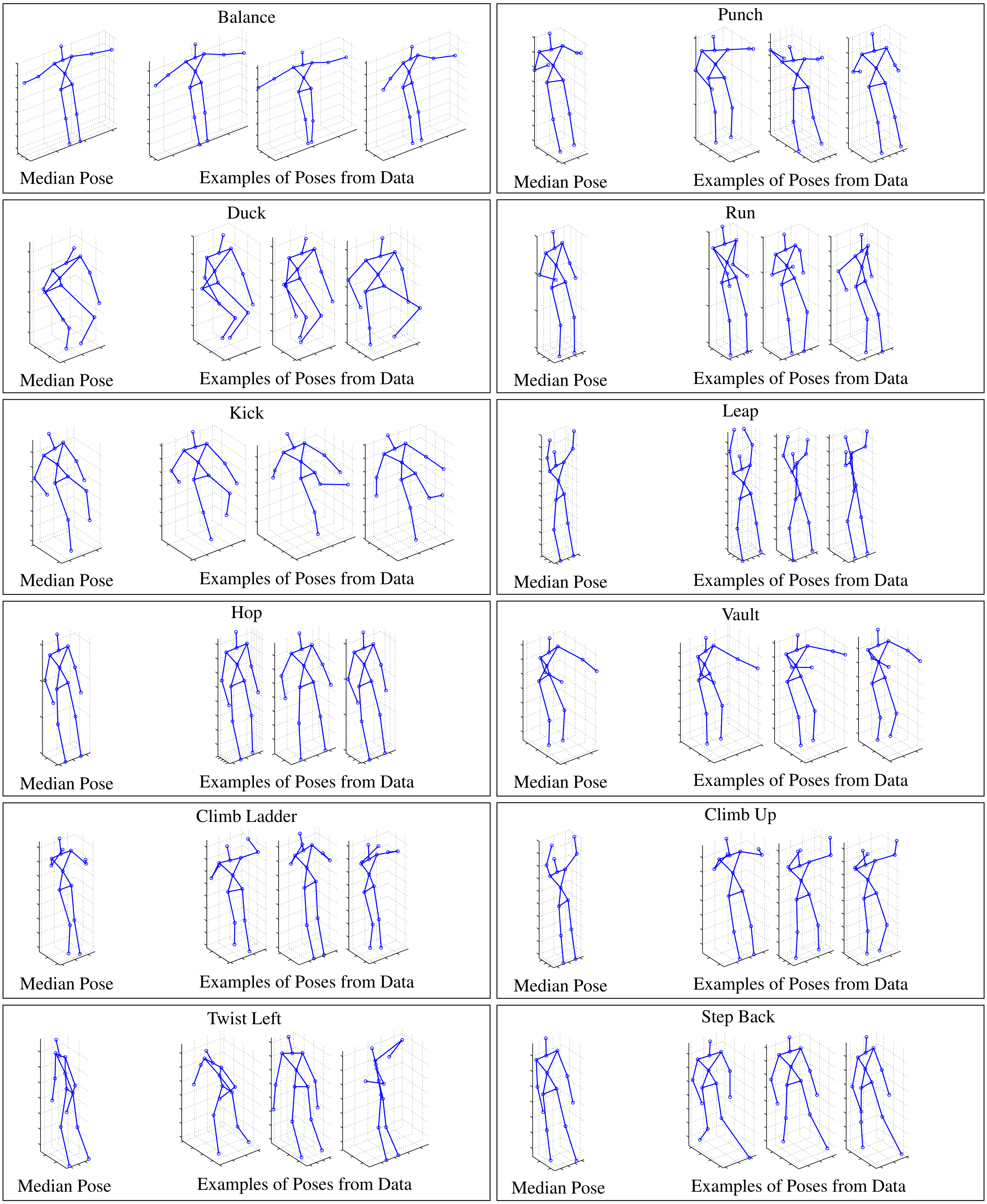


Fig. 1 These skeletons shows several of the poses associated with different actions. The skeleton on the left of each panel is the median of poses associated with each action. The skeletons on the right are examples of poses considered to be most like the canonical pose in a particular video

1. Observational Latency, which is the time it takes for the system to observe enough frames so that there is sufficient information to make a good decision, and
2. Computational Latency, which is the time it takes the system to perform the actual computation on the observations.

It should be noted that a cleverly designed system may be able to perform the necessary computations in between observations, effectively masking the computational latency with the total latency being dependent on only the observational latency.

In this paper, we focus on observational latency because reducing this latency requires examining the fundamental recognition strategy. Once a good strategy is found, it can often be accelerated with optimizations like classifier cascades (Felzenszwalb et al. 2010; Viola and Jones 2001a; Viola and Jones 2001b). In Sect. 8, we show how a GentleBoost method (Friedman et al. 2000) leads to reduced computational latency and better recognition performance.

In the worst case, the observational latency would be the total number of frames it took for a user to perform the action. Such a large latency significantly degrades the user's interactive experience because the system cannot respond to an action until after it has completed. In the best case, the observational latency would be just one frame (at the start of the action), which is infeasible in practice since actions are initially very similar. We present a computational mechanism for designing classifiers that reduce this latency as much as possible, while maximizing recognition accuracy.

2.2 Defining and Measuring Observational Latency

Defining and measuring the observational latency of a system involves subtle decisions. In previous work, such as the Action Snippets proposed by Schindler and Van Gool (2008), and the work of Davis and Tyagi (2006), the system is tested on sequences where the action is being performed continuously (no transition from a rest position), ensuring that every subset of frames shows the action in full progress.

Evaluating data on video where the action is being performed continuously eliminates the ambiguity that occurs as the user transitions into different actions. Observations that contain the user beginning an action can be ambiguous as the user moves through poses that are common to several different actions. For example, at the start of both climbing and punching actions, the user's hand often passes near the head.

This introduces a different type of latency than those measured in Davis and Tyagi (2006) or Schindler and Van Gool (2008). As shown in our experiments, even if it is still possible to recognize the action from a small number of frames, many more frames may be required for the user to assume a distinctive pose that can be reliably recognized.

Our dataset is gathered with each action starting from the initial rest state, where the participant is standing up straight with their arms hanging loosely at their sides, ensuring that the classifier must cope with ambiguous poses at the start of each action. This at-rest pose also enables us to precisely measure the observational latency and to minimize the variation due to the reaction time of the participant. The learning method described in Sect. 4 is designed to find distinctive poses within each action that can be reliably classified. The ambiguity issues are compounded by the large number of actions (16 actions as opposed to the 6 KTH dataset actions used in (Schindler and Van Gool 2008)) because an increased number of actions naturally increases the chance that the different actions appear visually similar.

We argue that measuring latency in this fashion is useful because it is quite likely that an action recognition system will have to recognize multiple actions over the course of the session with the system. In this situation, the lag perceived by the user depends on how quickly the system can detect the beginning of the action. In this dataset, this is measured in terms of the time to move from a rest state to a definitive frame in an action. As mentioned above, this is done to simplify the data collection process.

In future work, we plan to extend this analysis to data observing continuous streams of actions.

3 Related Work

Our work is related to general action recognition systems (Ali and Shah 2010; Li et al. 2010; Shen and Foroosh 2009; Yang et al. 2010). A key, unique aspect of this work lies in our focus on reducing observational latency. Traditionally, action recognition systems have focused on recognizing from temporally segmented videos after the action has been completed. This type of recognition is less applicable for interactive systems as they require real-time recognition. Some systems perform temporal segmentation (Cao et al. 2010; Yuan et al. 2009), but these systems also assume that the action has already been recorded in video.

Efforts have been made in the past to try and extract key pose frames in action video sequences (Cuntoor and Chellappa 2006; Shao and Ji 2009) and use them for the task of action recognition. Carlsson and Sullivan (2001) present a recognition system that matches shape information of individual frames to prototype key frames. Zhao and Elgammal (2008) finds discriminative key frames that are used to weight features in a bag-of-words classifier. However, more recent work action recognition has found better results using simpler bag-of-words representations (Masood et al. 2011), as discussed in Sect. 6.1.1.

Vahdat et al. (2011) use multiple discriminative frames, chosen in a separate learning process. In contrast, our approach chooses the optimal key frames as part of the learning process. Cheema et al. (2011) propose to learn weights for contour-based distinctive key poses and classify using a weighted voting system. Lv and Nevatia (2007) represent actions using a series of 2D human poses and perform silhouette matching between input and key frames. None of the above approaches, however, tackle the problem of observational latency in recognizing actions. Additionally, these methods rely on manual selection of key frames (Carlsson and Sullivan 2001) or the availability of accurate silhouette images (Lv and Nevatia 2007; Cheema et al. 2011).

Hoai and De la Torre's concurrent work on early event detection (Hoai and De la Torre 2012) is philosophically relevant to our research, but approaches the problem from a different angle. They propose two key modifications to structured output SVMs in the typical max-margin framework: (1) augmenting the training set using partial events as positive examples; (2) enforcing monotonicity on detection scores such that smaller partial events are not classified more confidently than their encompassing events.

Techniques exist for reducing latency in sequential data, such as Narasimhan et al. (2006). However, these focus on reducing the latency associated with decoding hidden state sequences from observed data, rather than classifying individual actions as quickly as possible.

A popular strategy for recognizing gestures, used in Alon et al. (2009), and Davis and Tyagi (2006), is based on fitting Hidden Markov Models to different states in the gesture. An advantage of the system proposed in Alon et al. (2009) is that it is also able to spot and temporally segment the actions. However, this segmentation has also not been evaluated in terms of the latency induced.

Pose information has also been incorporated into tracking systems, such as Ramanan et al. (2005), which looks for specific poses while tracking users performing specific actions, such as walking.

The recent availability of commodity RGB-D sensors, such as the Microsoft Kinect, has led to increased research in the application of human pose data (Girshick et al. 2011; Shotton et al. 2011). While this work has resulted in a significant improvement in the ability to estimate body pose, additional recognition steps are still needed to translate these poses into actions. Recent work in Raptis et al. (2011) uses data from the Kinect sensor to recognize dance movements. While this work presents a powerful representation of skeletal data, it was evaluated using around 4 seconds of data per test sequence. This creates a significant amount of observational latency in the system.

A truly interactive system should have the ability to temporally segment actions in the stream of observations, such as the system in Alon et al. (2009) that uses batch-style processing on a complete video to spot gestures. The structure of the dataset used here, with one action per video, leads us to focus on just spotting the beginning of the action. This is discussed in more detail in Sect. 7.

4 Finding Poses with Multiple Instance Learning

To minimize the observational latency at the outset, the classifier must be designed to require as few observations as possible, similar to Schindler and Van Gool (2008). Using the minimum number of frames possible, the system is able to focus on the observational latency inherent in human motion and pose, as discussed in Sect. 2.2.

To minimize the number of observations necessary, this classifier classifies actions based on pose and motion information available from the current, the frame captured 10 frames previously, and the frame captured 30 frames previously, as discussed in Sect. 5. The underlying idea behind the classifier is that the action can be reliably recognized when the user assumes a distinctive pose that characterizes the action. As demonstrated in Sect. 6.2, this strategy perform very well.

The virtue of *automatically* identifying a distinctive "canonical" pose for each action is that this makes it possible to ignore confusing intermediate poses that make classifying similar actions difficult. For example, as shown in Fig. 1, the "climb up" and "leap" actions have a similar median pose as both involve raising the arms. However, the learning process has *automatically* found canonical poses for each action that look very different. When the system observes the canonical pose, it can unambiguously classify the action. This is effective because it enables the system to ignore ambiguous data leading up to the canonical pose without fixating on ambiguous poses that could potentially be misclassified.

4.1 Classifying Videos by Examining Individual Frames

In our dataset, discussed in detail in Sect. 5, each video consists of one individual performing one action only once. These videos are labeled based on the similarity of a frame in the sequence to a canonical pose associated with each action. Thus, the labeling process can be thought of as labeling a bag of frames according to the instances inside that bag.

Formally, the classification begins with a set of weight vectors, $\theta_1, \dots, \theta_{N_A}$, where N_A is the number of actions. The first step in classifying a video is to *automatically* find the frame for each action class that exhibits a canonical pose most similar to that class. Formally, we denote this as a max-response for class c:

$$r_c(\mathbf{x}) = \max_{f \in F} \mathbf{x}_f \cdot \theta_c \tag{1}$$

where F denotes the set of all frames in the bag and $\mathbf{x}_f$ represents the vector of features for frame f.

The probability that the label l of a video should take the correct label T can then be computed using the soft-max function, as in logistic regression:

$$P[l = T|\mathbf{x}] = \frac{\exp(r_T(\mathbf{x}))}{1 + \sum_c \exp(r_c(\mathbf{x}))} = \frac{\exp(\max_{f \in F} \mathbf{x}_f \cdot \theta_T)}{1 + \sum_c \exp(\max_{f \in F} \mathbf{x}_f \cdot \theta_c)}. \tag{2}$$

Adding a 1 to the denominator of Eq. (2) is different than the typical soft-max function. In this formulation, the addition of 1 implicitly models a null action that always has a response of 0. In practice, this makes it possible for the classifier to better manage uncertainty as it classifies an action as null until the user assumes a pose that makes the current action clear.

As mentioned above, this formulation is similar to multiple instance learning because the video, or bag of frames, is classified according to how one of the frames in that bag is classified. The use of the max operator is also somewhat similar to Felzenszwalb et al.'s latent SVM formulation for object detection in images (Felzenszwalb et al. 2010).

4.2 Smooth Approximation

While logistic regression models are typically trained using gradient-based optimization, the introduction of the max operator in Eq. (2) makes the training criterion non-smooth. This can be overcome by using the approximation to the maximum of a set of values $V = v_1, \dots, v_N$ as

$$\max(v_1, v_2, \dots, v_N) \approx \log\bigl(e^{v_1} + e^{v_2} + \cdots + e^{v_N}\bigr). \tag{3}$$

Incorporating this approximation into Eq. 1 leads to the following expression for computing the probability of a particular class:

$$P[l = T|\mathbf{x}] = \frac{\exp(\log(\sum_{f \in F} \exp(\mathbf{x}_f \cdot \theta_T)))}{1 + \sum_c \exp(\log(\sum_{f \in F} \exp(\mathbf{x}_f \cdot \theta_c)))} = \frac{\sum_{f \in F} \exp(\mathbf{x}_f \cdot \theta_T)}{1 + \sum_c \sum_{f \in F} \exp(\mathbf{x}_f \cdot \theta_c)}. \tag{4}$$

As an aside, we note that in Eq. (4), the sharpness of the max approximation could be tuned using a scaling parameter as: $\max(v_1, \dots, v_N) \approx \log(e^{kv_1} + \cdots + e^{kv_N})$. However, such a scaling is subsumed in the weights, θ, during optimization and apart from changing the local minimum that is found, has no impact on the system. We experimentally verified this finding using a range of k from 1 to 100. Thus, we employ a unit scaling, $k = 1$.

Given training examples $\mathbf{x}_1, \dots, \mathbf{x}_{N_T}$ and training labels $t_1, \dots, t_{N_T}$, the weights $\theta_1, \dots, \theta_{N_A}$ for the N_A actions can be found by optimizing the log-loss criterion created by taking the log of Eq. (4). In our implementation, we use the non-linear conjugate gradient algorithm to optimize the log-loss. To increase the generalization performance of the system, a regularization term, $R(\theta_i)$ is summed over all entries in θ and added to the final optimization criterion. To encourage sparsity, we use a Lorentzian term:

$$R(\theta_i) = \alpha \log\bigl(1 + \beta\theta_i^2\bigr), \tag{5}$$

where α and β are chosen to be $1/4$ and 1 through cross-validation.

Replacing the max with the soft approximation causes the system to consider all of the observed frames when computing the label, though the greatest weight is assigned to the frames with the highest response.

In our experiments, the weights are initialized randomly with the initial weights drawn from a zero-mean, unit-variance Gaussian distribution.

5 Dataset and Features

Our dataset was gathered from 16 individuals (13 males and 3 females, all ranging between ages 20 to 35) using a Microsoft Kinect sensor and the OpenNI platform to estimate skeletons. Each individual performs all 16 actions 5 times for a total of 1280 action samples.[2] In each frame, the 3-dimensional coordinates of 15 joints are available. Orientation and binary confidence values are also available for each joint, but are not used in this work. It may prove useful to make use of confidence values in future algorithms to aid the selection of canonical poses, however, in practice we have found that our system is not particularly sensitive to noisy joint data. By allowing our system to learn the weights of each feature, it can automatically reduce the importance of features with high amounts of noise or low information gain.

When gathering the data for each action, we asked the individuals to stand in a relaxed posture with their arms hanging down loosely at their sides. They were then told the action they were to perform and if requested, given a demonstration of the action. A countdown was given at the end of which recording began and the individual performed the action. The recording was manually stopped upon completion of the action. Gathering the data in this fashion simulates a gaming scenario where the user performs a variety of actions, such as punches and kicks, and returns to a resting pose between actions.

We chose a set of features that can be computed quickly and easily from a set of frames. For each given frame of data, we construct a feature set from information in three frames: the current frame x_t, the frame captured 10 frames previously, x_{t-10}, and the frame captured 30 frames previously,

[2] The dataset has been made publicly available at http://www.cs.ucf.edu/~smasood/datasets/UCFKinect.zip.

x_{t-30}. While including data from x_{t-10} and x_{t-30} makes our features not precisely a "pose", we consider it a more intuitive term, as we do not use fine-grained sequence data, nor do we delay classification by looking further ahead in the data stream.

The first set of features is computed by calculating the Euclidean distance between every pair of points in the x_t. From the skeletons computed by the OpenNI software, the 15 joint positions are used to calculate 105 distances.

To capture motion information, the Euclidean distance for all joint location pairs between frame x_t and frame x_{t-10} are computed, resulting in an additional 225 distance pairs.

To capture the overall dynamics of body movement, similar distances are computed between frame x_t and a generic skeleton that simulates a typical pose of a person at rest. The features are computed by translating the center-of-mass of the generic skeleton to the same location of the center of mass of the user's skeleton at frame x_{t-30}. In the case that previous frames are not available, such as when t is less than 30, center of mass of the first frame is used as a substitute. The feature values are the distance between every possible pairs of points in the user's skeleton at frame t and the generic skeleton translated to the user's center-of-mass at frame $t-30$. This brings the total number of distance pairs up to 555. The generic skeleton is computed by averaging the skeleton in the first frame of the training set. Outside of translation, we did not find it necessary to scale or warp the generic skeleton to match the user's pose.

Each feature vector computed at a particular time instant is independently normalized by dividing the vector by the standard deviation of the vector.

The time required for training the system was significantly reduced by transforming these features into a binary, cluster-based representation. Each individual feature value was clustered into one of 5 groups via k-means and replaced with a 5 bit vector containing a 1 at the cluster index of the value, and a 0 for all other bits. Each of these vectors are concatenated to create a new discretized binary feature vector. We add one additional bias term which always has the value 1. The final feature size is thus $555 \times 5 + 1 = 2776$. This transformation leads to a small increase in recognition performance and a significant reduction in training time.

6 Experiments on Temporally Segmented Actions

First, the classifiers are trained on data where the temporal segmentation of actions is available. The goal of the training process is to find the weight vector θ such that a classifier that computes class probabilities using Eq. (4) classifies each video as accurately as possible. This is done by *automatically* finding the frame with the discriminative canonical pose for each action class. For each classifier, this process involves the following steps:

1. Processing the frames in the video to create a bag of feature vectors. A feature vector is computed for each frame after the initial frame in the video. As described in Sect. 5, the vector for a particular frame is computed from the frame, the preceding frame, and the first frame in the video.
2. Learn the weight parameters $\theta_1, \ldots, \theta_{N_a}$ according to the method described in Sect. 4.1.
3. For each action class, $c \in \{1, \ldots, N_A\}$, find the feature vector $\mathbf{x}_{f_c^*}$ such that $\mathbf{x}_{f_c^*} \cdot \theta_c$ has the highest value. At a high-level, this is equivalent to finding the frame in each video that most resembles the action class c. Notice that f_c^* is unique for each class.
4. Label the video with class c^*, where

$$c^* = \arg\max_c \mathbf{x}_{f_c^*} \cdot \theta_c. \quad (6)$$

For evaluation, we used a set of data gathered from 16 people (as discussed in Sect. 5). All of our experiments are implemented using 4-fold cross-validation.

Figure 1 shows visualizations of the best poses learned by the classifier. For each video in the training set, we *automatically* found the frame corresponding to f_c^* for the action contained in the video. The skeleton on the left of each panel in Fig. 1 shows the skeleton created by taking the median of each joint location across the best frames from each action video. The skeletons on the right of each panel show examples of the best frame skeletons. As can be seen in this figure, these skeletons are visually intuitive.

6.1 Baseline Models

Our experiments employ two different models for baseline comparisons: The first is Bag-of-Words (BoW), chosen for its popularity and simplicity of implementation; the second is a Linear Chain Conditional Random Field (CRF), which is a natural choice for a model that can exploit the temporal sequence of hidden state information.

For both these baseline approaches we use the same feature size and training procedure as our proposed method. For the CRF baseline we employ the same regularization term as in Eq. (5).

6.1.1 Bag of Words Model

Bag-of-Words (BoW) is a straightforward approach that is known to consistently perform well on a wide variety of action datasets, such as Liu and Shah (2008). While Zhao and Elgammal (2008) propose extensions for the BoW model that use key frames, we use the original BoW model because recent work (Masood et al. 2011) has shown that in direct comparisons on KTH, the original BoW outperforms the variant proposed in Zhao and Elgammal (2008).

In our baseline, the BoW employs the same distances described in Sect. 5, discretized to 1000 clusters using k-means. Each video is represented by a histogram describing the frequency of each cluster center. Histograms are normalized to avoid bias based on the length of the video. Classification is performed using an SVM with histogram intersection kernel.

6.1.2 Linear Chain CRF Model

The Conditional Random Field (CRF) model (Lafferty et al. 2001) has demonstrated strong performance in classifying time sequence data across several application domains and is thus a natural choice for a strong second baseline. The CRF-based classification strategy is similar to Eqs. (2) and (4). However, in this case, the probability is computed using a function $C_k(\mathbf{y};\mathbf{x})$ that expresses the cost of a sequence of hidden states, expressed in the vector $\mathbf{y}$, given the observation $\mathbf{x}$.

Following Sect. 4.2, the probability of a particular class is expressed as

$$p[l=T|x] = \frac{\exp\{\min_y(-C_T(y;x))\}}{\sum_k \exp\{\min_y(-C_k(y;x))\}} \tag{7}$$

$$\approx \frac{\exp\{-\log\sum_y \exp(C_T(y;x))\}}{\exp\{-\log\sum_k\sum_y \exp(C_k(y;x))\}}. \tag{8}$$

The function $C_k(\mathbf{y};\mathbf{x})$ is constructed to be a typical chain-structured CRF model, with pairwise Potts model potentials (Boykov et al. 2001) and the terms relating the observations to states being linear functions of the observations.

The primary difference between the CRF model and our approach is that the CRF model attempts to model the entire sequences of body poses, while our approach seeks the most informative poses. While it could be expected that the more detailed CRF model could lead to better recognition performance, our results clearly demonstrate the advantages of focusing on a single, reliably occurring, highly discriminative pose.

6.2 Results for Temporally Segmented Actions

To understand the time required for humans to make easily identifiable movements or poses, both the proposed system and the baseline BoW and CRF systems were trained and evaluated on videos of varying lengths. From the base dataset, new datasets were created by varying a parameter termed *maxFrames*. Each new dataset was created by selecting only the first *maxFrames* frames from the video. For videos shorter than *maxFrames*, the entire video was used.

Varying *maxFrames* makes it possible to measure how much information is available in a specific time span. It should be noted that our classifier operates by finding the best feature vector in the first *maxFrames* frames, but that this vector is itself only based on three frames. On the other hand, the BoW and CRF classifiers use the feature vectors from all *maxFrames* frames.

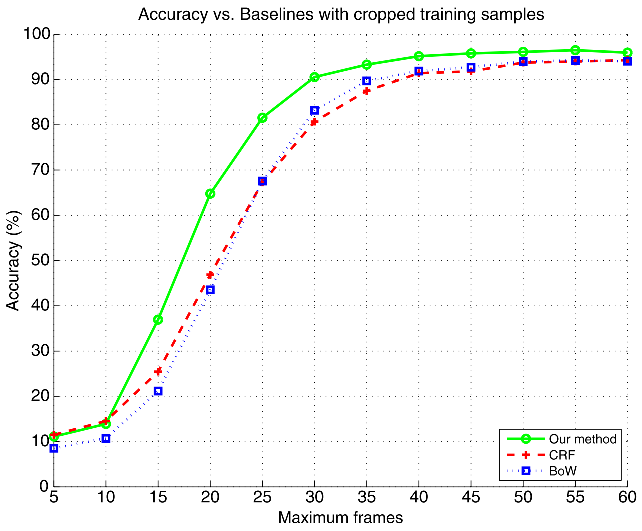
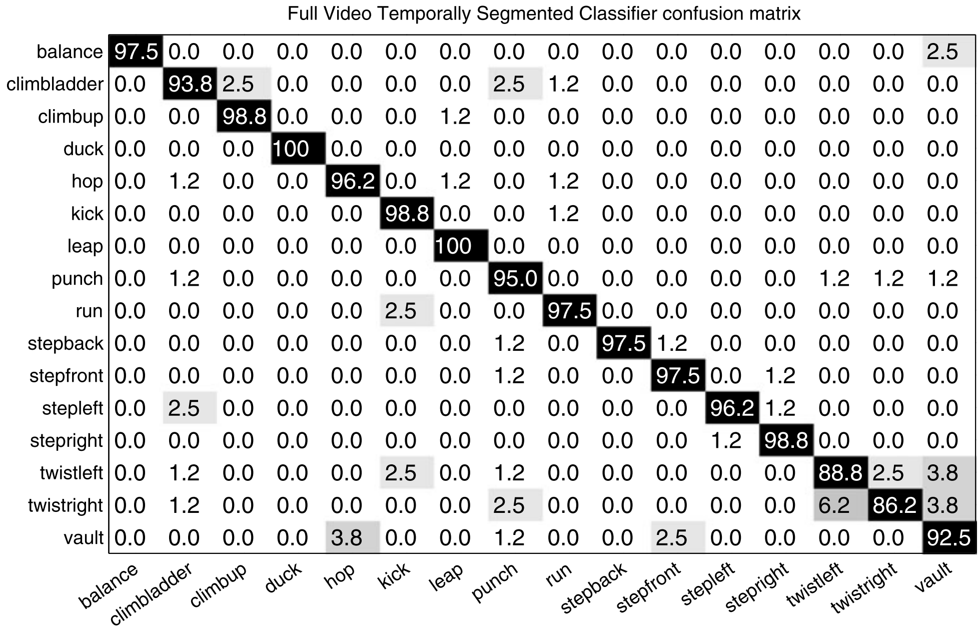


Fig. 2 Accuracy vs. Bag of Words and CRF over videos truncated at varying maximum lengths. The pose-based classifier proposed here achieves higher accuracy with fewer observations

Table 2 A tabular representation of the data from Fig. 2. Note that our proposed method outperforms baselines even when they have access to more frames of pose information

	Frames						
	10	15	20	25	30	40	60
Ours	13.91	**36.95**	**64.77**	**81.56**	**90.55**	**95.16**	**95.94**
CRF	**14.53**	25.46	46.88	67.27	80.70	91.41	94.29
BoW	10.70	21.17	43.52	67.58	83.20	91.88	94.06

As shown in Fig. 2, our classifier clearly outperforms both BoW and CRF classifiers. Each point on a curve in this figure shows the accuracy achieved by the classifier given only the number of frames shown on the horizontal axis. Thus, given only 30 frames of input, our system achieves 90.6 % accuracy, while BoW and CRF classifiers are only able to achieve accuracy rates of 83.2 % and 80.7 %, respectively. Table 2 shows numerical results at several points along the curves in Fig. 2. As these curves show, all of the systems perform poorly given a small number of frames because users have not had enough time to form distinctive poses. Likewise, all of the methods perform similarly well when a large number of frames can be observed. However, in the important middle range, our approach achieves a much higher accuracy given the same number of frames. This shows that our approach improves accuracy for a given latency requirement and can recognize actions at the desired accuracy with a much lower latency.


Full Video Temporally Segmented Classifier confusion matrix

	balance	climbladder	climbup	duck	hop	kick	leap	punch	run	stepback	stepfront	stepleft	stepright	twistleft	twistright	vault
balance	97.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5
climbladder	0.0	93.8	2.5	0.0	0.0	0.0	0.0	2.5	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
climbup	0.0	0.0	98.8	0.0	0.0	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
duck	0.0	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
hop	0.0	1.2	0.0	0.0	96.2	0.0	1.2	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
kick	0.0	0.0	0.0	0.0	0.0	98.8	0.0	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
leap	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
punch	0.0	1.2	0.0	0.0	0.0	0.0	0.0	95.0	0.0	0.0	0.0	0.0	0.0	1.2	1.2	1.2
run	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	97.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
stepback	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	97.5	1.2	0.0	0.0	0.0	0.0	0.0
stepfront	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0	97.5	0.0	1.2	0.0	0.0	0.0
stepleft	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.2	1.2	0.0	0.0	0.0
stepright	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	98.8	0.0	0.0	0.0
twistleft	0.0	1.2	0.0	0.0	0.0	2.5	0.0	1.2	0.0	0.0	0.0	0.0	0.0	88.8	2.5	3.8
twistright	0.0	1.2	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	6.2	86.2	3.8
vault	0.0	0.0	0.0	0.0	3.8	0.0	0.0	1.2	0.0	0.0	2.5	0.0	0.0	0.0	0.0	92.5

Fig. 3 Confusion matrix for full video temporally segmented classification. Results shown are from uncropped action data. Overall accuracy achieved is 95.94 %

Figure 3 shows recognition results of our method with respect to each action in the dataset. We can observe that the *twistleft* and *twistright* actions are confused with each other as well as the *vault* action. Since our feature set is the difference between skeleton joint positions, the limb configurations in *twistleft* and *twistright* are found to be similar to arm and leg positions in each other, as well as *vault*.

This result validates our strategy of looking for "canonical" poses instead of trying to aggregate pose information over time. The BoW and CRF classifiers can be thought of as trying to aggregate weaker pose information over time to get an estimate of the action, but these classifiers do not outperform our method at any frame window size.

Figure 2 also shows that with fewer than 15 frames each classifier performed poorly, but with more than 15 frames the performance of our system rises appreciably. This can be understood by considering the range of movements observed in the beginning frames of the actions. Figure 4 depicts the variation in feature vectors over time. Each point on the graph is created by computing the standard deviation of each feature across all feature vectors at that time. It is clear that the variation in pose and movement at frame 10 is very similar to that at frame 2, indicating that the users have not had the time to assume poses or movement that are significantly different. The peak in variation occurs around frame 30, but our classifier does benefit from having more frames available because these extra frames give more opportunity for the user to assume an easily identifiable pose. By frame 40, our system performs as well as when trained on the full video. The drop-off for larger frames is explained by the low number of videos that have such a large number of frames.

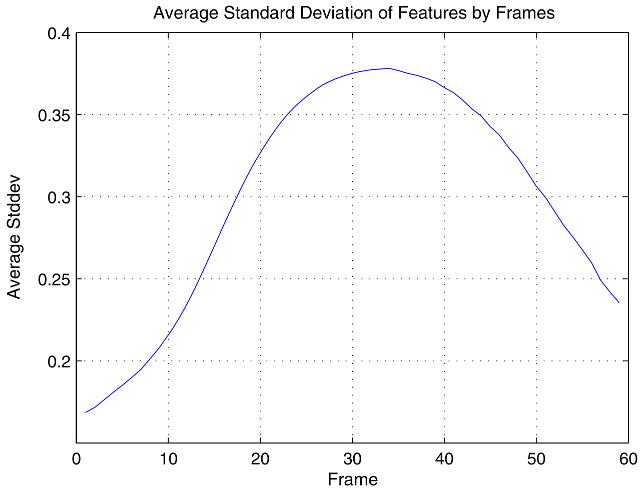


Fig. 4 The standard deviation aggregated over all features per frame. On average, the most informative frame is 30 frames into the action. Our online classifier can accurately recognize actions an average of 10 frames before this peak

Figure 2 also shows that the data gathering procedure, where the user begins from a relaxed pose, does not simplify the recognition task. The improvement in classification accuracy as more frames become available indicates that the system prefers to use later frames; the improved accuracy comes directly from the benefits of observing the distinctive features that are visible only in these later frames.

6.3 Benefits of Soft Approximation

The choice of the frame used for classifying the action is treated as a latent variable during the training process, much like the location of parts in the object detection model in

Felzenszwalb et al. (2010). In this work, we use a soft approximation of the max operator during the training process while the training system in Felzenszwalb et al. (2010) uses a coordinate descent optimization that involves two alternating steps. The first step fixes the location of the parts. This results in a standard margin-based criterion that is optimized using sub-gradient descent.

To measure the influence of the soft approach taken in Sect. 4, we also implemented a coordinate descent approach, similar to Felzenszwalb et al. (2010), with two steps. In the first step, the frames are selected to calculate the score of the different action classifications. For a task with 16 different actions, this means that 16 different frames are chosen for each training video. The frame chosen for a particular action is the frame with the highest score.

Once these frames are fixed, the parameters can be optimized with a negative log-loss criterion similar to Eq. (4). As mentioned above, this criterion is based on one frame per action category. The indices of the frames chosen for each action will be denoted as $f_1, \ldots, f_{N_A}$, where N_A is the number of actions. With these frames, the negative log-loss learning criterion becomes

$$L = -\mathbf{x} \cdot \theta_T + \log\left(1 + \sum_c \exp(\mathbf{x}_{f_c} \cdot \theta_c)\right). \tag{9}$$

This can be optimized with standard minimization techniques. In our implementation, this second step was implemented with a fixed number of iterations of non-linear conjugate gradient descent. We then return to step 1, taking the learned parameters from the second step, and reselect the maximum scoring frames. This process iterates until the sum of squared differences of the learned parameters converge.

To measure the effect of the soft approximation, we performed the same experiment described by Sect. 6.2, with whole videos and four-fold cross-validation. We found the behavior of the coordinate descent approach, with a hard choice of frames, to be sensitive to the initialization of the optimization. As mentioned at the end of Sect. 4.2, a random initialization is used to learn the weights using the soft approximation. If coordinate descent optimization is started with the same type of random initialization, then the average classification accuracy was 73.1 %. This compares poorly with the 95.94 % accuracy achieved using the soft approximation.

However, if we take advantage of the observation that later frames in the video tend to be more indicative, the accuracy can be significantly improved. In a second experiment, we initialized the coordinate descent optimization by fixing the frame used for each action to the 35th frame, or final frame for short videos. This frame was chosen because, on average, this frame was one of the most distinct between different actions. With this initialization, the classification accuracy increases significantly to 92.7 %, which is still slightly worse than the accuracy of our soft approach.

Our conclusion from this experiment is that using a coordinate descent approach, similar to Felzenszwalb et al. (2010), can perform similarly to the soft approach used in this paper, but is much more sensitive to the initialization used.

7 Experiments with Online Detection of Actions

While the temporally segmented results are useful for understanding baseline performance, in real-world scenarios, the system must identify actions in real-time. We focus on a particular sub-problem of the general online action spotting task by focusing on spotting the beginning of each action. This is in line with our goal of characterizing and reducing the observational latency of the recognition system.

The spotting is implemented using the probabilities computed with the soft-max probability, similar to Eq. (4). The weights, θ, are the identical weights learned for the experiments in Sect. 6. The key difference is that they are applied to every frame.

An action is spotted by computing the probability for each class on each frame in the video and comparing each probability to a threshold T, which is optimized on the training set by linear search. Once any class probability exceeds T, that probability is used to classify the action in the whole video. This simulates the task of detecting actions from a stream of real-time sensor input, as the classifier does not know a priori when the action begins or ends.

This process can be thought of as scanning the video until one of the classifiers fires strongly enough, then using that result to classify the whole video. If no probability exceeds T, the video is considered a missed detection and an error.

7.1 Modifying the Learning Criterion to Improve Online Detection Performance

A weakness of the approach described in the previous section is that the classification weights have been trained for the situation where the classifier can view all of the frames to make a decision. This is quite different from the online detection task described above and the weights may not be suitably adapted to this different task.

To better adapt the weights, the learning criterion can be modified to reflect the online detection task more purposefully. This can be done by introducing a new loss L_m that is basically identical to the original training loss, but is computed on videos that have been cropped to m frames, similar to the procedure in Sect. 6.2 with *maxFrames*. This is combined with the original loss to create the learning criterion

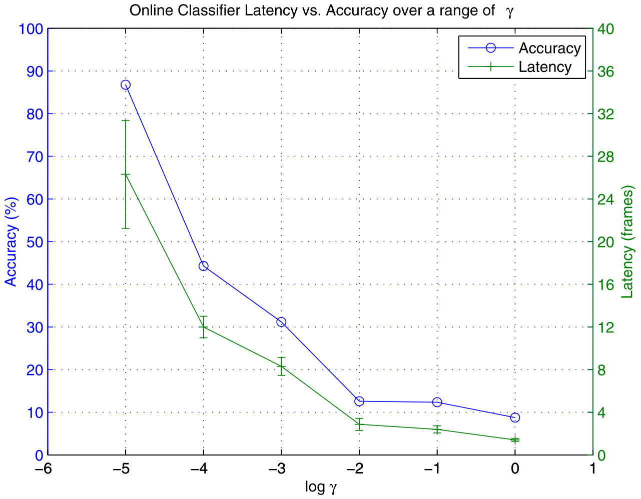


Fig. 5 Latency compared with accuracy, evaluated on the testing set, for different values of γ. The optimal threshold value T was determined by linear search to be 0.9999. Action accuracy is maximal at the 26th frame on average. As γ increases, latency is reduced at the cost of decreased accuracy

for online detection, denoted as $L_{Online}(\cdot)$,

$$L_{Online}(\theta) = L_{Full}(\theta) + \gamma \sum_{m \in M} m L_M(\theta) + R(\theta), \tag{10}$$

where $R(\theta)$ is the regularization term from Eq. (5).

In this criterion, the loss computed over smaller time scales is added to the overall loss thus providing an incentive for detecting the action in as few frames as possible. The set M contains the time scales used in the training process. In our experiments, we use the set $M = \{10, 15, 20\}$. The term $\gamma \cdot m$ is a scaling factor. Incorporating m into the scaling factor places more weight on correctly classifying longer timescales. This is to avoid over-fitting noise in videos with fewer frames.

7.2 Measuring Latency and Accuracy

It is possible to measure the observational latency of the system directly because the system waits until it is confident enough to make a classification. Figure 5 shows the relationship between the observational latency and system accuracy on the testing set for different values of γ in Eq. (10).[3]

Figure 5 shows that as γ rises, the accuracy of the online detection system decreases along with the latency. This indicates that the learning criterion in Eq. (10) provides a parameter to tune the classifier between accuracy and latency. At the optimal γ, the system has an accuracy of 85.78 %. This compares well with the result from Sect. 6.2 as this task is much harder. It should also be pointed out that the online detector still outperforms the baseline classifiers, even though they do not have the burden of detecting the action in the stream. The classifier is able to achieve this accuracy by the 26th frame of the action on average, even though the standard deviation over all features does not peak until after the 30th frame.

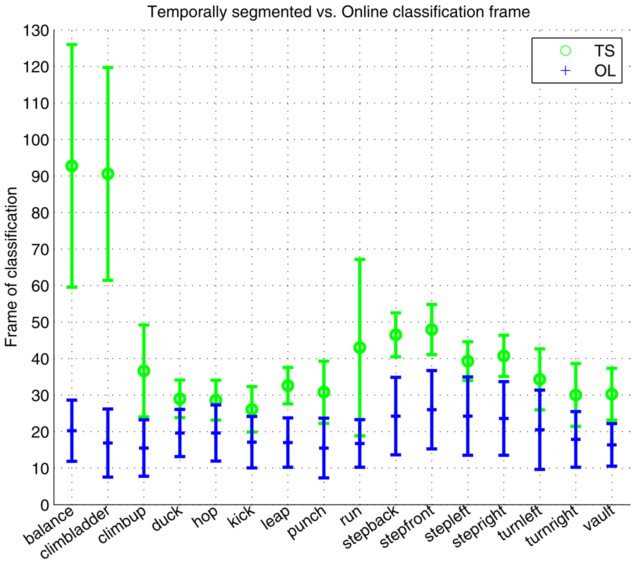


Fig. 6 Comparison of frame of highest response from full video TS classifier with frame of classification from OL classifier. The value of γ for the results shown is 0. The error bars depict the std. deviation of the frame of classification. Recall that the TS classifier must look at the entire pre-segmented action to classify, so its frames correspond to the frames with the highest probability of being the correct action. The OL classifier frame is the earliest point that the probability of the correct action passes the threshold

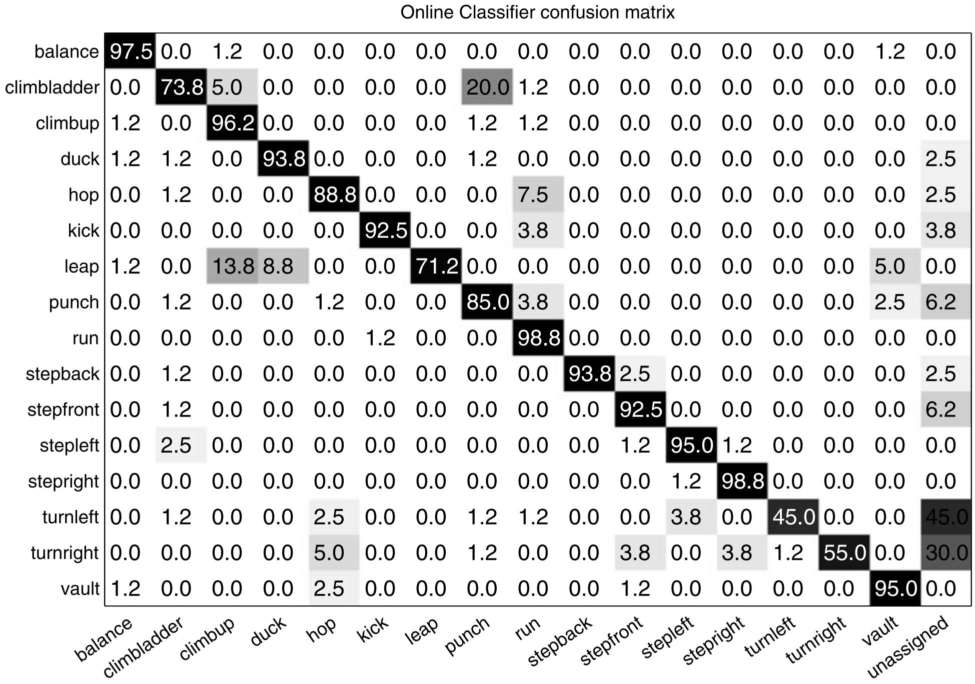
The reason for the drop in classification accuracy can be seen in Fig. 6, which compares the median frame, per action class, chosen by the classifier for temporally segmented videos against that chosen by the online detection system. As can be seen in this figure, the online detection system typically chooses a frame earlier than would be chosen if the entire video could be viewed prior to classification. However, for a 66 % average reduction in classification time, accuracy only drops approximately 8 %.

Figure 7 shows the confusion matrix in the online detection system. A column has been added for those actions where video has been mistakenly labeled as having no action.

7.3 Reducing Latency

Figure 5 also shows that this learning criterion can reduce the latency significantly, but that comes at the cost of significant reductions in accuracy. As γ increases, temporal segmentation classification accuracy decreases gradually. The

[3] The optimal value of the threshold T was found for each value of γ using the training set.



Online Classifier confusion matrix

	balance	climbladder	climbup	duck	hop	kick	leap	punch	run	stepback	stepfront	stepleft	stepright	turnleft	turnright	vault	unassigned
balance	97.5	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0
climbladder	0.0	73.8	5.0	0.0	0.0	0.0	0.0	20.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
climbup	1.2	0.0	96.2	0.0	0.0	0.0	0.0	1.2	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
duck	1.2	1.2	0.0	93.8	0.0	0.0	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5
hop	0.0	1.2	0.0	0.0	88.8	0.0	0.0	0.0	7.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5
kick	0.0	0.0	0.0	0.0	0.0	92.5	0.0	0.0	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.8
leap	1.2	0.0	13.8	8.8	0.0	0.0	71.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0
punch	0.0	1.2	0.0	0.0	1.2	0.0	0.0	85.0	3.8	0.0	0.0	0.0	0.0	0.0	0.0	2.5	6.2
run	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0	98.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
stepback	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	93.8	2.5	0.0	0.0	0.0	0.0	0.0	2.5
stepfront	0.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	92.5	0.0	0.0	0.0	0.0	0.0	6.2
stepleft	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	95.0	1.2	0.0	0.0	0.0	0.0
stepright	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	98.8	0.0	0.0	0.0	0.0
turnleft	0.0	1.2	0.0	0.0	2.5	0.0	0.0	1.2	1.2	0.0	0.0	3.8	0.0	45.0	0.0	0.0	45.0
turnright	0.0	0.0	0.0	0.0	5.0	0.0	0.0	1.2	0.0	0.0	3.8	0.0	3.8	1.2	55.0	0.0	30.0
vault	1.2	0.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0	0.0	0.0	95.0	0.0

Fig. 7 Confusion matrix for online classification with optimal γ. Due to the added constraint of recognizing actions as soon as possible, we see more confusion between the actions. However, by sacrificing a small drop of approximately 8 % in recognition performance (from 95.94 % in Fig. 3 to 85.78 % above), we are able to achieve a significant drop (approx 66 %) in classification latency

online classifier also degrades in performance gradually until the classifier begins firing too early, after which accuracy drops off sharply.

From these results, the accuracy and latency of the system appear strongly correlated. When γ is small, accuracy is high and the system classifies only when it is highly probable to be correct. With large γ, too much emphasis is placed on early classification. Since the amount of variance in the early frames of the data is negligible for accurate classification, we see a drop in both accuracy and latency.

8 Reducing Computational Latency

While our focus is on reducing the observational latency, real-time applications may also face issues with computational latency. To improve this type of latency, we use a boosting approach to find a subset of features that perform as well. This makes it possible to greatly improve the efficiency of our system with negligible reduction in recognition performance.

For selecting the best features, we used a GentleBoost (Friedman et al. 2000) technique to greedily select a set of features. This algorithm operates through a stage-wise minimization of the negative log-likelihood of Eq. (4). At each iteration, the system chooses a feature and parameter value by minimizing a quadratic approximation to the criterion.

When testing our algorithm, we evaluated the system at multiples of 100 features up to a maximum of 300. Table 3 shows our results achieved on temporally segmented videos. We can see that this approach is able to achieve an overall recognition accuracy of 94.06 % by only using less than *one-third* of the original features. This is negligibly lower than our original best result of 95.94 % (as shown in Table 3) but with greatly improved efficiency. The same trend is observed for online classification for the best possible γ value (refer to Table 4).

Table 3 GentleBoost recognition performance for different number of best selected features against our temporally segmented (TS) results. By using only 300 best features out of a total of 2776, we can achieve recognition performance within 2 % of our best temporally segmented result

	GentleBoost Features (TS)			All Features (TS)
Features	100	200	300	2776
Accuracy	92.11 %	93.52 %	94.06 %	95.94 %

Table 4 GentleBoost vs. All Features for online (OL) classification. We see the same trend as observed for the temporally segmented videos in Table 3. For the best possible value of γ, the boosted feature online classification system is within 1.7 % of our original online result

	GentleBoost Features (OL)	All Features (OL)
$\gamma = 1e{-}5$	83.08 %	85.78 %

8.1 Examining the Boosted Features

When examining the features chosen by the boosting algorithm, we can gain insight as to which features are more useful for classification. As described in Sect. 5, the features for each frame are constructed from pairwise distances between joints in the current, the frame captured 10 frames previously and the frame captured 30 frames previously. Of the 300 features selected by the boosting algorithm, 18 contained two joints from the current frame, 88 contained a joint from the current frame paired with a joint from the frame captured 10 frames previously, and 194 contained joints paired between the current frame and the frame captured 30 frames previously. As nearly two-thirds of the features were selected from the latter category, we can infer that pairwise distances between the current frame and frame captured 30 frames previously yield the most useful information toward classification. In other words, our most informative features are those that show how different the user's pose is from their "at rest" reference pose. On the other hand, the least useful features simply measure distances between joints within a single frame.

Now that we know which frames are the most interesting, we should examine which joints are the most informative. Figure 8 shows the occurrence of each joint in the 300 boosted features. The right and left hands are the most common, with the right hand in the lead, most likely owing to the right-handedness of the majority of the training subjects. Less articulate and less used joints, such as the head, torso, and feet, are much less commonly used.

By eliminating less important features, we can reduce the computational latency commonly associated with large sparse feature vectors such as the one used in our classification system. This is especially useful in systems where classification must be done on inexpensive commodity hardware alongside other computational tasks, such as in PCs and game consoles. Further reduction in computational latency can be achieved by tracking fewer joints, especially from the central and lower body regions, as these joints were less commonly selected by the boosting algorithm and are likely to be less informative.

9 Managing Accuracy and Latency by Reducing Possible Actions

While the focus of this paper is on minimizing latency for a fixed set of actions, some applications could allow for flexibility in the specification of which actions must be detected. To study the effect of the choice of actions, we measured how accuracy and latency changed as we iteratively eliminated actions. The set of actions were reduced by greedily removing an action from the set of actions one at a time per value of γ. After training the system across the same set of values of γ used in Fig. 5, the action that was most often confused with other actions was removed. The action sets chosen are shown in Table 5. In this problem, different values of γ affect the actions chosen because higher values of γ encourage the use of actions that can be recognized early.

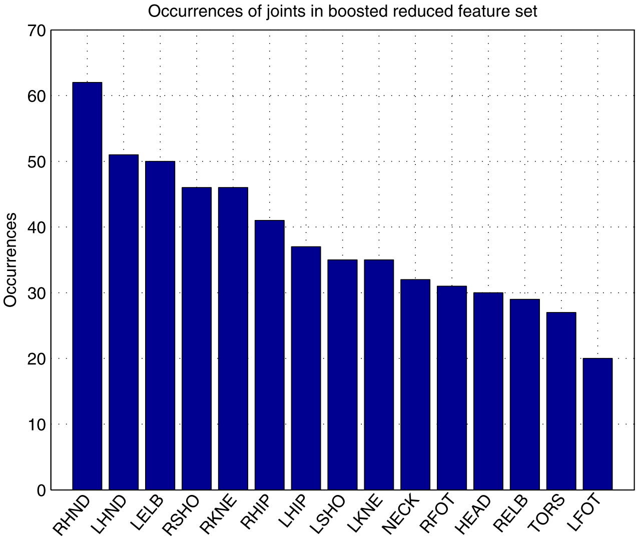


Fig. 8 List of joints by occurrence in the 300 feature set found by boosting. Notice that the right and left hands are the two most commonly used joints. Less articulate joints, such as the head and torso, as well as less used joints, such as the left and right foot, are used less often

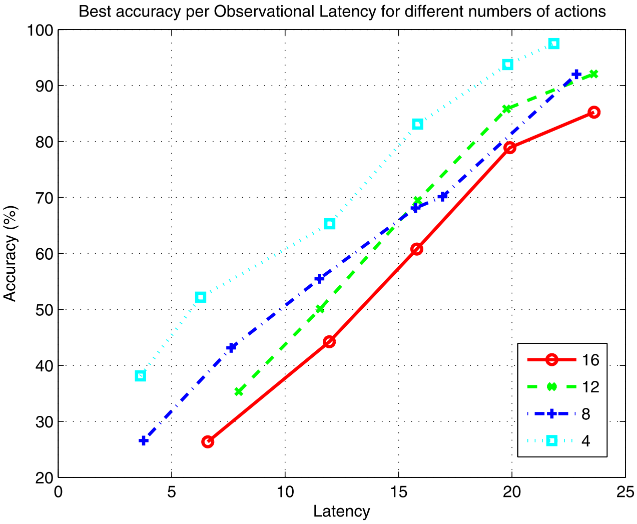


Fig. 9 These curves show how the accuracy and latency in recognition changes as actions are eliminated. As actions that are difficult to recognize are greedily eliminated, the recognition rate at different latencies rises. The accuracies shown are the maximum over the evaluated threshold values

Figure 9 shows curves representing achievable accuracy and latency results for problems with action sets of different sizes. Each of these curves was created by first training our system across the sets created by different values of γ. Using the online classification strategy described in Sect. 7, we then evaluated the system across variety of thresholds, T in Sect. 7. Figure 9 shows the best accuracies achieved for different latency values. All accuracies are averaged in

Table 5 This figure shows the action sets chosen per γ at 16, 12, 8, and 4 actions. Note that for each given γ, at 8 actions, the action set includes the actions from both the 8 and 4 action rows. Likewise, the 12 action set includes the actions from the 12, 8, and 4 action rows, and 16 actions includes the entire column

γ	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0
4 actions	balance	balance	balance	balance	balance	balance	balance	duck	balance
	hop	hop	climbladder	climbup	leap	leap	leap	leap	duck
	stepleft	stepleft	stepleft	stepfront	stepfront	stepfront	run	run	leap
	stepright	stepright	stepright	stepright	stepright	stepright	stepfront	stepfront	run
8 actions	climbup	climbup	climbup	duck	duck	duck	duck	balance	climbup
	kick	kick	leap	kick	run	kick	stepback	climbup	stepback
	twistright	twistright	twistright	stepback	stepback	stepback	stepleft	stepleft	stepfront
	vault	vault	vault	stepleft	stepleft	stepleft	stepright	stepright	stepright
12 actions	duck	duck	hop	hop	climbladder	climbup	climbup	hop	hop
	leap	leap	run	leap	climbup	hop	hop	kick	kick
	stepback	stepback	stepback	twistright	hop	run	kick	punch	punch
	twistleft	twistleft	twistleft	vault	kick	vault	vault	stepback	stepleft
16 actions	climbladder	climbladder	duck	climbladder	punch	climbladder	climbladder	climbladder	climbladder
	punch	punch	kick	punch	twistleft	punch	punch	twistleft	twistleft
	run	run	punch	run	twistright	twistleft	twistleft	twistright	twistright
	stepfront	stepfront	stepfront	twistleft	vault	twistright	twistright	vault	vault

the same four-fold cross-validation framework described in Sect. 6.

As Fig. 9 shows, reducing the number of actions generally increases accuracy. The difference is most significant for lower levels of latency because less information is available to the classifier, making the reduction in the number of possible actions more beneficial. Once only four actions remain, recognition rates rise rapidly, though it should be remembered that these four actions are chosen to make discrimination easy. These actions tend to be easier for the classifier to distinguish from one another, such as *balance* and *step right*.

10 Accuracy Results on Additional Datasets

We also validated our system in terms of recognition accuracy using two additional datasets—the MSR Action3D dataset from Li et al. (2010) and the MSRC-12 Kinect Gesture dataset (Fothergill et al. 2012). Both of these datasets were gathered with Kinect or Kinect-like devices. Similar to our work, the MSRC-12 dataset chooses gestures inspired by interactive games. The MSR Action3D dataset predates the release of the Kinect device and focuses on a mix of sports-based and interaction-based gestures.

10.1 Results on MSRC-12 Kinect Gesture Dataset

The MSRC-12 Kinect Gesture dataset was constructed to both measure the performance of recognition systems and evaluate various methods of teaching human subjects how to perform the different actions (Fothergill et al. 2012). Thus, the dataset is partitioned along different methods of instruction, such as text-only or text and video. Similar to our work, this dataset is designed to make the consideration of latency possible. An action point is identified in the data stream that captures a unique pose, similar to the idea of the canonical pose proposed in this work. However, latency is considered differently. Rather than directly minimizing latency, the experiments in Fothergill et al. (2012) measure whether the system can correctly recognize the gesture within a window 333 milliseconds before and after the gesture's action point.

To replicate the experimental setup described in Sect. 6, we used the action point to divide the videos in this dataset into temporally-segmented examples of each action. The different instruction types were ignored and videos from all instruction types were combined together. Following the protocol in Sect. 6, our system achieved a recognition accuracy of 88.7 %. This is similar to the performance on our dataset and shows that our method can be applied to a wide variety of gestures.

We followed the protocol from Sect. 6 to balance limitations in both our methodology and the protocol in Fothergill et al. (2012). While the experiments in Fothergill et al. (2012) can only measure detections within a fixed window of latency, our experimental method can directly measure latency in recognition. The disadvantage of this methodology is that focusing on time-segmented examples can make the system prone to false-firing in streams of data.

Table 6 This table compares the recognition achieved with our system against previous work on the MSR Action3D dataset from Li et al. (2010). Our approach outperforms a number of previous approaches in terms of accuracy. The methods that outperform our system require that the complete action be viewed before recognition is possible. As we have argued earlier, low-latency, interactive recognition is impossible if the whole gesture must be seen before it can be recognized

Method	Accuracy (%)
Recurrent Neural Network (Martens and Sutskever 2011)	42.5 %
Dynamic Temporal Warping (Müller and Röder 2006)	54 %
Hidden Markov Model (Lv and Nevatia 2006)	63 %
Our Approach	**65.7 %**
Action Graph on Bag of 3D Points (Li et al. 2010)	74.7 %
Actionlet Ensemble (Wang et al. 2012)	88.2 %

10.2 Results of MSR Action3D Dataset

The MSR Action3D dataset from Li et al. (2010) consists of a set of temporally segmented actions, so we followed the experimental methods outlined in Li et al. (2010). Table 6 compares the recognition accuracy produced using our method against previous systems. As this table shows, our method outperforms a number of time-series based methods, including dynamic time warping and a Hidden Markov Model. Our approach is outperformed by two recently proposed methods, but this result should be viewed in the context of the accuracy/latency trade-off. The two approaches that outperform our approach require that the entire action be viewed before recognition can occur.

Insight into our system's performance can be gained by examining recognition accuracy for specific action classes. Table 7 shows the accuracy on the five worst-performing actions and five best-performing classes. Our system is able to easily distinguish between actions that have different body poses, such as a golf swing and wave motions. However, our method has difficulty distinguishing between actions that have a similar body pose and differ primarily in motion, such as a hammering motion and a high throwing motion.

Our system has a difficult time distinguishing between the three types of Draw actions in the dataset: the Draw X, Draw Circle, and Draw Tick actions. These actions in particular do not have a single canonical pose, instead needing a temporally aligned series of poses for classification. Due to the temporal nature of these actions, the entire action must be observed before classification is possible, and thus our low-latency driven approach is not as appropriate. Further study is needed to determine precisely how important low latency is in these types of abstract actions.

Table 7 This table shows the accuracy of the five least-recognized actions in the MSR Action3D dataset (Li et al. 2010) and the five best-recognized actions. Our system performs the worst when the gestures have similar body poses and the motion between gestures is the primary differentiating factor. However, when the actions have different body poses, our system performs quite well

Action	Accuracy	Action	Accuracy
Hammer	0 %	Hand Clap	100 %
Hand Catch	0 %	Two Hand Wave	100 %
High Throw	14.3 %	Forward Kick	100 %
Draw Circle	20 %	Golf Swing	100 %
Draw X	35.7 %	Tennis Serve	100 %

11 Conclusions

Human motion is fast becoming a new paradigm in user interfaces, particularly in entertainment. These systems need to be accurate and have low latency if they are to become widespread. We have proposed a novel system for online action classification that addresses both of these concerns.

Our proposed method converts skeleton data from the Microsoft Kinect to a feature vector comprised of clustered pairwise joint distances between the current, frame captured 10 frames previously, and frame captured 30 frames previously in an action video. In this sense our classifier identifies actions based on canonical body poses. We evaluated a temporally segmented version of the classifier against baseline Bag of Words and Conditional Random Field implementations and found our model to be superior, yielding 95.94 % accuracy.

We then adapted our model to an online variant, and evaluated two schemes to drive down the latency due to classification. We found that we were capable of classifying a large set of actions with a high degree of accuracy and low latency. We additionally introduced a parameter which can be used to fine-tune the trade off between high accuracy and low latency. With this variant we achieved an overall accuracy of 85.78 %.

To address computational latency, we used GentleBoost to select a reduced set of best features. We then examined this set of features and found that the most informative joints were the upper extremities, and the most informative joint pairwise distances were between the current and the generic reference pose. Using these boosted features, we were able to achieve greater efficiency with a negligible drop in recognition performance (94.06 % and 83.08 % for temporally segmented and online classification, respectively).

We further explored the trade-off between accuracy and latency in domains where the number of actions being classified is flexible. We then demonstrated that as the number of actions being classified is reduced, higher accuracy is achievable at lower latency.

Finally, we evaluated our approach on two additional datasets. We achieve high accuracy on the MSRC-12 dataset, and most of the MSR Action3D dataset, and identify a class of actions which are not appropriate for canonical pose techniques.

Acknowledgements Marshall F. Tappen, Syed Z. Masood and Chris Ellis were supported by NSF grants IIS-0905387 and IIS-0916868. Joseph J. LaViola Jr. was supported by NSF CAREER award IIS-0845921 and NSF awards IIS-0856045 and CCF-1012056.

References

Ali, S., & Shah, M. (2010). Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*, 288–303.

Alon, J., Athitsos, V., Yuan, Q., & Sclaroff, S. (2009). A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(9), 1685–1699.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239.

Cao, L., Liu, Z., & Huang, T. (2010). Cross-dataset action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1998–2005).

Carlsson, S., & Sullivan, J. (2001). Action recognition by shape matching to key frames. In *IEEE international workshop at CVPR on models versus exemplars in computer vision*.

Cheema, S., Eweiwi, A., Thurau, C. & Bauckhage, C. (2011). Action recognition by learning discriminative key poses. In *IEEE international workshop at ICCV on performance evaluation on recognition of human actions and pose estimation methods* (pp. 1302–1309).

Cuntoor, N., & Chellappa, R. (2006). Key frame-based activity representation using antieigenvalues. In *Proceedings of the Asian conference on computer vision* (pp. 499–508).

Davis, J. W., & Tyagi, A. (2006). Minimal-latency human action recognition using reliable-inference. *Image and Vision Computing*, *24*(5), 455–472.

Felzenszwalb, P. F., Girshick, R. B., & Mcallester, D. (2010). Cascade object detection with deformable part models. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2241–2248).

Fothergill, S., Mentis, H. M., Kohli, P., & Nowozin, S. (2012). Instructing people for training gestural interactive systems. In *Proceedings of the ACM conference on human factors in computing systems* (pp. 1737–1746).

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *38*(2), 337–407.

Girshick, R., Shotton, J., Kohli, P., Criminisi, A., & Fitzgibbon, A. (2011). Efficient regression of general-activity human poses from depth images. In *Proceedings of the IEEE international conference on computer vision* (pp. 415–422).

Guan, P., Weiss, A., Bălan, A. O., & Black, M. J. (2009). Estimating human shape and pose from a single image. In *Proceedings of the IEEE international conference on computer vision* (pp. 1381–1388).

Hoai, M., & De la Torre, F. (2012). Max-margin early event detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the international conference on machine learning* (pp. 282–289).

Li, W., Zhang, Z., & Liu, Z. (2010). Action recognition based on a bag of 3D points. In *IEEE international workshop at CVPR on human communicative behavior analysis* (pp. 9–14).

Liu, J., & Shah, M. (2008). Learning human actions via information maximization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Lv, F., & Nevatia, R. (2006). Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *Proceedings of the IEEE European conference on computer vision* (pp. 359–372).

Lv, F., & Nevatia, R. (2007). Single view human action recognition using key pose matching and Viterbi path searching. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–8).

Martens, J., & Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the international conference on machine learning* (pp. 1033–1040).

Masood, S., Nagaraja, A., Khan, N., Zhu, J., & Tappen, M. (2011). Correcting cuboid corruption for action recognition in complex environment. In *IEEE international workshop at ICCV on video event categorization, tagging and retrieval for real-world applications* (pp. 1540–1547).

Metacritic (2011). Fighters uncaged critic reviews. http://www.metacritic.com/game/xbox-360/fighters-uncaged/critic-reviews.

Müller, M., & Röder, T. (2006). Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of SIGGRAPH/Eurographics symposium on computer animation* (pp. 137–146).

Narasimhan, M., Viola, P. A., & Shilman, M. (2006). Online decoding of Markov models under latency constraints. In *Proceedings of the international conference on machine learning* (pp. 657–664).

Norton, J., Wingrave, C., & LaViola, J. (2010). Exploring strategies and guidelines for developing full body video game interfaces. In *Proceedings of the international conference on the foundations of digital games* (pp. 155–162).

Ramanan, D., Forsyth, D. A., & Zisserman, A. (2005). Strike a pose: tracking people by finding stylized poses. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 271–278).

Raptis, M., Kirovski, D., & Hoppe, H. (2011). Real-time classification of dance gestures from skeleton animation. In *Proceedings of SIGGRAPH/Eurographics symposium on computer animation* (pp. 147–156).

Schindler, K., & Van Gool, L. J. (2008). Action snippets: How many frames does human action recognition require? In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–8).

Shao, L., & Ji, L. (2009). Motion histogram analysis based key frame extraction for human action/activity representation. In *Proceedings of the conference on computer and robot vision* (pp. 88–92).

Shen, Y., & Foroosh, H. (2009). View-invariant action recognition from point triplets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*, 1898–1905.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1297–1304).

Sigal, L., Balan, A., & Black, M. J. (2010). HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, *87*(1–2).

Vahdat, A., Gao, B., Ranjbar, M., & Mori, G. (2011). A discriminative key pose sequence model for recognizing human interactions. In *IEEE international workshop at ICCV on visual surveillance* (pp. 1729–1736).

Viola, P., & Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 511–518).

Viola, P., & Jones, M. (2001b). Robust real-time object detection. *International Journal of Computer Vision*, *57*(2), 137–154.

Wang, J., Liu, Z., Wu, Y., & Yuan, J. (2012). Mining actionlet ensemble for action recognition with depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1290–1297).

Yang, W., Wang, Y., & Mori, G. (2010). Recognizing human actions from still images with latent poses. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2030–2037).

Yuan, J., Liu, Z., & Wu, Y. (2009). Discriminative subvolume search for efficient action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2442–2449).

Zhao, Z., & Elgammal, A. (2008). Information theoretic key frame selection for action recognition. In *Proceedings of the British machine vision conference* (pp. 1–10).