Measuring and Reducing Observational Latency when Recognizing Actions

Syed Zain Masood Christopher Ellis[*] Adarsh Nagaraja Marshall F. Tappen Joseph J. LaViola Jr.
University of Central Florida
Orlando, Florida

Rahul Sukthankar[†]
Robotics Institute, Carnegie Mellon University
Pittsburgh, Pennsylvania

Abstract

An important aspect in interactive, action-based interfaces is the latency in recognizing the action. High latency will cause the system's feedback to lag behind user actions, reducing the overall quality of the user experience. This paper presents a novel dataset and algorithms for reducing the latency in recognizing the action. Latency in classification is minimized with a classifier based on logistic regression that uses canonical poses to identify the action. The classifier is trained from the dataset using a learning formulation that makes it possible to train the classifier to reduce latency. The classifier is compared against both a Bag of Words and a Conditional Random Field classifier and is found to be superior in both pre-segmented and on-line classification tasks.

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 system that is intuitive and pleasant to use will have 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. A system that lags behind user actions will feel cumbersome. This is particularly important for entertainment applications, where complaints about lag have led to very critical reviews for some motion-based games [11].

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 the latency caused when the recognition system must wait for the human to move or pose in a fashion that is clearly recognizable, in contrast to *computational latency*, which is the latency caused by the recognition system itself. The focus of our work is to develop a thorough understanding of the accuracy/latency trade-off which 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 dataset where every action is performed from a rest state. This set is unique in that it measures how quickly a recognition system can overcome the ambiguity in initial poses when performing an action.

We used this dataset to train and evaluate a novel classification strategy, based on Logistic Regression, that discriminates between different actions by finding canonical body poses that indicate the action being performed. Section 6 shows how this classifier can significantly outperform the baseline Bag of Words and Conditional Random Field classifiers. We also introduce a learning strategy that is able to find the poses that optimize recognition accuracy. This learning strategy makes it possible to rigorously explore the trade-off between accuracy and latency when spotting actions in an input stream.

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 will be using the data from a Kinect sensor, we assume that the user will be standing within the field of view of and facing the sensor. Our method could be extended to

[*]Syed Zain Masood and Christopher Ellis contributed equally towards this paper
[†]Rahul Sukthankar is now also with Google Research.

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

Table 1. The list of actions used in constructing the data set.

traditional imagery, but that would require some method of estimating pose, such as [7, 20].

Each video in the data set consists of one person performing one action, from a set of 16 actions, a single time. Each action is performed starting from a rest state, making it possible to measure how quickly the action is recognized from this rest state. This approach can be criticized as unrealistic since in practice the user will be transitioning between different actions, instead of from action to rest state to action. However, gathering transitions between all actions for a large dataset masks the true latency of each action due to the added cognitive delay. Data starting from a rest state thus simplifies the process of measuring latency, which will be discussed in Section 2.1.

We gathered a new dataset, rather than using an existing one such as the HumanEVA dataset [17], because previous datasets have 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.

Figure 1 lists the set of actions used. These actions are chosen based on experiments in [13], which used the game *Mirror's Edge* to identify a set of actions which would be natural for an interactive gaming experience. In Section 6.2, results will be reported for simultaneously distinguishing between all sixteen actions. This is substantially more actions than in previous, similar work such as [9].

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 returns the action. This total time has several different components. At a high level, the latency can be broken down into the time it takes for the system to observe enough frames so that there is sufficient information to make a good decision, which we will refer to as the *observational latency*, and the time it takes the system to perform the actual computation on the observations, which we call the *computational latency*. 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 just dependent on 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 [6, 18, 19].

In the worst case, the observational latency would be the total number of frames it took for a user to perform the action. Latency this large significantly reduces the user experience because the system can only respond to an action after it is already completed. In the best case, the observational latency would be one frame, at the start of the action, but this poses a challenging detection problem. This work presents a computational mechanism for designing classifiers that reduce this latency as much as possible, while maximizing accuracy in recognition.

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 system of Schindler and van Gool [15], and the work of Davis and Tyagi [5], the system is tested on sequences where the action is being performed continuously, 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, as the user transitions from an initial rest state to the climbing action or punching action, the hands will be in a position near the head. At this time, it will be very difficult to distinguish these actions.

This introduces a different type of latency than those measured in [5] or [15]. As will be 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 pose that can be easily recognized.

The data set used here is gathered with each action starting from a rest state, so the classifier must cope with ambiguous poses at the beginning of the action. The learning method described in Section 4 is designed to find the poses that can be classified unambiguously. The ambiguity issues are compounded by the large number of actions, with 16 actions instead of the 6 actions used in the KTH data set used in [15], because an increased number of actions increases the chance that the different actions will be visually similar.

We argue that measuring latency in this fashion is useful because it is most 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.

3. Related Work

Our work is related to general action recognition systems [1, 9, 16, 20]. A key, unique aspect of this work lies in our focus on the 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. Some systems perform temporal segmentation [4, 21], but these systems also assume that the action has already been recorded in video.

Technique exist for reducing latency in sequence data, such as [12], 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 [2, 5] is based on fitting Hidden Markov Models to different states in the gesture. An advantage of the system proposed in [2] 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 [14], which looks for specific poses while tracking users performing specific actions, such as walking.

A truly interactive system will have the ability to temporally segment actions in the stream of observations, such as [2]. 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 will be discussed in more detail in Section 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 [15]. Using the minimum number of frames possible makes it possible to focus on the observational latency inherent in human motion and pose, as discussed in Section 2.2.

To minimize the number of observations necessary, this classifier will classify gestures based on pose and motion information available from two consecutive frames of data, plus the initial frame in the video, as will be discussed in Section 5. The underlying idea behind the classifier is that the action can be recognized when the user assumes a canonical pose that unambiguously indicates the action. As will be shown in Section 6.2, this strategy can perform quite well.

4.1. Classifying Videos by Examining Individual Frames

In our dataset, discussed in detail in Section 5, each video consists of one individual performing one action a single time. This video is labeled based on the similarity of a frame in the sequence to each canonical pose, which is associated with one of the actions. 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, $\theta_1, \ldots, \theta_{N_A}$, where N_A is the number of actions. The first step in classifying a video is to find the frame for each action class that is most similar to that class. Formally, we denote this as a max-response for class c, where

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

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 L can then be computed using the soft-max function, as in logistic regression:

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

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 similar to the latent SVM described in [6], which is also similar to multiple instance learning.

4.2. Smooth Approximation

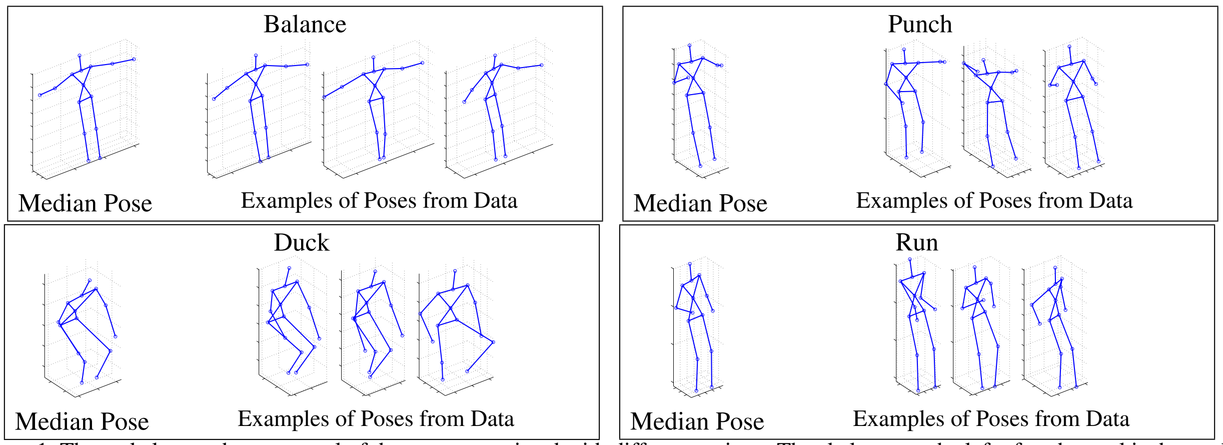
While logistic regression models are typically trained using gradient-based optimization, the introduction of the $\max$ operator in Equation 2 makes the training criterion non-differentiable. This can be overcome using the approximation of the maximum of a set of values $V = v_1, \ldots v_N$ as

$$\max(v_1, v_2, \ldots, v_N) \approx \log\left(e^{v_1} + e^{v_2} + \ldots e^{v_N}\right). \qquad (3)$$

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

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

$$= \frac{\sum_{f \in F} \exp\left(\mathbf{x}_f \cdot \theta_T\right)}{\sum_c \sum_{f \in F} \exp\left(\mathbf{x}_f \cdot \theta_c\right)}. \qquad (5)$$


Balance
Median Pose
Examples of Poses from Data
Punch
Median Pose
Examples of Poses from Data
Duck
Median Pose
Examples of Poses from Data
Run
Median Pose
Examples of Poses from Data
Figure 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.

Given training examples, $\mathbf{x}_1, \ldots, \mathbf{x}_{N_T}$ and training labels $t_1, \ldots, t_{N_T}$, the weights $\theta_1, \ldots, \theta_{N_A}$ for the N_A actions can be found by optimizing the log-loss criterion created by taking the log of Equation 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:

$$R(\theta_i) = \alpha \log(1 + \beta\theta_i^2) \tag{6}$$

where α and β are chosen through cross-validation.

5. Dataset and Features

Our dataset was gathered from 16 individuals using a Microsoft Kinect sensor and the OpenNI platform to estimate skeletons, and is available publicly at `www.cs.ucf.edu/~smasood/datasets/UCFKinect.zip`. 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.

We chose a set of features that can be computed quickly and easily from a set of frames. The basic set of features is computed by calculating the Euclidean distance between every pair of points in the skeleton. Using the skeletons from the OpenNI software, the 15 skeletal positions are used to calculate 105 distances.

To capture motion information, the Euclidean distance is also computed between between all pairs of joint locations between the current frame and the preceding frame, adding 225 more distance pairs. To capture the overall dynamics of the body, similar distances are computed between all pairs of joints between the most recent frame and the first frame in the sequence, bringing the total number of distance pairs up to 555. The first frame is meant to approximate the at-rest pose of the user. In practice, this information can be obtained by a brief initial calibration on the user in an at-rest pose.

The time required for training the system was significantly reduced by transforming these features into a cluster-based quantization. Individual feature values were clustered and replaced with the cluster index of the value. This transformation leads only to a small increase in recognition accuracy, but achieves 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 Equation 2 classifies each video as accurately as possible. 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 Section 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 $\theta_1, \ldots, \theta_f$ according to the method described in section 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

$$c^* = \arg\max_c \mathbf{x}_{f_c^*} \cdot \theta_c. \tag{7}$$

For the learning process in each classifier, the action samples are divided by person into two sets of 10 and 6 people respectively. The accuracy values reported are obtained by training on the first set of actions, and then classifying the second set of the actions.

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

6.1. Baseline Models

We chose two different types of models as baseline models for comparison. We have selected a Bag of Words (BoW) model due to the simplicity of its implementation, and a Linear Chain Conditional Random Field (CRF) model to take advantage of the temporal sequence of hidden state information.

6.1.1 Bag of Words Model

We chose to use a Bag-of-Words (BoW) model for baseline comparison because this approach is straightforward and and has consistently performed well on a wide variety of action datasets, e.g., [10].

The bag of words is computed using the same distances described in Section 5. The expansion to binary features was not used because the raw distance values performed best on this classifier. The BoW representation is created by quantizing the feature representation of each frame to one of 1000 clusters. The clusters were chosen randomly from the dataset. This had similar performance to using the k-means algorithm to find the centers, but was significantly faster.

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. The classifier is implemented using an SVM based on the histogram intersection kernel.

6.1.2 Linear Chain CRF Model

We selected a Conditional Random Field model [8] to compare against due to its strength in classifying time sequence data. The CRF-based classification strategy is similar to Equations 2 and 4. However, in this case, the probability is computed using a cost function $C_k(\mathbf{y};\mathbf{x})$ based on a CRF. This function expresses the cost of a sequence of hidden states, expressed in the vector $\mathbf{y}$, given the observation $\mathbf{x}$.

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

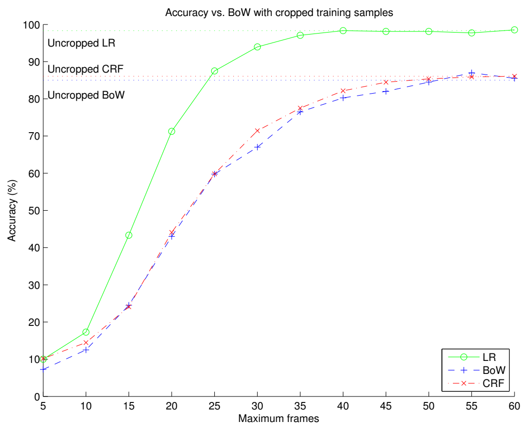


Figure 2. Accuracy vs. Bag of Words and CRF over videos of varying maximum length. The horizontal dashed lines near the top of the graph represent each classifier's results when applied to uncropped samples. The pose-based classifier proposed here significantly outperforms baselines for all lengths, as well as when uncropped videos are used.

$$p[l = T|x] = \frac{\exp\left\{\min_y -C_T(y;x)\right\}}{\sum_k \exp\left\{\min_y -C_k(y;x)\right\}} \tag{8}$$

$$\approx \frac{\exp\left\{-\log \sum_y \exp\left(C_T(y;x)\right)\right\}}{\exp\left\{-\log \sum_k \sum_y \exp\left(C_k(y;x)\right)\right\}}. \tag{9}$$

The function $C_k(\mathbf{y};\mathbf{x})$ is constructed to be a typical chain-structured CRF model, with pairwise Potts model potentials [3] and the terms relating the observations to states being linear functions of the observations.

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 which will be referred to as *maxFrames*. Each new dataset was created by cropping down to only the first *maxFrames* frames from the video. If the video was shorter than *maxFrames*, then the whole 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 uses the feature vectors from all *maxFrames* frames.

As shown in Figure 2, our classifier based on finding the best pose clearly outperforms the baseline classifiers. The

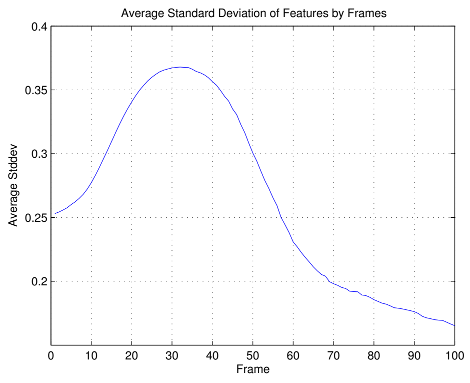


Figure 3. The standard deviation aggregated over all features per frame. On average, the most informative frame is 30 frames into the action. Our on-line classifier can accurately recognize actions an average of 10 frames before this peak. The right tail diminishes rapidly since few actions reach 100 frames in length.

horizontal lines on Figure 2 indicate the performance when each system has access to all of the frames in each video. Overall, our system performs quite well on this large set of actions with 98.33% accuracy, while the BoW and CRF classifiers only achieve an accuracy of 87% and 86% respectively.

This result validates our strategy of looking for "canonical" poses instead of trying to aggregate pose information over time. The BoW and CRF classifiers could 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 less than 15 frames each classifier performed poorly, but with more than 15 frames the performance of our system rises quickly. This can be understood by considering the range of movement observed in the beginning frames of the actions. Figure 3 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 the 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 caused by the low number of videos that have such a large number of frames.

7. Experiments with On-line Detection of Actions

While the temporally-segmented results are useful for understanding baseline performance, in real-world usage scenarios, the system will have to identify when actions are being performed. We focus on a particular sub-problem of the general on-line 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 Equation 4. The weights, θ, are the identical weights learned for the experiments in Section 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 video. If no probability exceeds T, the video is considered a missed detection and an error.

7.1. Modifying the Learning Criterion to Improve On-Line Detection Performance

A weakness of the approach described in the previous section is that the classification weights have been trained in the situation where the classifier can view all of the frames to make a decision. This is quite different from the on-line 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 adapted to better reflect the on-line detection task. 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 Section 6.2 with *maxFrames*. This is combined with the original loss to create the learning criterion for on-line detection, denoted as $L_{\textit{On-Line}}(\cdot)$,

$$L_{\textit{On-Line}}(\theta) = L_{\textit{Full}}(\theta) + \sum_{m \in M} (\gamma \cdot m) L_M(\theta) + \alpha R(\theta), \quad (10)$$

where $R(\theta)$ is the regularization term from Equation 6.

In this criterion, the loss computed over smaller time scales is added to the overall loss to add a bonus 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 confi-

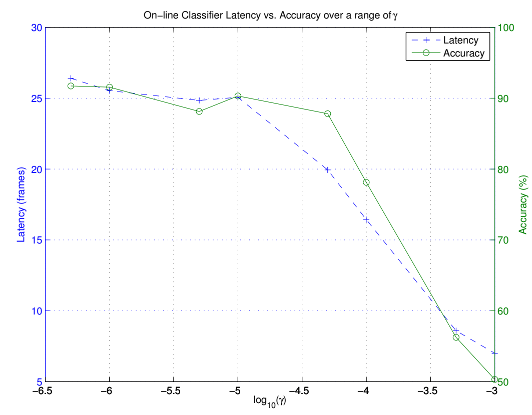


Figure 4. Latency compared with accuracy, evaluated on the testing set, for different values of γ. Accuracy peaks actions are classified at the 23rd frame on average. Beyond this, higher values of γ add too much loss for earlier classification and drive down both latency and accuracy. This is due in part to the lower amount of distinguishing information available in earlier frames, see Fig. 3.

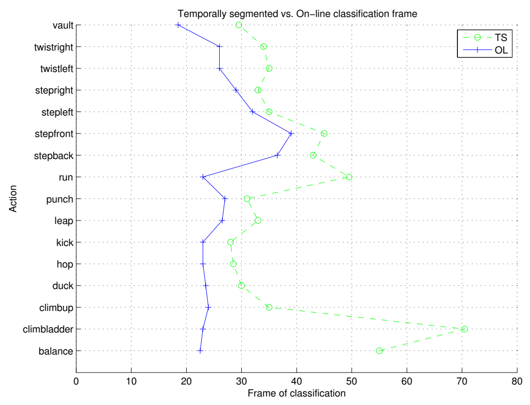


Figure 5. Comparison of frame of highest response from full video TS classifier with frame of classification from OL classifier. Recall that the TS classifier must look at the entire pre-segmented action to classify, so its frames correspond to the frames with the highest probability of being the correct action. The OL classifier frame is the earliest point that the probability of the correct action passes the threshold.

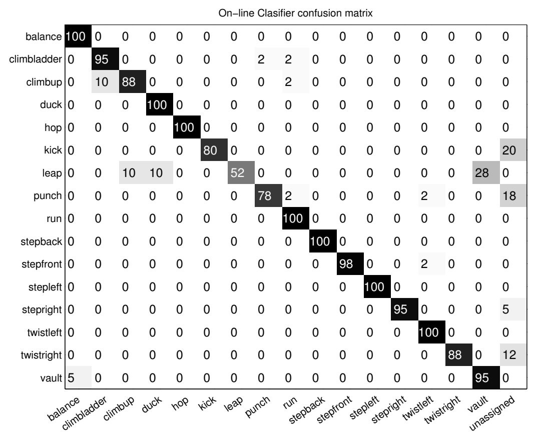
dent enough to make a classification. Figure 4 shows the relationship between the observational latency and system accuracy on the testing set for different values of γ in Equation 10.[1]

Figure 4 shows that as γ rises, the accuracy of the on-line detection system gradually decreases along with the latency. This indicates that the learning criterion in Equation 10 provides a parameter to tune the classifier between accuracy and latency. At the optimal γ, the system has an accuracy of nearly 92%. This compares well with the result from Section 6.2 as this task is much harder. It should also be pointed out that the on-line 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.

The reason for the drop in classification accuracy can be seen in Figure 5, which compares the median frame chosen by the classifier that can observe the entire temporally segmented action against the median frame chosen by the on-line detection system for each action class. As can be seen in this figure, the on-line 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 6.6%.

Figure 6 shows the confusion matrix in the on-line detection system. A column has been added for those actions where video has been mistakenly labeled as having no action.

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



On−line Clasifier confusion matrix

	balance	climbladder	climbup	duck	hop	kick	leap	punch	run	stepback	stepfront	stepleft	stepright	twistleft	twistright	vault	unassigned
balance	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
climbladder	0	95	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0
climbup	0	10	88	0	0	0	0	0	2	0	0	0	0	0	0	0	0
duck	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
hop	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
kick	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	20
leap	0	0	10	10	0	0	52	0	0	0	0	0	0	0	0	28	0
punch	0	0	0	0	0	0	0	78	2	0	0	0	0	2	0	0	18
run	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0
stepback	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
stepfront	0	0	0	0	0	0	0	0	0	0	98	0	0	2	0	0	0
stepleft	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
stepright	0	0	0	0	0	0	0	0	0	0	0	0	95	0	0	0	5
twistleft	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
twistright	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	0	12
vault	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	0

Figure 6. Confusion matrix for on-line classification with optimal γ. Punch is now more often unassigned than confused. Leap is confused with two actions which it is very similar to: duck and climb up. Overall accuracy remains high at nearly 92%.

7.3. Reducing Latency

Figure 4 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 on-line 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. When γ is too large, too much emphasis is placed on early classification, despite there not being a significant amount of variance in the data with which to accurately classify the action, so accuracy and latency both drop.

8. 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, previous, and first frame 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 98.33% accuracy.

We then adapted our model to an on-line 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 nearly 92%.

Acknowledgements

Marshall F. Tappen, Syed Zain Masood, Christopher Ellis and Adarsh Nagaraja 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

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

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

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

[4] L. Cao, Z. Liu, and T. Huang. Cross-dataset action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1998–2005, 2010.

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

[6] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2241–2248, 2010.

[7] P. Guan, A. Weiss, A. O. Blan, and M. J. Black. Estimating human shape and pose from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1381–1388, 2009.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning*, 2001.

[9] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE International Workshop on CVPR for Human Communicative Behavior Analysis*, pages 9–14, 2010.

[10] J. Liu and M. Shah. Learning human actions via information maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[11] metacritic. Fighters uncaged critic reviews. `http://www.metacritic.com/game/xbox-360/fighters-uncaged/critic-reviews`, February 2011.

[12] M. Narasimhan, P. A. Viola, and M. Shilman. Online decoding of markov models under latency constraints. In *ICML'06*, pages 657–664, 2006.

[13] J. Norton, C. Wingrave, and J. LaViola. Exploring strategies and guidelines for developing full body video game interfaces. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 155–162, 2010.

[14] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 271–278, 2005.

[15] K. Schindler and L. J. Van Gool. Action snippets: How many frames does human action recognition require? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

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

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

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 511, 2001.

[19] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, pages 137–154, 2001.

[20] W. Yang, Y. Wang, and G. Mori. Recognizing human actions from still images with latent poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2030–2037, 2010.

[21] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2442–2449, 2009.