

Electronic Theses and Dissertations, 2004-2019

2012

A Study Of Localization And Latency Reduction For Action Recognition

Syed Zain Masood
University of Central Florida

 Part of the [Computer Sciences Commons](#), and the [Engineering Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Masood, Syed Zain, "A Study Of Localization And Latency Reduction For Action Recognition" (2012).
Electronic Theses and Dissertations, 2004-2019. 2406.
<https://stars.library.ucf.edu/etd/2406>

A STUDY OF LOCALIZATION AND LATENCY REDUCTION FOR ACTION
RECOGNITION

by

SYED ZAIN MASOOD

B.S. Computer Science, Lahore University of Management Sciences, 2005
M.S. Computer Science, University of Central Florida, 2008

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2012

Major Professor: Marshall Tappen

© 2012 Syed Zain Masood

ABSTRACT

The success of recognizing periodic actions in single-person-simple-background datasets, such as Weizmann and KTH, has created a need for more complex datasets to push the performance of action recognition systems. In this work, we create a new synthetic action dataset and use it to highlight weaknesses in current recognition systems. Experiments show that introducing background complexity to action video sequences causes a significant degradation in recognition performance. Moreover, this degradation cannot be fixed by fine-tuning system parameters or by selecting better feature points. Instead, we show that the problem lies in the spatio-temporal cuboid volume extracted from the interest point locations. Having identified the problem, we show how improved results can be achieved by simple modifications to the cuboids.

For the above method however, one requires near-perfect localization of the action within a video sequence. To achieve this objective, we present a two stage weakly supervised probabilistic model for simultaneous localization and recognition of actions in videos. Different from previous approaches, our method is novel in that it (1) eliminates the need for manual annotations for the training procedure and (2) does not require any human detection or tracking in the classification stage. The first stage of our framework is a probabilistic action localization model which extracts the most promising sub-windows in a video sequence where an action can take place. We use a non-linear classifier in the second stage of our framework for the final classification task. We show the effectiveness of our proposed model on two well known real-world datasets: UCF Sports and UCF11 datasets.

Another application of the weakly supervised probabilistic model proposed above is in the gaming environment. 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 and thus significantly degrade the interactivity of the user experience. With slight modification to the weakly supervised probabilistic model we proposed for action localization, we show how it can be used for reducing latency when recognizing actions in Human Computer Interaction (HCI) environments. This latency-aware learning formulation trains a logistic regression-based classifier that automatically determines distinctive canonical poses from the 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.

I dedicate this work to my daughter Amsah, my wife Sana, my parents Masood and Rubina and my siblings Haani and Eema. Thank you for all your support.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Marshall Tappen, for his continued support and help is assisting me throughout my PhD program. I would also like to thank my labmates Kegan Samuels, Nazar Khan, Adarsh Nagaraja, Jiejie Zhu, Jason Hochreiter, Chris Ellis, Zhongkai Han and Hakan Boyraz for all the help they provided.

TABLE OF CONTENTS

LIST OF FIGURES	xii
LIST OF TABLES	xvi
CHAPTER 1 INTRODUCTION	1
1.1 Importance Of Application Of Localization	1
1.2 Localization Using A Weakly Supervised Probabilistic Model	3
1.3 Low Latency Action Recognition for Human Computer Interaction Systems	5
CHAPTER 2 RELATED WORK	8
CHAPTER 3 IMPORTANCE OF APPLICATION OF LOCALIZATION	12
3.1 Constructing a New Dataset to Understand the Effect of Background Complexity	13
3.1.1 Construction Choices	14
3.1.2 Construction Methods	15
3.1.2.1 Addressing Matting Artifacts	17

3.2	Baseline Method and Performance	17
3.2.1	Baseline: Basic Bag-of-Features Classifier	17
3.3	Measuring Performance Degradation	18
3.3.1	Unsuccessful Strategies For Dealing With Performance Degradation	19
3.4	Utilizing Action Localization For Handling Performance Degradation	20
3.4.1	Automatic Localization	22
3.4.2	Interest Points Pruning	23
3.4.3	Cuboid Correction	24
3.5	UCF Sports	27
CHAPTER 4	LOCALIZATION USING A WEAKLY SUPERVISED PROBABILIS-	
TIC MODEL		31
4.1	Action Localization and Recognition	32
4.1.1	Overview of Video Representation	32
4.1.2	Stage 1: Action Localization	34
4.1.3	Stage 2: Action Recognition	35
4.2	Learning The Model	37

4.2.1	Learning to Localize	37
4.2.2	Training The Second Stage Non-Linear Classifier	39
4.3	Learning a Conditional Random Field (CRF) Model	40
4.4	Video Feature Representation	42
4.5	Experimental Evaluation	43
4.5.1	Datasets	43
4.5.2	Results on UCF Sports Action Dataset	44
4.5.2.1	Localization Score	46
4.5.3	Results on UCF11 Action Dataset	48
4.5.4	Computational Performance	49
CHAPTER 5 LOW LATENCY ACTION RECOGNITION FOR HUMAN COMPUTER		
INTERACTION (HCI) SYSTEMS 51		
5.1	Basic Approach and Assumptions	53
5.1.1	Latency and Action Recognition	54
5.1.2	Defining and Measuring Observational Latency	55
5.2	Finding Poses with Multiple Instance Learning	57

5.2.1	Classifying Videos by Examining Individual Frames	59
5.2.2	Smooth Approximation	60
5.3	Dataset and Features	61
5.4	Experiments on Temporally Segmented Actions	64
5.4.1	Baseline Models	65
5.4.1.1	Bag of Words Model	65
5.4.1.2	Linear Chain CRF Model	66
5.4.2	Results for Temporally Segmented Actions	67
5.4.3	Benefits of Soft Approximation	71
5.5	Experiments with Online Detection of Actions	74
5.5.1	Modifying the Learning Criterion to Improve Online Detection Performance	75
5.5.2	Measuring Latency and Accuracy	77
5.5.3	Reducing Latency	80
5.6	Reducing Computational Latency	81
5.6.1	Examining the Boosted Features	82

5.7	Managing Accuracy and Latency by Reducing Possible Actions	83
5.8	Accuracy Results on Additional Datasets	87
5.8.1	Results on MSRC-12 Kinect Gesture Dataset	87
5.8.2	Results of MSR Action3D Dataset	88
CHAPTER 6	CONCLUSION	92
LIST OF REFERENCES	95

LIST OF FIGURES

Figure 3.1	Examples of the Weizmann (top row) and UCF Weizmann Dynamic (bottom row) datasets. Each video in the UCF Weizmann Dynamic dataset has a highly complex background. This indicates the background complexity of gradients, textures and contrasts on which the actions are overlaid.	15
Figure 3.2	Examples of the UCF Weizmann Dynamic dataset. The figure shows frames 1, 11, 21, 31 and 41 of 2 running actions with complex, dynamic backgrounds. The top row indicates running action overlaid on a background video with fast moving trees with high gradients and textures. Bottom row indicates running action overlaid on a slow moving eagle video. Care was taken not to have background videos with humans in order to isolate the effect of background motion as opposed to multiple human actions.	16
Figure 3.3	Top row shows the interest points without pruning for Weizmann and UCF Weizmann Dynamic datasets respectively. Bottom row shows the interest points for the same frame after pruning. For better recognition, it is thus important to remove background interest points.	29
Figure 3.4	The figure shows the effect of cuboid masking. Column 1: Shows the same running action performed by the same person matted on 3 different complex moving backgrounds. Column 2: Shows cuboids extracted from each video sequence. Size of each cuboid is 13x13x7, where all 7 frames are shown in a single row. Column 3: Illustrates the exact same cuboids as in column 2 after applying cuboid masking. Column 4: Shows the temporal gradients of cuboids in column 2. Column 5: Shows the temporal gradients of cuboids in	

column 3. The gradient in column 4 corresponding to background content (red outlined) appear different for each video sequence. However, the gradients of all three actions look similar after applying cuboid masking, as depicted in column 5. This is confirmed by average SSIM values of 0.67 and 0.75 for original temporal gradients (column 4) and cuboid masked temporal gradients (column 5) respectively.	30
Figure 4.1 Two stage action localization and classification model.	33
Figure 4.2 Given a video sequence, N_f sub-windows are extracted for each action class using Equation 4.9. Histograms are computed for the corresponding action class using the STIP descriptors within the sub-windows.	36
Figure 4.3 We show some of the localization results obtained using our method on the UCF Sports action dataset. The white square indicates the subwindow chosen by the localization algorithm to represent the action being performed in the video. A key advantage of our approach is that the localization is learned automatically from just the label of each video.	47
Figure 4.4 Confusion matrices for both UCF Sports and UCF11 dataset using LOOCV testing approach.	48
Figure 4.5 We show localization results obtained using our method on the UCF11 action dataset. We can see that our model is able to correctly identify a sub-window around the human actor as the best possible representation of the action being conducted in the video.	49
Figure 5.1 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.	58

Figure 5.2 Accuracy vs. Bag of Words and CRF over videos truncated at varying maximum lengths. The pose-based classifier proposed here achieves higher accuracy with less observations. 68

Figure 5.3 Confusion matrix for full video temporally segmented classification. Results shown are from uncropped action data. Overall accuracy achieved is 95.94%. 69

Figure 5.4 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. 72

Figure 5.5 Latency compared with accuracy, evaluated on the testing set, for different values of γ . Action accuracy is maximal at the 26th frame on average. As γ increases, latency is gradually reduced at the cost of decreased accuracy. 76

Figure 5.6 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. 78

Figure 5.7 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 Figure 5.3 to 85.78% above), we are able to achieve a significant drop (approx 66%) in classification latency. 79

Figure 5.8 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. 84

Figure 5.9 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. 86

LIST OF TABLES

Table 3.1 Comparison of our baseline and other state-of-the-art techniques on well known datasets. Comparable results on KTH and Youtube datasets shows robustness of our baseline approach.	19
Table 3.2 Comparison of our baseline and other state-of-the-art technique on the UCF Weizmann Dynamic dataset. We observe a significant drop in performance when switching from the Weizmann dataset to the UCF Weizmann Dynamic dataset.	20
Table 3.3 The above table shows the accuracy on UCF Weizmann Dynamic dataset when using interest point pruning with automatic localization. Best possible results for interest point pruning with ground-truth localization are also shown. Although results improve, they are still not comparable to those achieved on the Weizmann dataset using our baseline system (Table 3.2).	23
Table 3.4 The above table shows the accuracy on UCF Weizmann Dynamic dataset using combination of Interest Point Pruning (IPP) and Cuboid Masking (CM) w.r.t Automatic masks. We can see that optimal accuracy is achieved when using both IPP and CM strategies.	26
Table 3.5 The above table shows the accuracy on UCF Weizmann Dynamic dataset using combination of Interest Point Pruning (IPP) and Cuboid Masking (CM) w.r.t Ground truth masks. We can see that optimal accuracy is achieved when using both IPP and CM strategies.	26

Table 3.6	The table shows the results on UCF sports with automatic localization masks. It is evident that interest point pruning (IPP) and cuboid masking (CM) strategies improve the accuracy by 12%	28
Table 3.7	The table shows the results on UCF sports with ground-truth masks. It is evident that interest point pruning (IPP) and cuboid masking (CM) strategies improve the accuracy by 17%	28
Table 4.1	Mean per-class action recognition accuracies (split) on UCF Sport actions. We show that our method outperforms both global and results reported in [1] for UCF Sports dataset	45
Table 4.2	Mean per-class action recognition accuracies (LOOCV) on UCF Sport dataset. Our LOOCV results are comparable to the state-of-the-art on the UCF Sports dataset	46
Table 4.3	Mean per-class action recognition accuracies using 25 fold LOOCV on UCF11 dataset. We show improved performance over the global bag-of-words features	50
Table 5.1	The list of actions used in constructing the dataset.	54
Table 5.2	A tabular representation of the data from Figure 5.2. Note that our proposed method outperforms baselines even when they have access to more frames of pose information.	67
Table 5.3	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 2775, we can achieve recognition performance within 2% of our best temporally segmented result.	80
Table 5.4	GentleBoost vs All Features for online (OL) classification. We see the same trend as observed for the temporally segmented videos in Table 5.3. For the best possible value	

of γ , the boosted feature online classification system is within 1.7% of our original online result. 81

Table 5.5 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. 85

Table 5.6 This table compares the recognition achieved with our system against previous work on the MSR Action3D dataset from [2]. 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. 89

Table 5.7 This table shows the accuracy of the five least-recognized actions in the MSR Action3D dataset [2] 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. 91

CHAPTER 1 INTRODUCTION

1.1 Importance Of Application Of Localization

Given a video sequence, an action recognition system focuses on determining what type of action is being performed. In the recognition process, videos are typically treated holistically by aggregating video features together in a representation, such as bag-of-words. For single-person-simple-background datasets, such as Weizmann and KTH, only features pertaining to the action are detected as the background is simple and uninteresting. Thus, holistic systems perform almost perfectly on these simple datasets.

Having achieved near-perfect results on these simple datasets, the focus of the research community has shifted towards recognizing actions in more realistic and complex environments e.g. UCF Sports, UCF11 Youtube, Hollywood datasets. Generally, the problem of recognizing actions in these complex datasets is tackled using holistic recognition methodologies that work best for simple datasets. One of the benefits of such an approach is that it implicitly reasons about the context in which the action is being conducted. For example, the presence of waves in a video is a strong cue that the action is related to water activities.

While implicitly leveraging context can be helpful, the holistic treatment of an action video sequence makes it impossible for the system to separate the action from its context. The presence of irrelevant background information in complex action datasets makes the task extremely difficult. In this scenario, the knowledge of where the action is being performed

in the video, spatially as well as temporally, is essential in eradicating irrelevant background interest points and thus concentrate on action relevant features. In recent work, Lan et al. [1] have shown that action recognition can be improved by localizing the action and then considering only the features extracted from the localized sub-window within the overall video. Incorporating the ability to localize the action is shown to significantly improve the overall recognition accuracy. Additionally, one expects to achieve results comparable to those obtained on simple action datasets.

Although employing localization for eliminating background interest points helps improve results, it is not sufficient in achieving the best possible recognition accuracy. The reason being that spatio-temporal cuboid volumes extracted at action relevant locations are still corrupted by the complex background motion in the video sequences. Pruning is helpful in eliminating erroneous background interest points, but it fails to deal efficiently with irrelevant background information within selected interest point cuboids. Systems failing to address this issue are limited in performance [3, 4, 5, 6]. We show how the corruption within these cuboids can be removed by simple filtering steps, even with not-so-perfect automatic localization. Combined with the interest point pruning strategies, the system performs equally well on simple as well as complex datasets.

To understand how background complexity affects recognition accuracy, we introduce a new synthesized dataset that contains videos of simple actions on complex background. Using this dataset makes it easier to analyze how simply modifying background complexity influences results. We present our basic classifier method and show that it performs as well as state-of-the-art on well known datasets. However, it fails to perform equally well on the new synthesized dataset. We show how localization is imperative for achieving improved results on this dataset. Even using average automatic localization, we show how simple *but*

effective techniques like interest point pruning and correcting cuboid corruption lead to a significant improvement in results.

We focus on a bag-of-words systems, a very popular strategy for action recognition [7, 8, 9, 10, 11], where the classifier is based on quantizing image descriptors gathered at interest points and examining the frequency of different types of descriptors.

1.2 Localization Using A Weakly Supervised Probabilistic Model

Even though removing corruption within cuboids leads to improvement in results, it is not an ideal solution from a practical point-of-view. Reason being that, in order for proper functioning of such a system, the following requirements need to be addressed:

- Develop a mechanism of obtaining automatic localization of the action person.
- Even with perfect localization, best results are only possible using silhouette masks of the actor. Obtaining silhouette masks is a manual and highly cumbersome process for large scale datasets.
- As shown in Section 3.5, near-perfect localization on realistic datasets, like UCF Sports, is still unable to surpass the state-of-the-art results because background is highly discriminative and thus helps improve recognition performance.

It is important to note that the success of the above system is heavily dependent on localization. It is thus pertinent to construct a system that is able to localize as accurately as possible. [1] provide a localization technique that is shown to improve the recognition performance. However, training such a system requires manual annotation of actions for every frame in the training video set. This is feasible on a small dataset like UCF Sports

but is highly costly and cumbersome on larger, more complex datasets e.g. UCF11 action dataset. Additionally, the localization may be obvious given a small set of very distinct activities, but becomes more subjective as the number of action categories grow.

To counter this problem, we propose a a localization-based action recognition system that *automatically* localizes the action during the training process and thus eliminates the need for manual, ground-truth localization of the action. Our presented method is efficient as it eliminates the need for pre-processing heuristics and requires no human detector pre-processing steps. It is shown to significantly increases recognition accuracy, from 73.1% in [1] to 80.8% on the UCF Sports dataset.

The key insight in this work lies in how non-linear discrimination is incorporated into the system. Both [1] and our experiments in Section 4.5 show that it is difficult for linear models to simultaneously localize actions and discriminate between different action categories.

The response in [1] to this problem is to create a non-linear model for both localizing and discriminating by introducing a number of new latent variables that make it possible for the model to selectively ignore descriptors. Unfortunately, this approach expands the size of the search space for the latent variables in a fashion that affects the computation needed for all aspects of the system, including both training and testing.

In contrast, we propose that the limitation of the linear model can be solved by *separating* localization and discrimination. Our two-stage approach uses linear models to localize, then applies a non-linear classifier to recognize the action category. Because the stages are executed sequentially, no increase in the search space over latent variables is necessary. In addition, the sequential approach allows flexibility in the choice of the classifier. As Section 4.5 will show, the practical benefit of this is that our system produces higher accuracies, while not requiring ground-truth localizations in the training data. Our approach

also utilizes well-known, well-understood tools – making it easier to implement and apply in a variety of situations.

1.3 Low Latency Action Recognition for Human Computer Interaction Systems

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, which is discussed in Section 5.1.1, is a key issue for interactive experiences. A system that lags behind users' 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[12].

Traditionally, accuracy has driven the design of recognition systems. This work 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. This is 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.

One of the ways to address the latency problem is to enable the system to recognize the action as soon as possible. It is reasonable to suggest that each action sequence contains canonical body poses that clearly discriminate it from all other actions. Our goal is to find a canonical body pose for each action in as few observed frames of the video sequence as possible. Early classification on body poses might result in a lower latency but also a significantly lower accuracy. On the otherhand, selecting a canonical body pose too late might lead to higher accuracy but would also mean an uncomfortably high latency. Thus we need to maintain a healthy balance between accuracy and latency.

Rather than manually selecting key poses for each action as in [13], we present a novel Logistic Regression learning framework, similar to the weakly supervised linear probabilistic model proposed in Section 4.2. The system is designed to *automatically* find the most discriminative canonical body pose representation of each action and then perform 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 additional parameter-controlled costs 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. Experiments are conducted on a unique dataset collected using Microsoft Kinect which allows us to measure the latency due to the ambiguity involved in assuming a particular pose. Using the recently introduced Open-NI platform, we use this approach to implement a skeleton-based action recognition

system that recognizes 16 different actions. We show how this classifier can significantly outperform the baseline Bag of Words and Conditional Random Field (CRF) classifiers.

Additionally, we study the effects of reducing the feature count using a GentleBoost algorithm. We find that we can achieve similar classification accuracy by using a small subset of our initial features and thus reduce the computational latency of the recognition system. 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 two other datasets, namely MSRC-12 [14] and MSR Action3D [2]. We classify actions in MSRC-12 with high accuracy, along with most of the actions in the MSR Action 3D set. The failure cases in the actions in MSR Action3D set are analyzed in Section 5.8.

CHAPTER 2 RELATED WORK

A large literature on the problem of recognizing actions in videos has developed over the past decade. Weinland et al. [15] and Poppe [16] provide an overview of the various action recognition methods and datasets explored. Wang et al. [17] show comparisons of different methods on a variety of available well-known complex datasets.

Most action recognition systems are centered around a visual word representation for videos [7, 3, 18, 19]. Using these visual codebooks, some have suggested codebook refinement techniques for improved recognition results [20, 4] while others employ higher-order relations between visual words [21, 9, 10]. For recognition on complex datasets however, the fundamental problem is not only erroneous interest points due to background complexity but also the presence of background information within action relevant cuboids. Our proposed method of pruning irrelevant background interest points coupled with correcting cuboid corruption within action relevant interest point cuboids results in significant improvement in the recognition results.

One of the limitations of the our cuboid correction method is the dependency on action localization information. Without somewhat decent localization information, the results of our method can suffer. Although we use a combination of the human detector [22] and an image saliency detection method [23], it is imperative that we construct a formulation of *automatically* localizing the action within the video sequence.

Localization has been shown to be an important step for action recognition in complex environments [24, 25]. There have been efforts in the past that have focused on either using available person-location information or action detection prior to the task of recognition. Lan et al. [1] propose a figure-centric representation for action localization and recognition by treating person location as a latent variable and infer it while simultaneously recognizing the action. Yao et al. [26] classify and localize human actions in videos using a Hough transform voting framework. Amer et al. [27] formulate a generative chains model of group activities to localize and recognize group activities. Yuan et al. [28] propose and use a discriminative pattern matching technique to locate the action in the 3D video space using a branch-and-bound search mechanism. Boyraz et al. [29] propose a technique that transforms the 3D action localization problem into a series of 2D detection tasks. Lu et al. [30] propose a generative probabilistic model for concurrent action tracking and recognition. Ikizler et al. [6] employ a “tracking-by-detection” method in association with Felzenswalb’s human detector [22] for action detection.

Unlike [1, 24, 25, 26, 27, 28, 29, 30, 6], our method does not require manual annotation of the action person in the video. Instead we present a system that automatically localizes the action, eliminating the need for manual, ground-truth localizations, which may be costly to produce in large datasets. Our focus is on finding a per-frame sub-window within the video that best describes the action being performed and we will show how this best selected sub-window localizes on the human performing the action.

Another useful application of the above proposed method is real-time gesture recognition. We construct a framework that allows us to recognize an action sequence in as little time as possible. This helps reduce the latency in recognizing actions which is extremely useful for Human Computer Interactive (HCI) environments where lags and delays are cumbersome. Instead of finding the location of *where* the action is taking place we concentrate on

when the action is distinctly recognizable. In other words, in a frame by frame observations of an action video, we focus our efforts in determining which frame depicts information that discriminates the action from all other actions.

Our work is related to general action recognition systems [31, 2, 32, 33]. 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 as it is not real-time. Some systems perform temporal segmentation [28, 34], but these systems also assume that the action has already been recorded.

Efforts have been made in the past to try and extract key pose frames in action video sequences [35, 36] and use them for the task of action recognition. Carlsson et al. [13] present a recognition system that matches shape information of individual frames to prototype key frames. Zhao et al. [37] finds discriminative key frames that are used to weight features in a bag-of-words classifier. However, more recent work on action recognition has found better results using simpler bag-of-words representations [38], as discussed in Section 3.2.

In [39], Vahdat 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. [40] propose to learn weights for contour-based distinctive key poses and classify using a weighted voting system. Lv et al. [41] 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 [13] or the availability of accurate silhouette images [41, 40].

Techniques exist for reducing latency in sequential data, such as [42]. 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 [43, 44], is based on fitting Hidden Markov Models to different states in the gesture. An advantage of the system proposed in [43] 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 [45], 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 [46, 47]. 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 [48] 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 [43] 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 Section 5.5.

CHAPTER 3

IMPORTANCE OF APPLICATION OF LOCALIZATION

As discussed in Section 1.1, both accurate localization and its correct application are equally important for improved performance. The primary purpose of accurate localization is to differentiate between interest points pertinent to the action and those detected as a result of background motion in the video. Using this information helps eradicate irrelevant interest points and thus leads to improved results.

While the above application of localization is helpful, it does not lead to the best possible results that can be achieved. This is because of the presence of background information in action relevant interest points for out-of-place actions sequences. The term out-of-place refers to actions where the person moves as a whole with respect to the background e.g. running, walking, jogging, etc. As we will show below, using localization information in eliminating this background information from these ‘good’ interest points is what leads to the best possible recognition performance.

To investigate the effects of background clutter, we require accurate silhouette-level localization information. Such information is not readily available for the current complex datasets like UCF Sports, UCF 11, Hollywood, etc. For this purpose, we create a new synthesized complex dataset and show how application of localization for both action irrelevant interest point pruning and removing background information from within action relevant interest points leads to improved recognition performance.

3.1 Constructing a New Dataset to Understand the Effect of Background Complexity

In order to understand the effect of background complexity on recognition performance, we create a synthetic dataset with the aim of *isolating* the effects due to complex backgrounds. We do so by constructing synthetic videos of the same action being performed on different complex backgrounds. This way the difference in videos comes only from the background complexity.

To maintain focus on the problem of recognizing specific actions, we introduce a new synthetic complex dataset based on the Weizmann [49] dataset. This dataset is constructed by extracting action masks, provided on-line ¹, for each Weizmann dataset video and then replacing the background with a randomly selected Youtube video.

In establishing our reasoning for the construction of a new dataset, it is helpful to first consider the key properties of the Weizmann dataset. It contains a single actor performing simple periodic actions with simple fixed backgrounds. This construction forces the recognition system to focus directly on recognizing the action being performed by the actor. Also, the dataset allows us to control the quality of localization of the action being performed.

For this new synthesized dataset, the central recognition problem remains the same, but the task is made more difficult by the addition of the complex background. Essentially, our goal is to only modify one aspect, the background, during the recognition experiments.

¹<http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html>

3.1.1 Construction Choices

The Weizmann dataset was chosen because the actions are simple and coherent. In addition, each video has an associated action mask which makes it possible to extract the action and construct new videos with complex backgrounds.

We avoid the use of realistic complex datasets like Youtube [10, 4] and Hollywood [3, 8] because isolating the effect of background complexity from within the highly complex structure (multiple people, multiple actions, camera movement, high diversity within action class) of these datasets is extremely challenging.

We chose not to make a similar construction for the KTH dataset because the running and jogging actions in that dataset have not been recorded perfectly. Recent action recognition systems have near 100% accuracy on all actions except jogging and running [50, 10, 4, 3]. This is because the difference between these actions is not discernible for portions of this dataset, such as the videos from person 2.

To justify this decision, we conducted an experiment, involving humans, to gauge the difficulty of correctly recognizing actions between jogging and running. Each person was shown 2 training videos of each jogging and running and then was asked to correctly label a total of 50 test videos. Human subjects were only able to correctly recognize 90% of the jogging and running videos shown, which is approximately the same accuracy as the state-of-the-art. There was a high degree of agreement between human raters as to which videos are running and which are jogging. Additionally, most of the videos that the system misclassified were also incorrectly labeled by humans, highlighting problems in the dataset. The difficulty that humans have with running and jogging in this set makes it less desirable for evaluating machine vision systems.



Figure 3.1: Examples of the Weizmann (top row) and UCF Weizmann Dynamic (bottom row) datasets. Each video in the UCF Weizmann Dynamic dataset has a highly complex background. This indicates the background complexity of gradients, textures and contrasts on which the actions are overlaid.

3.1.2 Construction Methods

We create a new dataset using Weizmann action masks and background from Youtube videos. We downloaded a total of 15 Youtube videos making sure that each of them contain some complex scene. We then randomly select a Youtube video from this pool and perform matting with one of the Weizmann action mask. Keeping the Youtube video pool considerably lower than the number of action masks (93 in this case) ensures different actions being performed on the same background and thus diminishing the role of background in differentiating actions.

The dataset is developed using the following strategy:

- **UCF Weizmann Dynamic** The whole video is matted with the action mask (refer to Figure 3.2). The moving background makes it a much harder problem to recognize



Figure 3.2: Examples of the UCF Weizmann Dynamic dataset. The figure shows frames 1, 11, 21, 31 and 41 of 2 running actions with complex, dynamic backgrounds. The top row indicates running action overlaid on a background video with fast moving trees with high gradients and textures. Bottom row indicates running action overlaid on a slow moving eagle video. Care was taken not to have background videos with humans in order to isolate the effect of background motion as opposed to multiple human actions.

actions. This helps to analyze how increased background complexity affects recognition.

This new dataset will be made public and provided online ². It should be noted that when creating the dynamic set, we make sure that none of the Youtube backgrounds have humans in it. This is a necessity as the presence of humans in background videos is most likely to be accompanied by some action, leading to multiple actions in a single video. Our aim is to isolate the effect of background motion as opposed to multiple human actions and therefore we avoid using background videos with humans in them.

²<http://www.cs.ucf.edu/~smasood/datasets/UCFWeizmannDynamic.zip>

Our methodology of creating a complex dataset for simple actions is different from [34]. Our synthesized dataset is complete replica of the simple dataset in terms of the action being performed, and accuracy of the recognition can be compared directly. Since, we use matting [51] to create new dataset, it will not add any biases, due to change in the actor performing the action. Because of the synthetic construction of this dataset, matting artifacts could pose an issue and this is discussed next.

3.1.2.1 Addressing Matting Artifacts

To measure the effect of matting artifacts, we constructed a separate dataset by matting the Weizmann action masks with a simple static gray background. We found negligible ($\approx 4\%$) change in performance, making us confident that matting artifacts were not an issue.

3.2 Baseline Method and Performance

Having created this new synthesized dataset, we need to decide on a baseline system to be used. In this section, we explain the basic classifier approach we adopted and later evaluate its performance on both simple and complex datasets.

3.2.1 Baseline: Basic Bag-of-Features Classifier

We use a standard bag-of-features approach [7] as our baseline method. We make use of the code provided on-line¹. Given any video sequence, we detect spatio-temporal interest

¹<http://vision.ucsd.edu/~pdollar/toolbox/doc/>

points, extract cuboids centered around the interest points and compute gradient descriptor histograms. These histogram of gradients (HoG) are concatenated and Principal Component Analysis (PCA) is applied to project the gradients into lower dimensional space. Visual vocabulary is constructed using subset of the dataset followed by histogram representation for each video sequence. For classification, a Support Vector Machine (SVM) classifier ² is learnt using Histogram Intersection Kernel (HIK) and testing is done using leave-one-out-cross-validation (LOOCV).

Since the Weizmann dataset is relatively small, most research studies use the video reflection technique to double the size of the dataset [52]. This involves horizontally flipping each video and saving it as a new video. We use the same reflection approach for all our datasets.

The performance of this basic bag-of-features classifier as well as that of the state-of-the-art [3, 4] on different datasets is shown in Table 3.1. Despite being a simple technique, our baseline method performs reasonably well and is robust across different known datasets.

In the next section, we will discuss why performance degrades for these new synthesized complex datasets and what measures can be taken to improve results. Derived solutions are later tested on a realistic action dataset i.e. UCF Sports.

3.3 Measuring Performance Degradation

Having evaluated our basic classifier system on well known datasets, we now focus on how the system performs on the new synthesized dataset. Table 3.2 shows a comparison of the Weizmann and UCF Weizmann Dynamic datasets for our baseline system. We observe a

²<http://www.csie.ntu.edu.tw/~cjlin/libsvm>

Table 3.1: Comparison of our baseline and other state-of-the-art techniques on well known datasets. Comparable results on KTH and Youtube datasets shows robustness of our baseline approach.

Dataset	Our Baseline (Section 3.2)	STIPS (HOF) [3]	Liu et al. [4]
Original Weizmann	98%	92%	91%
KTH	93.5%	92%	93.8%
Youtube	65%	–	71.2%

sharp drop in accuracy when switching from the original to the newly synthesized dynamic dataset. Since the actions are exactly the same for both datasets, it is only logical to assume that the performance degradation is caused by the increased background complexity.

Before devising a new solution, we first try some of the well known strategies in order to achieve improved results. The next section details these methods and shows how they fail to solve the posed problem.

3.3.1 Unsuccessful Strategies For Dealing With Performance Degradation

A general approach towards solving this degradation in performance is to fine tune the system parameters. For this reason, we experimented using:

- different vocabulary sizes of 250, 500 and 1000 clusters
- averaging of features across different temporal scales [3, 53]
- cleaner vocabulary generated for Weizmann dataset

Table 3.2: Comparison of our baseline and other state-of-the-art technique on the UCF Weizmann Dynamic dataset. We observe a significant drop in performance when switching from the Weizmann dataset to the UCF Weizmann Dynamic dataset.

Dataset	Our Baseline (Section 3.2)	STIPS (HOF) [3]
Weizmann	98%	92%
UCF Weizmann Dynamic	36.5%	31%

- χ^2 kernel for SVM classification [17]

We achieved a maximum improvement of 2% using these techniques, thus failing to solve the particular problem that we pose here – recognition with complex backgrounds.

Background complexity plays a vital role when recognizing actions in videos. Even if the actions are simplistic, recognition systems performance is heavily dependent on the background they are performed on. In the next section we will discuss how the use of action localization goes a long way in rectifying this problem. It is no surprise that localization is helpful but, as will be shown below, it is the application of localization that is equally important.

3.4 Utilizing Action Localization For Handling Performance Degradation

We observed that the introduction of complex background in videos for simple actions greatly affects recognition performance (refer to Table3.2). Since the only change between the Weizmann and UCF Weizmann Dynamic datasets is of the background, it is reasonable to say that the drop in accuracy is only due to the change in background complexity. This is because

increased background complexity leads to detection of irrelevant background interest points that are a main source of performance degradation. One would assume that eliminating these background interest points should solve the problem. However, *that is not the case*. In fact, it is the use of localization for both pruning irrelevant interest points and eradicating background corruption inside cuboids that leads to optimal results. Thus we can say that:

- Action localization is important but
- Application/use of localization is equally significant

We propose a stepwise solution to the above posed problem:

- First and foremost, we need a good automatic action localization methodology (preferably a tight bounding box around the person performing the action).
- Once we have localization information, we eliminate all interest points detected due to background motion
- Having removed erroneous interest points, we use localization to remove cuboid corruption due to background information i.e. mask out background pixel values within valid cuboids.

Below, we will discuss each of the above strategies in detail. We will show how simply localizing the action and pruning irrelevant interest points is insufficient and that optimal results are achieved only when localization is directly used to modify the cuboids. Thus, these experiments will show that systems like [5, 6, 54] that use localization just to eliminate irrelevant interest points will have inferior performance compared with a system that uses localization information to also directly modify the cuboids.

We will build on the baseline system described in Section 3.2. To gauge performance of our system and to provide an upper bound on achievable accuracy, we will also present results obtained using ground-truth localization. Ground-truth localization masks are generated by forming a tight bounding box around the silhouette mask, available with the Weizmann dataset, at each frame.

Having analyzed and proposed solutions to the posed problem, we will show results on the UCF Sports dataset which is a commonly used complex action dataset in the vision community.

3.4.1 Automatic Localization

Since adding background complexity leads to significant increase in false positive interest point detections, it is imperative to design a system that accurately detects regions where the action is being performed. This is especially important for the UCF Weizmann Dynamic dataset where there is significant background motion. Once we have good localization of the action, discarding irrelevant interest points and modifying cuboids can be easily implemented. In reality however, such localization is hard to achieve for realistic datasets.

We combine an off-the-shelf human detection system [55, 56] and a saliency detection method [23] for obtaining automatic localization information of the action being performed. We employ the same technique when dealing with realistic UCF Sports dataset.

Table 3.3: The above table shows the accuracy on UCF Weizmann Dynamic dataset when using interest point pruning with automatic localization. Best possible results for interest point pruning with ground-truth localization are also shown. Although results improve, they are still not comparable to those achieved on the Weizmann dataset using our baseline system (Table 3.2).

Method	UCF Weizmann Dynamic
Our Baseline (Section 3.2)	36.5%
Automatic Localization + Interest Point Pruning	41%
Ground-truth Localization + Interest Point Pruning	68%

3.4.2 Interest Points Pruning

Directly running our baseline system on the UCF Weizmann Dynamic dataset results in interest points detected due to both the action and background motion. Having computed automatic localization information, we can now remove irrelevant background interest points. The goal is to discard all interest points lying outside the automatic localization mask calculated previously. This technique is applied at each frame of the action video sequence. With the removal of these background interest points, the recognition performance is expected to improve.

Figure 3.3 shows the interest points generated for the mentioned dataset. We see that almost all interest points in the Weizmann dataset are on or near the person performing the action. For the UCF Weizmann Dynamic dataset however, a significant number of interest points are due to background motion. It is essential that we remove these interest points for improved recognition accuracies. We thus prune interest points lying outside the automatic

localization masks generated for this dataset. It should be noted that these localization masks are in fact rectangular bounding boxes and so different from silhouette masks. After pruning, interest points for the Weizmann dataset remain the same. However, interest points from the UCF Weizmann Dynamic dataset are reduced by large extent (see Figure 3.3). Since pruning helps remove irrelevant interest points in the UCF Weizmann Dynamic dataset, we see improvement in recognition results (see Table 3.3). We also present the best possible recognition accuracy that can be achieved using ground-truth localization masks.

Although there is improvement in recognition accuracy for the UCF Weizmann Dynamic dataset, it is still not comparable to that achieved on the Weizmann dataset (even when using ground-truth localization). This can be attributed to the presence of background information within the cuboids extracted around the relevant interest points. This background is incorporated in the descriptor construction process and thus negatively affects performance.

In the next section, we will discuss actions that are more prone to the presence of background in extracted cuboids and how localization can be used to eliminate this irrelevant information.

3.4.3 Cuboid Correction

Previously, we showed how generating automatic action localization and using it to prune interest points helps improve recognition accuracy on the UCF Weizmann Dynamic dataset. However, the results obtained (refer to Table 3.3) are still not comparable to those achieved by baseline systems on the Weizmann dataset. In this section, we will explore

the problem further and show how eliminating background information from within relevant cuboids further improves results.

Out-of-place actions (e.g. running, walking) are more prone to be affected by complex backgrounds than in-place actions (e.g. bending, waving). Despite pruning interest points, cuboids may still contain background pixels; cuboids extracted near the mask boundary contain irrelevant spatial information while cuboids extracted for fast moving actions (such as legs of running and walking) contain temporal background information. To deal with this, we make use of localization masks by forcing all pixels of the extracted cuboids, that lie outside the localization bounding region, to a constant value. This helps *mask* out the irrelevant background pixel values, resulting in similar gradients across same actions in the descriptor construction phase. This modification to the cuboid is what helps in achieving optimal results for the UCF Weizmann Dynamic dataset.

An illustration of this is shown in Figure 3.4 for the UCF Weizmann Dynamic dataset. Each row shows the *same* running action performed by the *same* person on *different* dynamic backgrounds. The 2nd column shows some of the extracted cuboids of the corresponding video sequence while the 3rd column shows the same cuboids after applying cuboid masking. The 4th shows temporal gradients corresponding to column 2 while the 5th column shows temporal gradients corresponding to column 3.

For convenience, we highlight cuboid frames showing background pixels in column 2 through 5 with a red outlining. We observe that the background content in the cuboids (column 2) varies significantly for each video, leading to different temporal gradients (column 4) and eventually different descriptors. Although all 3 videos are of the same action, differences in background force systems to index these videos under different classes and thus decrease overall recognition performance.

Table 3.4: The above table shows the accuracy on UCF Weizmann Dynamic dataset using combination of Interest Point Pruning (IPP) and Cuboid Masking (CM) w.r.t Automatic masks. We can see that optimal accuracy is achieved when using both IPP and CM strategies.

Method	UCF Weizmann Dynamic
Our Baseline (Section 3.2)	36.5%
Automatic Localization + Interest Point Pruning	41%
Automatic Localization + Interest Point Pruning + Cuboid Masking	48%

Table 3.5: The above table shows the accuracy on UCF Weizmann Dynamic dataset using combination of Interest Point Pruning (IPP) and Cuboid Masking (CM) w.r.t Ground truth masks. We can see that optimal accuracy is achieved when using both IPP and CM strategies.

Method	UCF Weizmann Dynamic
Our Baseline (Section 3.2)	36.5%
Ground-truth Localization + Interest Point Pruning	68%
Ground-truth Localization + Interest Point Pruning + Cuboid Masking	89%

On the contrary, application of our cuboid masking technique handles this problem. Column 3 shows how all cuboid frames composed of background content are blackened out. As a result, temporal gradients associated with background information inside cuboids (column 5) are highly similar for each of the action video. This helps in assigning the same label for all 3 videos and thus improve recognition performance.

To strengthen our case, we measure the average structural similarity (SSIM) for temporal gradients with and without cuboid masking of all 3 videos shown in Figure 3.4. We found the average SSIM value to be 0.67 for the case without cuboid masking and 0.75 for

the case with cuboid masking. With higher SSIM score, it is evident that cuboid gradients are more similar after cuboid masking and hence improve the recognition results.

Tables 3.4 and 3.5 shows results associated with cuboid masking for both automatic and ground-truth localization. We see an improvement of 11.5% and 52.5% respectively over the baseline results. We can see that even with an average automatic localization method, we are able to achieve more than 10% improvement over the baseline performance. This is a significant jump in performance and shows how cuboid masking is able to handle complex static and dynamic backgrounds. With better localization techniques however, there is scope of even more improvement as depicted by the results obtained using ground-truth localization.

Having analyzed the problem using the synthesized dataset, we next test our system on a realistic dataset. Instead of Youtube [10, 4] and Hollywood [3, 8] datasets, we used the UCF Sports dataset for this task. The reason for this choice being that the UCF Sports dataset is more coherent with regards to the action categories as opposed to both Youtube and Hollywood datasets.

3.5 UCF Sports

In order to show that our suggestions are applicable to real life datasets, we test our system on the UCF Sports datasets. UCF sports dataset has the complex background and camera movement which were simulated in the synthetic dataset. At the same time, actions are more coherent and well captured unlike Youtube and Hollywood.

The results of different experiments on this dataset are presented in tables 3.6 and 3.7. We see that interest point pruning alone does not improve results but when combined

Table 3.6: The table shows the results on UCF sports with automatic localization masks. It is evident that interest point pruning (IPP) and cuboid masking (CM) strategies improve the accuracy by 12%

Method	UCF Sports
Our Baseline (Section 3.2)	68%
Automatic Localization + Interest Point Pruning	77%
Automatic Localization + Interest Point Pruning + Cuboid Masking	80%

Table 3.7: The table shows the results on UCF sports with ground-truth masks. It is evident that interest point pruning (IPP) and cuboid masking (CM) strategies improve the accuracy by 17%

Method	UCF Sports
Our Baseline (Section 3.2)	68%
Ground-truth Localization + Interest Point Pruning	79%
Ground-truth Localization + Interest Point Pruning + Cuboid Masking	85%

with cuboid masking, we see a 12% improvement over the baseline results. We also tested using ground-truth masks for the best possible results and observed a 17% improvement over the baseline results. Using either automatic or ground-truth localization, we observe that the application of localization for the purpose of interest point pruning is not sufficient. It is the use of localization to remove cuboid corruption that leads to significant improvement over the baseline method.

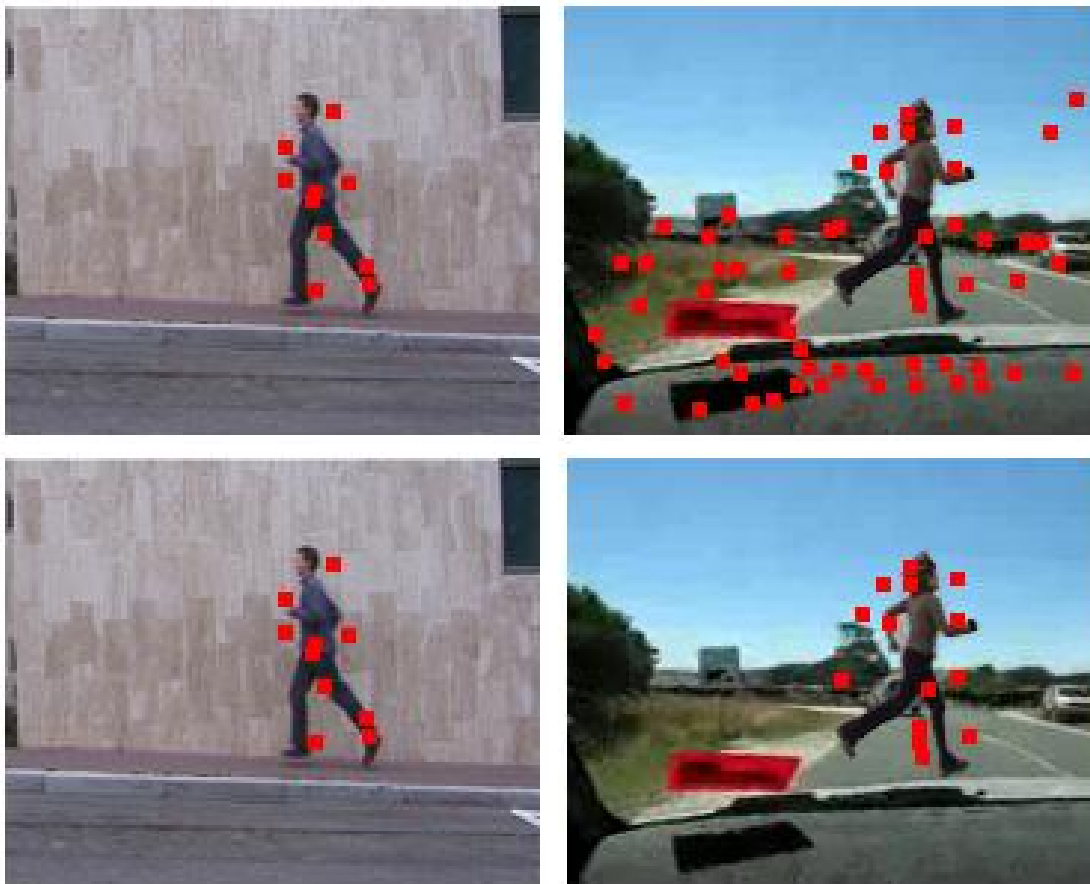


Figure 3.3: Top row shows the interest points without pruning for Weizmann and UCF Weizmann Dynamic datasets respectively. Bottom row shows the interest points for the same frame after pruning. For better recognition, it is thus important to remove background interest points.

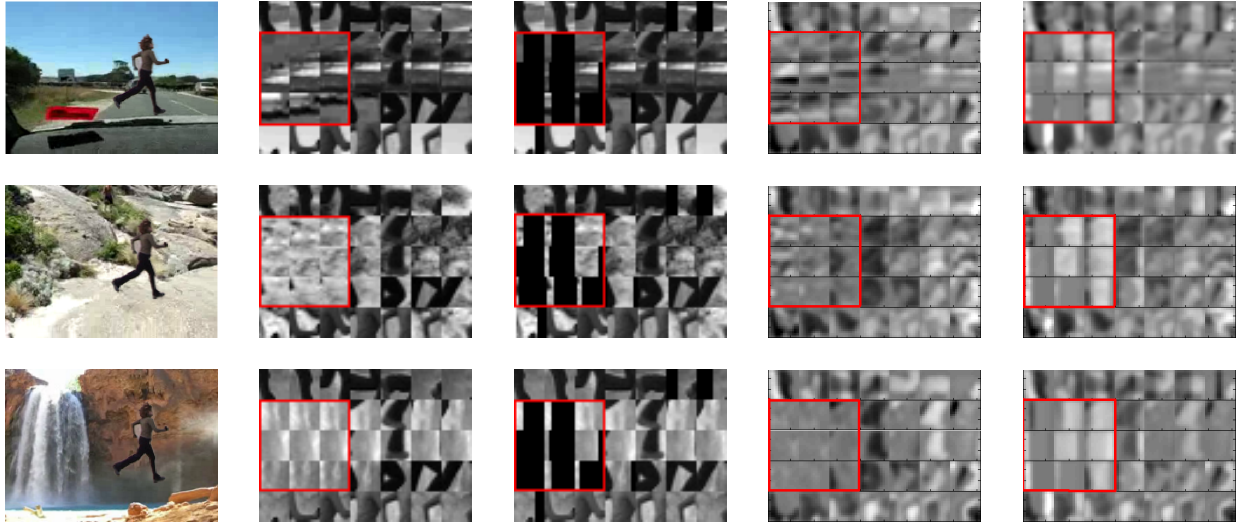


Figure 3.4: The figure shows the effect of cuboid masking. Column 1: Shows the same running action performed by the same person matted on 3 different complex moving backgrounds. Column 2: Shows cuboids extracted from each video sequence. Size of each cuboid is $13 \times 13 \times 7$, where all 7 frames are shown in a single row. Column 3: Illustrates the exact same cuboids as in column 2 after applying cuboid masking. Column 4: Shows the temporal gradients of cuboids in column 2. Column 5: Shows the temporal gradients of cuboids in column 3. The gradient in column 4 corresponding to background content (red outlined) appear different for each video sequence. However, the gradients of all three actions look similar after applying cuboid masking, as depicted in column 5. This is confirmed by average SSIM values of 0.67 and 0.75 for original temporal gradients (column 4) and cuboid masked temporal gradients (column 5) respectively.

CHAPTER 4 LOCALIZATION USING A WEAKLY SUPERVISED PROBABILISTIC MODEL

In the above chapter, we showed how proper application of localization for pruning irrelevant interest points and removing cuboid corruption within relevant ones significantly helps improve the recognition accuracy. However, improvement observed for realistic datasets (e.g. UCF Sports) was very limited in comparison to the improvement for the UCF Weizmann Dynamic dataset. In addition, even after employing near-perfect localization on UCF Sports, we were unable to surpass state-of-the-art results on this dataset. This observation can be attributed to the following reasons:

- Unlike the UCF Weizmann Dynamic dataset, we lack silhouette masks for realistic datasets. Although silhouette masks might improve results on realistic datasets, obtaining them is a manual and highly cumbersome process for large scale datasets.
- Since the background is highly discriminative for UCF Sports, it helps improve recognition performance and thus even near-perfect localization on UCF Sports is unable to surpass the state-of-the-art results.

Before we address the above concerns, it is important to note that the success of the above system is heavily dependent on accurate localization. Reliance on human detector [55, 56] and saliency methods [23] as a means of localization is not an ideal solution. It is thus pertinent to construct a system that is able to *automatically* localize as accurately as possible and thus eliminating the need for manual, ground-truth localization of the action.

Unlike [1] who create a non-linear model for both localization and discrimination by introducing a number of new latent variables, we propose a localization-based action recognition system that *separately* handles localization and discrimination. This is beneficial as the approach in [1] multiplicatively expands the size of the search space for the latent variables in a fashion that affects the computation needed for all aspects of the system, including both training and testing. On the otherhand, our two-stage approach uses linear models to localize, then applies a non-linear classifier to recognize the action category. Because the stages are executed sequentially, no increase in the search space over latent variables is necessary.

Also the system proposed in [1] requires the action be manually localized for every frame in the training video set. Our system *automatically* localizes the action during the training process and thus eliminates the need for manual, ground-truth localization of the action. Our presented method is efficient as it also eliminates the need for pre-processing heuristics and requires no human detector pre-processing steps. It is shown to significantly increase recognition accuracy, from 73.1% in [1] to 80.8% on the UCF Sports dataset.

4.1 Action Localization and Recognition

The following sub-sections will describe how an action is localized and recognized in a test video. Section 4.2 will discuss how the model is trained.

4.1.1 Overview of Video Representation

While Section 4.4 will discuss our video representation in more detail, here we give a brief overview of the video representation. Our descriptor representation is based on the

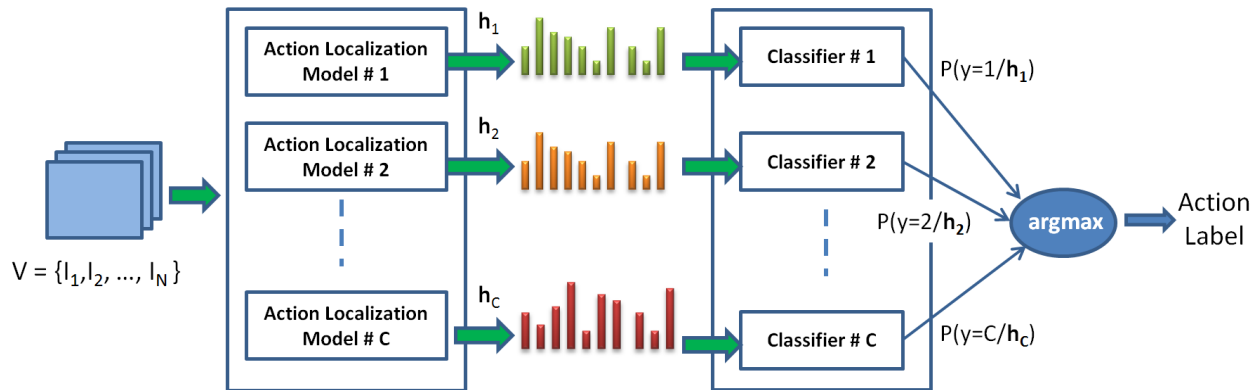


Figure 4.1: Two stage action localization and classification model.

popular Space-Time Interest Point (STIP) detector proposed by Laptev et al. in [57]. Using code provided by [57], we compute a dense representation of the video. Spatio-temporal descriptors are computed at regular intervals, both spatially and temporally, in the video. Building on the success of bag-of-words approaches, the descriptors are represented in a standard vector quantization representation. The descriptors are clustered to create a codebook and individual descriptors are replaced with the index of the closest descriptor in the codebook.

After this processing, the video is represented as a series of frames where each pixel in the frame holds a descriptor index value. Because the descriptor code computes these descriptors at regular spatial and time intervals, each frame in this new representation is a descriptor computed from several consecutive frames in the original video. This has the practical side effect of reducing both the size and number of frames in the video.

Figure 4.1 depicts our two-stage framework for recognizing the action in a video:

1. A series of action-specific localization models, one per action class, are used to find a set of sub-windows in the action that may contain the action being performed in the video.
2. A non-linear classifier, based on the histogram-intersection kernel, examines the sub-window associated with each possible action class and produces the final classification of the video.

4.1.2 Stage 1: Action Localization

The first step in our system is the localization of the action. For a particular action c , such as *running*, the action is localized in a frame by finding the sub-window that maximizes a response score, r_c :

$$r_c = \max_{w \in W} \vec{x}_w \cdot \theta_c \quad (4.1)$$

where W is the set of all possible sub-windows for a particular frame within the STIP video, \vec{x} denotes the feature within the sub-window w , and θ_c are the weights used to localize the action class c .

The set W of all possible sub-windows contains both sub-windows at all possible locations in the frame and also sub-windows of various sizes. In practice, we use full size, three-quarter-sized and half-sized sub-windows w.r.t. the frame size for our model computation.

As mentioned above, the feature vector describing each sub-window is based on STIP descriptors [57] that have proven successful in other action recognition systems. Building on the success of bag-of-words systems, each sub-window is represented by a histogram describ-

ing how many times different quantized descriptors appear in the window. For consistency purposes, the histograms were normalized with respect to the size of the subwindow used. Section 4.4 will discuss feature computation in more detail.

4.1.3 Stage 2: Action Recognition

The response scores computed during localization, as described in the previous section, can be used to classify the video by finding the class c^* such that the sub-window response r_{c^*} is maximized. However, since the model is linear, the recognition performance is likely to be lower than what we can achieve using a non-linear classifier. Thus, the second stage of our model uses a non-linear classifier to improve recognition accuracy.

For a given video sequence v , we predict the action label as follows:

1. For each frame in the video, determine the best scoring sub-window for each action label c . Each sub-window is represented by a histogram \vec{h}_c^f that contains the frequency of various quantized video descriptors in the highest-scoring sub-window in frame f for class c .
2. Aggregate the histograms across all frames. Formally, the histogram for action c is created by

$$h_c = \sum_{f=1}^{N_f} h_c^f, \quad (4.2)$$

where N_f is the total number of frames in the descriptor representation of the video.

Figure 4.2 gives a visual description of this step. For a given video sequence, we select the best scoring sub-window per frame for each of the action classes. Once we have all sub-windows for all classes, we construct histograms w.r.t each action class (as in

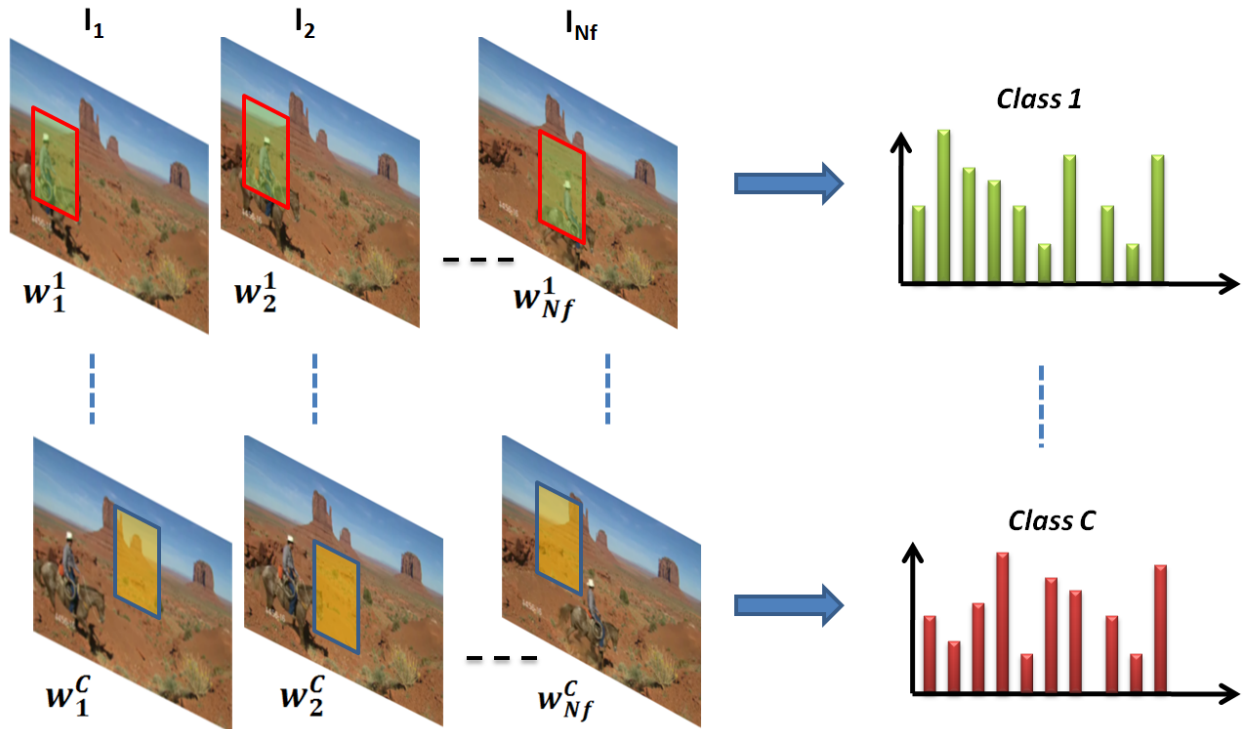


Figure 4.2: Given a video sequence, N_f sub-windows are extracted for each action class using Equation 4.9. Histograms are computed for the corresponding action class using the STIP descriptors within the sub-windows.

Equation 4.2) based on the features observed in the highest scoring sub-windows for that class, in the video sequence.

3. Use the histograms h_1, \dots, h_c to find the most likely class. In our current implementation, we use a set of support vector machines (SVMs), trained in a one-versus-all manner, to find the video's label. Each SVM, trained in the fashion described in Section 4.2.2, computes the probability $P(y = c|h_c)$, where y denotes the label of the current video. The video is assigned the action label that returns the highest probability score, i.e. $\arg \max_{k \in C} (P(y = k|h_k))$.

4.2 Learning The Model

Just as the recognition process occurs in two stages, the localization and recognition stages are trained in two different steps.

4.2.1 Learning to Localize

The first step is to learn the weights, $\theta_1, \dots, \theta_C$ in Equation 4.9 for localizing the action per frame. In this work, we assume that only the label of each video is provided. With just this information, the location of the action in each frame is treated as a latent variable and the weights are trained to optimize the ability of the system to discriminate between different action classes.

This is implemented using a probabilistic soft-max criterion. The probability of a frame f in a video sequence v belonging to the ground-truth class T is computed as:

$$P_f^v[l = T|\vec{x}] = \frac{\exp(r_T(\vec{x}^f))}{\sum_c \exp(r_c(\vec{x}^f))} = \frac{\exp\left(\max_{w \in W_f^v} \vec{x}_w^f \cdot \theta_T\right)}{\sum_c \exp\left(\max_{w \in W_f^v} \vec{x}_w^f \cdot \theta_c\right)}. \quad (4.3)$$

where \vec{x}^f contains all of the features in frame f and \vec{x}_w^f contains the features in sub-window w of frame f .

In this formulation, each frame in the video representation is independently classified with one of the action labels. This label is chosen based on the sub-window with the highest response.

Assuming that frames and videos are independent, the negative log-likelihood function reduces to the summation

$$L = - \sum_{v=1}^{N_v} \sum_{j=1}^{N_f} \log(P_f^v[l = T|\vec{x}]). \quad (4.4)$$

When implementing this learning process, it is useful to make an additional approximation. The max operation in Equation (4.10) makes it difficult to compute the gradient of the loss. This issue can be overcome by replacing the max operation with a smooth approximation. Given a set of values x_1, x_2, \dots, x_N , we can approximate the maximum of the set by using a differentiable approximation:

$$\max(x_1, x_2, \dots, x_N) \approx \log(e^{x_1} + e^{x_2} + \dots e^{x_N}) \quad (4.5)$$

Substituting this expression in Equation 4.10 we get:

$$P_f^v[l = T|\vec{x}^f] = \frac{\exp\left(\log\left(\sum_{w \in W_f^v} \exp(\vec{x}_w^f \cdot \theta_T)\right)\right)}{\sum_c \exp\left(\log\left(\sum_{w \in W_f^v} \exp(\vec{x}_w^f \cdot \theta_c)\right)\right)} = \frac{\sum_{w \in W_f^v} \exp(\vec{x}_w^f \cdot \theta_T)}{\sum_c \sum_{w \in W_f^v} \exp(\vec{x}_w^f \cdot \theta_c)} \quad (4.6)$$

It should be noted that the equations can also be derived from a strictly probabilistic view, but we find this perspective on the derivation intuitive.

This formulation makes it possible to train the localization weights $\theta_1, \dots, \theta_C$ using standard gradient-based techniques. In our experiments, we have had success with both non-linear conjugate gradient descent¹ and the stochastic meta-descent algorithm [58]. The stochastic meta-descent algorithm reduced optimization time by nearly half.

¹<http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/downloads/>

4.2.2 Training The Second Stage Non-Linear Classifier

Once the localization weights have been optimized, the classifier at the second stage can be trained. Assuming that our training dataset consists of V videos, C action classes and is represented using a k -sized visual codebook, we construct the data representation for the second stage of our model as follows:

- For each training video sequence $v \in V$ and the ground truth action label $c \in C$, we determine the the best scoring window on a per frame basis for action label c using the weights learned from the action localization model, i.e. $w_f^c(\vec{x}) = \arg \max_{w \in W_f} (\vec{x}_w \cdot \theta_c)$, where f is video frame and θ are the action localization weights.
- Using these best windows, we construct a k -sized histogram for video v w.r.t the ground truth action label c . In other words, each training video is represented as a feature histogram and corresponding ground truth action label.

Once we have the histograms for all training videos, we train a total of C one-vs-all non-linear classifiers, one for each action class. We select SVM with Histogram-Intersection-Kernel (SVM-HIK) as the non-linear classifier for this task. Histograms of the correct action class are labeled positive while the histograms of all other classes are labeled negative.

Since the SVM scores computed from different classifiers are not calibrated we use the probability estimates as the confidence score for each classifier. Probability estimates are computed by fitting a logistic regression model to the SVM output score. For binary classification problems, the probability of input \vec{x} belonging to label 1 is given by Equation 4.7, where parameters A and B are learned during the SVM training.

$$P(y = 1|\vec{x}) = \frac{1}{1 + \exp^{Af(\vec{x})+B}} \quad (4.7)$$

4.3 Learning a Conditional Random Field (CRF) Model

When analyzing human action sequences, we observe a temporal consistency between consecutive frames with respect to the location of the person performing the action. In other words, if we observe a person performing an action in frame ‘i’, we expect the person to be in the same vicinity in frame ‘i+1’. This is governed by our knowledge of the real world since we know that it is humanly impossible to move from one corner to the other within a *30th* of a second. This observation highlights a fundamental aspect of human movement in video sequences: location smoothness across time.

One of the drawbacks of the proposed approach is the lack of smoothness of the subwindow search across frames. In other words, since each frame is treated independently, there is no concept of a connection between adjacent frames within a video sequence. Thus, we see that the subwindow jumps from one corner of the video to the other in consecutive frames. This defies our knowledge of the real world and needs to be rectified.

One way of correcting this behavior is by introducing a smoothness term across frames. Subwindows in the next frame that are roughly in the same location as that of the current frame be given a higher weight as opposed to other subwindows. This way we can ensure a consistent behavior when localizing the action.

To handle this situation, we formulate a Conditional Random Field (CRF) model that optimizes the subwindow locations for the entire video sequence. Thus, the response scoring function is modified as follows: For a particular action c , the action is localized by finding a sequence of sub-windows, one per frame, that maximize the response score, r_c :

$$r_c = \max_{\vec{w} \in W_{All}} \sum_{f=1}^F (\vec{x}_{w_f} \cdot \theta_c + \alpha S(w_f, w_{f+1})) \quad (4.8)$$

where W_{All} is the set of all possible sub-windows permutations for a STIP video, \vec{x}_{w_f} denotes the feature for frame f within the sub-window w , α is the weight assigned to the smoothness term and S is a gaussian smoothness term for subwindows across adjacent frames defined as:

$$S(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{(i_x - j_x)^2 + (i_y - j_y)^2}{2\sigma^2}} \quad (4.9)$$

The new soft-max criterion is thus modified to reflect the above changes:

$$P^v[l = T|\vec{x}^v] = \frac{\exp(r_T(\vec{x}^v))}{\sum_c \exp(r_c(\vec{x}^v))} = \frac{\exp\left(\max_{\vec{w} \in W_{All}^v} \sum_{f=1}^{F^v} (\vec{x}_{w_f}^v \cdot \theta_T + \alpha S(w_f, w_{f+1}))\right)}{\sum_c \exp\left(\max_{\vec{w} \in W_{All}^v} \sum_{f=1}^{F^v} (\vec{x}_{w_f}^v \cdot \theta_c + \alpha S(w_f, w_{f+1}))\right)}. \quad (4.10)$$

Substituting the approximation to the maximum, we get:

$$\begin{aligned} P^v[l = T|\vec{x}^v] &= \frac{\exp\left(\log\left(\sum_{\vec{w} \in W_{All}^v} \exp\left(\sum_{f=1}^{F^v} (\vec{x}_{w_f} \cdot \theta_T + \alpha S(w_f, w_{f+1}))\right)\right)\right)}{\sum_c \exp\left(\log\left(\sum_{\vec{w} \in W_{All}^v} \exp\left(\sum_{f=1}^{F^v} (\vec{x}_{w_f} \cdot \theta_c + \alpha S(w_f, w_{f+1}))\right)\right)\right)} \\ &= \frac{\sum_{\vec{w} \in W_{All}^v} \exp\left(\sum_{f=1}^{F^v} (\vec{x}_{w_f} \cdot \theta_T + \alpha S(w_f, w_{f+1}))\right)}{\sum_c \sum_{\vec{w} \in W_{All}^v} \exp\left(\sum_{f=1}^{F^v} (\vec{x}_{w_f} \cdot \theta_c + \alpha S(w_f, w_{f+1}))\right)} \end{aligned} \quad (4.11)$$

Instead of localizing the action independently across frames, this new formulation ensures smoothness in adjacent frames. The added smoothness term penalizes subwindows from jumping randomly between consecutive frames and thus depicting a more realistic action localization scheme.

As done before, we train the localization weights $\theta_1, \dots, \theta_C$ using standard gradient-based techniques. Once the weights have been trained, we employ the second stage non-linear

classifier (Section 4.2.2) without any modification. As will be shown in Section 4.5, using the CRF model improves both the localization score as well as the recognition accuracy.

4.4 Video Feature Representation

The video regions are represented using Histogram of Gradient (HoG) and Histogram of Flow (HoF) descriptors computed at densely sampled interest points using Laptev’s STIP detector [57]. We then randomly select 100,000 descriptors from the training set and construct a visual codebook of size 4000 using the K-means clustering algorithm. Each STIP p_j is represented as the tuple (x_j, y_j, t_j, c_j) , denoting that a STIP was observed at (x_j, y_j) in the t_j ’th frame of the video; the label c_j corresponds to the index of the visual word in the codebook that is closest in feature space to p_j ’s descriptor.

The densely sampled interest point option for the STIP detector returns interest point locations based on the spatial and temporal scale size used. These locations are highly sparse, making it possible to compact the images and significantly reduce the amount of computation necessary to localize the action.

As shown in Equation 4.9, the action is localized using a score that is a linear function of the histogram of feature descriptors occurring in a window. An advantage of this function is that it can be computed efficiently using the integral image representation, similar to [59].

Using this technique, we can significantly reduce the size of the video data and thus improve efficiency of our model with respect to both time and memory usage. We will refer to this compact feature representation of a video as a STIP video whenever needed.

4.5 Experimental Evaluation

In this section we present results of our method described above on two well known real-world datasets and show the effectiveness of our proposed model at both localizing and recognizing actions in video sequences.

4.5.1 Datasets

We evaluate our model on two well known real world datasets, namely UCF sports dataset [60] and UCF11 dataset [4]. Both the UCF sports and UCF11 action datasets are very challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background and illumination conditions.

UCF Sports Action Dataset The UCF sports action dataset contains 150 video sequences and includes 10 human actions: swinging (on the pommel horse and on the floor), diving, golf swinging, horse riding, kicking (a ball), weight lifting, swinging (at the high bar), running, skateboarding and walking.

UCF11 Action Dataset The UCF11 action dataset contains 11 human actions: basketball shooting, bicycling, diving, golf swinging, horse riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking and walking with a dog. There are total of 1168 video sequences in the dataset with bicycling and walking with a dog actions having more videos than the rest. The videos are divided in a total of 25 folders per action class, where videos within a folder are treated as performed by the same person.

4.5.2 Results on UCF Sports Action Dataset

While it is common to use a Leave-One-Out-Cross-Validation (LOOCV) testing methodology when conducting experiments with the UCF Sports dataset, Lan et al. [1] have recently pointed out that many of the videos in this dataset are clips taken from a longer video. This is problematic when conducting LOOCV tests because several training clips will often be drawn from the same video as the testing clip. This causes the classifier to perform best when it effectively memorizes the appearance of the training clips to exploit the strong inherent context correlation among clips drawn from the same video segment.

In order to overcome this issue, [1] suggest using a third of the videos from each action class for testing while the remaining videos are used for training. We use the same train/test split².

Table 4.1 shows results using the train-test split suggested in [1]. The key results in this table include:

- Classifying the videos based on just the localization scores, as in Equation 4.11, performs comparably to a classifier based on a global representation of the image.
- However, using a linear SVM as a second stage classifier significantly improves classification accuracy by almost 8% to 72.3%. A likely reason for this is that treating the classification separately makes it possible for the classifier to focus on discriminating between action classes without having to compensate for the effect on how windows are localized.
- Utilizing a non-linear classifier based on the histogram intersection kernel produces another significant improvement to 80.8%.

²Available at http://www.sfu.ca/~tla58/other/train_test_split

Table 4.1: Mean per-class action recognition accuracies (split) on UCF Sport actions. We show that our method outperforms both global and results reported in [1] for UCF Sports dataset

Method	Accuracy
Global Bag-of-Words (SVM-Linear)	62.97%
Global Bag-of-Words (SVM-HIK)	68.81%
Lan et al.[1]	73.1
Our Method (Localization Model)	64.6%
Our Method (Localization Model + SVM-Linear)	72.3%
Our Method (Localization Model + SVM-HIK)	80.8%
Our Method (CRF Localization Model + SVM-HIK)	84.3%

- Using a CRF model, we were able to achieve better localization and improve the overall accuracy by 4% to 84.3%.

Our results using the linear SVM are comparable to those proposed in [1] with the added advantage of automatic action localization, i.e. no ground truth annotations are necessary for training. However, our best result using a CRF localization model with non-linear SVM-HIK shows significant improvement over both the baseline and results from [1]. It is important to note that we neither use ground truth action locations during training nor use any person detector/tracking algorithm for initial action location estimates when testing.

Figure 4.3 shows the localization results obtained using our proposed technique. We observe how the *automatic* localization model is able to accurately identify the actor in the video as the best representation of that action.

Table 4.2: Mean per-class action recognition accuracies (LOOCV) on UCF Sport dataset.

Our LOOCV results are comparable to the state-of-the-art on the UCF Sports dataset

Method	Accuracy
Kovashka et al. [21]	87.3%
Klaser [24]	87.3%
Wang et al. [17]	85.6%
Yeffet et al. [61]	79.3%
Rodriguez et al. [60]	69.2%
Lan et al. [1]	83.7%
Our Method	83.7%

We also performed experiments using the LOOCV method on the UCF sports dataset. The accuracies for the LOOCV method are provided in Table 4.2 and class specific breakdown is highlighted in Figure 4.4(a). We can see that our method is comparable to results reported by others.

4.5.2.1 Localization Score

Utilizing the ground-truth location information available for UCF sports dataset, we can compute a measure of how well our system is localizing. Using an evaluation criteria similar to the one in [1], we first compute the intersection-over-union score per frame using $\frac{Area(W_f \cap W_f^g)}{Area(W_f \cup W_f^g)}$ where W_f is the detection window in frame f and the W_f^g is the ground truth action bounding box. Then, we compute the average intersection-over-union score for a video

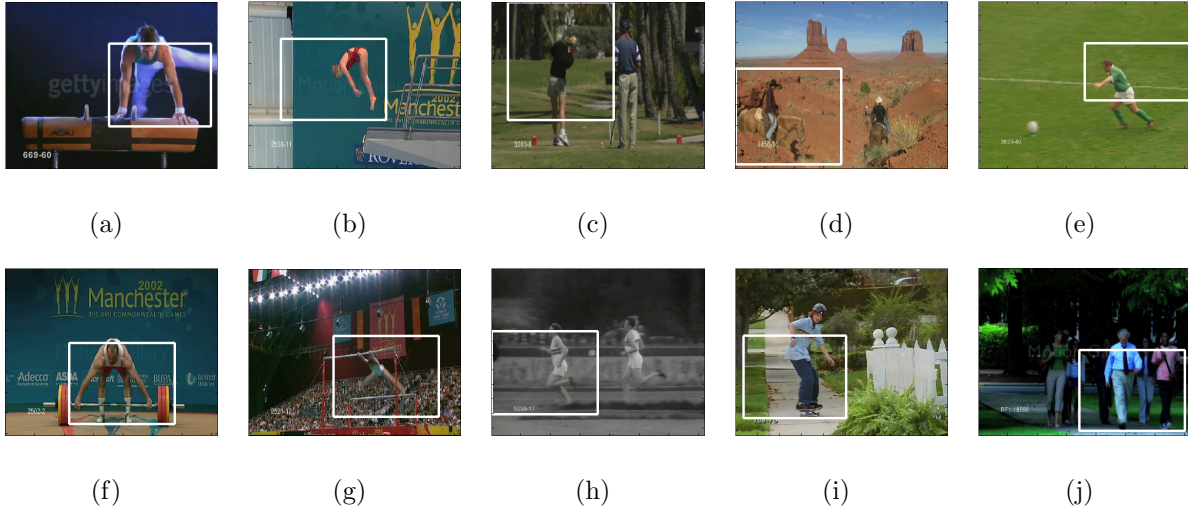


Figure 4.3: We show some of the localization results obtained using our method on the UCF Sports action dataset. The white square indicates the subwindow chosen by the localization algorithm to represent the action being performed in the video. A key advantage of our approach is that the localization is learned automatically from just the label of each video.

using all frames in the video. If this score is greater than a threshold σ , we consider the video to be correctly localized.

Using this measure with the $\sigma = 0.2$ used in [1], we were able to achieve true positive and false positive rates of approximately 38% and 42% respectively. However, once we implemented the CRF localization model, our localization results improved to 62% true positive rate and 50% false positive rate. The true positive rate is directly comparable to 64% reported by [1] while we achieve a lower false positive rate than the 60% reported in [1]. It should be noted that our system achieves a better localization without having access to ground truth localization information during training. We consider this to be a strong result given that our system is just trained with overall class labels.

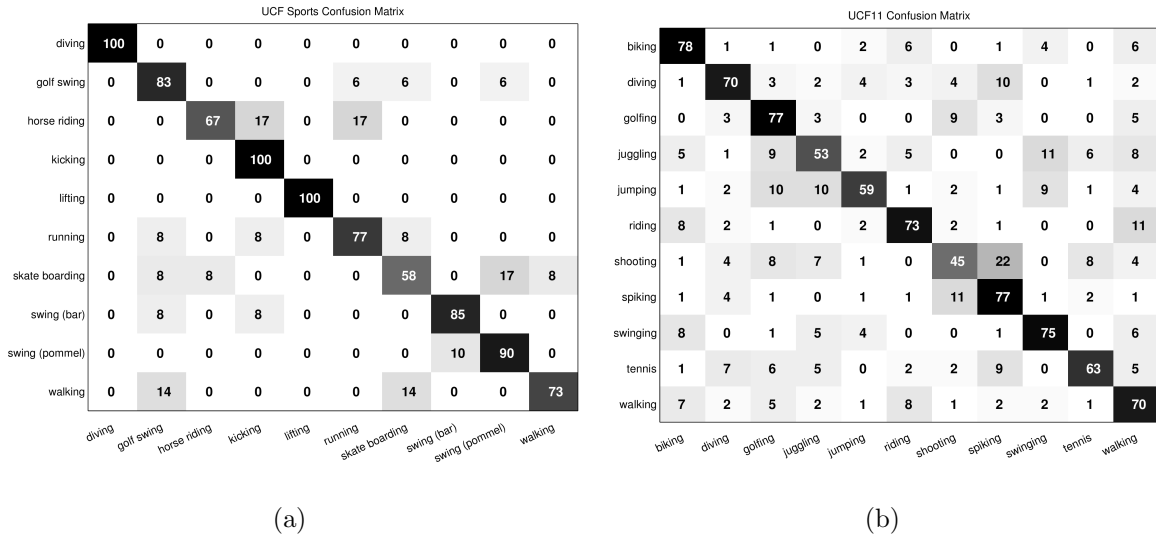


Figure 4.4: Confusion matrices for both UCF Sports and UCF11 dataset using LOOCV testing approach.

4.5.3 Results on UCF11 Action Dataset

With the success of our model on the UCF Sports action dataset, we perform a similar experiment on the UCF11 action dataset. We use a 25 fold LOOCV technique as suggested in [4]. The results obtained for both the baseline global bag-of-words representation as well as our method are listed in Table 4.3, with a more detailed confusion matrix shown in Figure 4.4(b). We again see that using our localization model, we can improve recognition performance over the baseline by 4%. Figure 4.5 shows sample localization results that demonstrate how our localization model is able to correctly identify the action sub-window within the video.

It should be noted that the final accuracy of our system is below that reported by [62]. This is largely due to our choice of video features. In [63], Oh et al. have observed that

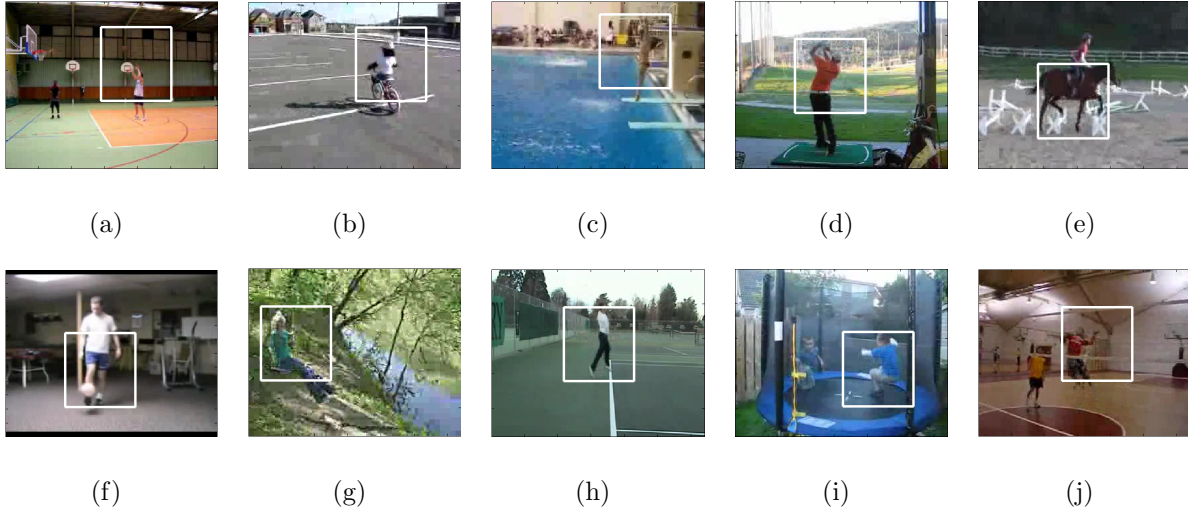


Figure 4.5: We show localization results obtained using our method on the UCF11 action dataset. We can see that our model is able to correctly identify a sub-window around the human actor as the best possible representation of the action being conducted in the video.

STIP-based descriptors are more accurate for high resolution sequences, like the UCF Sports dataset, but not well-suited for low resolution videos, which dominate the UCF11 dataset.

4.5.4 Computational Performance

The STIP video representation explained in Section 4.4 improves the efficiency of our model with respect to both time and memory requirements. The sub-window scores can be computed effectively for a STIP frame using an integral image. In order to show the efficiency of our model we provide some timing information for training and testing on the UCF sports action dataset: The average time for computing the gradient term for a single STIP video is around 36.5 ms with a 3.3 GHz processor. A single iteration of the conjugate gradient-descent algorithm on the whole split training dataset takes around 15 seconds and

Table 4.3: Mean per-class action recognition accuracies using 25 fold LOOCV on UCF11 dataset. We show improved performance over the global bag-of-words features

Method	Accuracy
Global Bag-of-Words (Linear SVM)	58.82%
Global Bag-of-Words (SVM-HIK)	63.36%
Our Method	67.77%

the algorithm converges within 30 iterations, therefore the overall training time for the first stage of our model is around 450 seconds. During the testing phase, it takes around 106.38 *ms* for classification of a STIP video. This can be improved further since the action location hypothesis and classification scores can be computed in parallel for each action class as shown in Figure 4.1.

CHAPTER 5

LOW LATENCY ACTION RECOGNITION FOR HUMAN COMPUTER INTERACTION (HCI) SYSTEMS

The weakly supervised probabilistic model proposed in Chapter 4 can be used in a variety of problems. In this chapter, we will discuss how this formulation can be used to reduce latency in recognizing actions for Human Computer Interaction (HCI) systems, more specifically gaming environments.

The two fundamental characteristics of successful HCI systems are:

1. High Accuracy - The system must be accurate at recognizing actions.
2. Low Latency - Latency, which is discussed in Section 5.1.1, 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[12].

As opposed to traditional recognition systems that focus solely on achieving high accuracy, we formulate a system that couples accuracy and latency in recognition. There are two main categories of latency that systems encounter:

- *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,
- *Computational Latency*, which is the latency caused by the recognition system itself.

Our primary focus is towards observational latency but we also propose methods to handle computational latency. In general, 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.

A solution to reducing observational latency is enabling the system to recognize the action with as few observations as possible. In other words, we want to design a system that recognizes an action based on some canonical body poses that easily discriminate the action from all other actions. The goal is to encourage the system to find a discriminating canonical pose in as few observed frames of the video sequence as possible. We have to be mindful of the fact that classifying an action too soon might lead to a significantly lower accuracy and so need the system to strike an agreeable balance between accuracy and latency.

We previously used the probabilistic model proposed in Chapter 4 for determining the spatial location of *where* the action is being conducted in a video sequence. We will modify the formulation slightly so that it focuses on *when* the action is discriminative i.e. finding a canonical pose from a sequence of frames that best describes the action. For reducing latency, we introduce additional parameter-controlled costs that force 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.

Experiments are conducted on a unique dataset collected using Microsoft Kinect which allows us to measure the latency due to the ambiguity involved in assuming a particular pose. We show how this classifier can significantly outperform the baseline Bag of Words and Conditional Random Field (CRF) classifiers. We also evaluate the performance of our algorithm against both the MSRC-12 [14] and MSR Action3D [2] datasets and discuss the results in detail.

5.1 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 [64, 33].

Each video in the dataset consists of one person performing one action, from a set of 16 actions, a single time.¹ Figure 5.1 lists the set of actions used. These actions are chosen based on experiments in [65], which used the game *Mirror's Edge* to identify a set of actions which would be natural for an interactive gaming experience. Section 5.4.2 reports results for simultaneously distinguishing between all 16 actions. Although the set of actions is not as extensive as [48], it is still substantially larger than in other previous work such as [2].

Each action is performed starting from a rest state, making it possible to measure how quickly the action is recognized from this rest state. Collecting data in this fashion is reasonable because the vast majority of the actions, which have been chosen through user studies in [65], require returning to a rest pose. In the set of 16 actions in Table 5.1, 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 [66]. This is because, at the time we began, previous datasets had not been gathered in a fashion that makes it possible to measure the recognition latency from the moment the

¹See Section 5.3 for more details on the data gathering process.

Table 5.1: 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

human begins the action. However, recently new datasets have become available, and we present our results in Section 5.8.

5.1.1 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. 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 [67, 68, 69]. In Section 5.6, we show how a GentleBoost method [70] 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.

5.1.2 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 [52], and the work of Davis and Tyagi [44], 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 [44] or [52]. 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 Section 5.2 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 [52]) 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.

5.2 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 [52]. 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 Section 5.1.2.

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 Section 5.3. The underlying idea behind the classifier is that the action can be reliably recognized when the user assumes a distinctive pose that unambiguously characterizes the action. As demonstrated in Section 5.4.2, 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 Figure 5.1, 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.

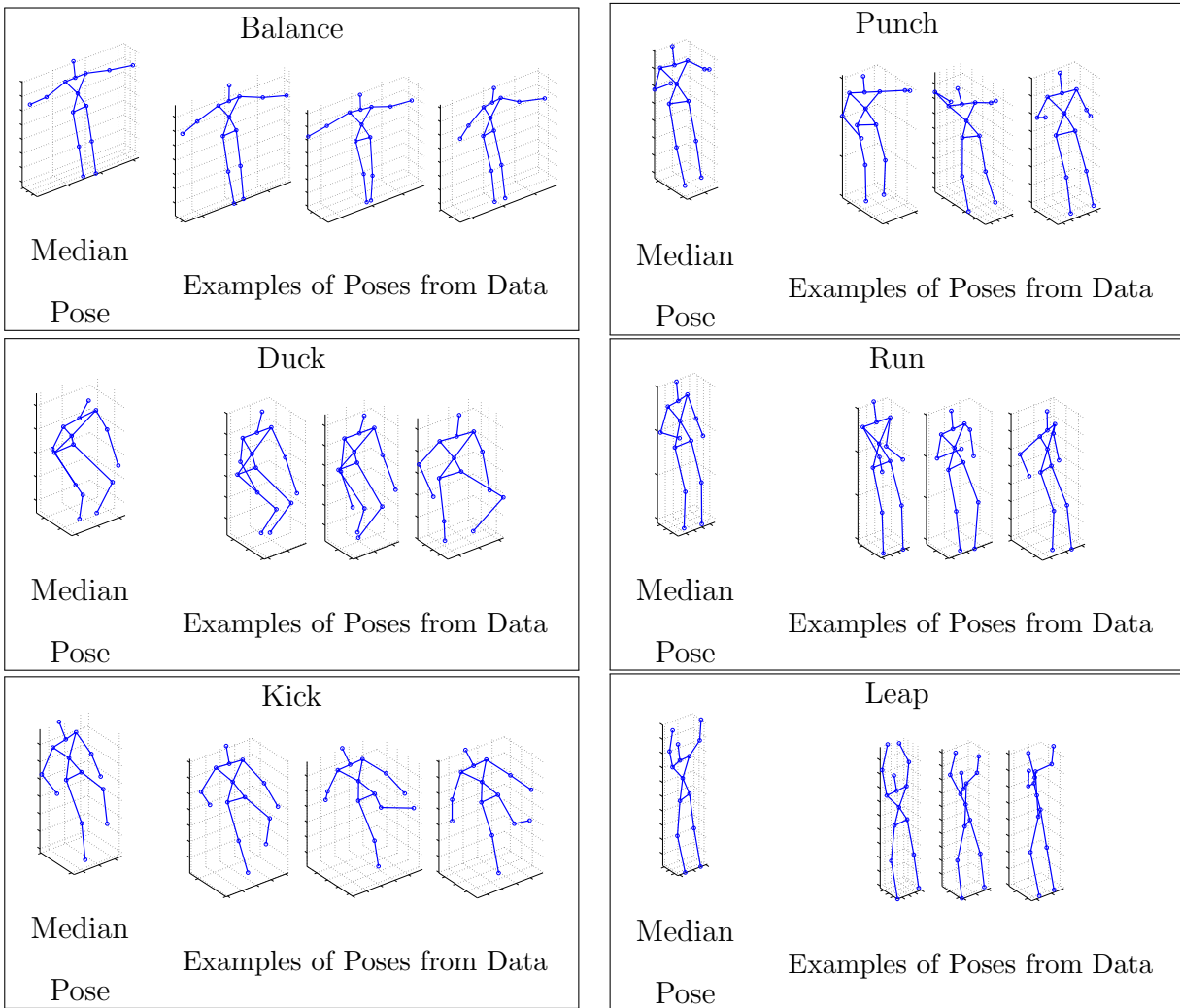


Figure 5.1: These skeletons shows several of the poses associated with different actions. The skeleton on the left of each panel is the median of poses associated with each action. The skeletons on the right are examples of poses considered to be most like the canonical pose in a particular video.

5.2.1 Classifying Videos by Examining Individual Frames

In our dataset, discussed in detail in Section 5.3, 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 :

$$r_c(\vec{x}) = \max_{f \in F} \vec{x}_f \cdot \theta_c \quad (5.1)$$

where F denotes the set of all frames in the bag and \vec{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:

$$\begin{aligned} P[l = T | \vec{x}] &= \frac{\exp(r_T(\vec{x}))}{1 + \sum_c \exp(r_c(\vec{x}))} \\ &= \frac{\exp\left(\max_{f \in F} \vec{x}_f \cdot \theta_T\right)}{1 + \sum_c \exp\left(\max_{f \in F} \vec{x}_f \cdot \theta_c\right)}. \end{aligned} \quad (5.2)$$

Adding a 1 to the denominator of Equation (5.2) 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 [67].

5.2.2 Smooth Approximation

While logistic regression models are typically trained using gradient-based optimization, the introduction of the max operator in Equation 5.2 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

$$\max(v_1, v_2, \dots, v_N) \approx \log(e^{v_1} + e^{v_2} + \dots e^{v_N}). \quad (5.3)$$

Incorporating this approximation into Equation 5.1 leads to the following expression for computing the probability of a particular class:

$$\begin{aligned} P[l = T|\vec{x}] &= \frac{\exp\left(\log\left(\sum_{f \in F} \exp(\vec{x}_f \cdot \theta_T)\right)\right)}{1 + \sum_c \exp\left(\log\left(\sum_{f \in F} \exp(\vec{x}_f \cdot \theta_c)\right)\right)} \\ &= \frac{\sum_{f \in F} \exp(\vec{x}_f \cdot \theta_T)}{1 + \sum_c \sum_{f \in F} \exp(\vec{x}_f \cdot \theta_c)}. \end{aligned} \quad (5.4)$$

As an aside, we note that in Equation 5.4, the sharpness of the max approximation could be tuned using a scaling parameter as: $\max(v_1, \dots, v_N) \approx \log(e^{kv_1} + \dots + e^{kv_N})$.

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 $\vec{x}_1, \dots, \vec{x}_{N_T}$ 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 Equation 5.4. 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:

$$R(\theta_i) = \alpha \log(1 + \beta \theta_i^2), \quad (5.5)$$

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.

5.3 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

²The dataset has been made publicly available at <http://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. It may prove useful to make use of confidence values 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, 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 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.

5.4 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 5.4, 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 Section 5.3, 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 $\theta_1, \dots, \theta_{N_a}$ according to the method described in Section 5.2.1.
3. For each action class, $c \in \{1, \dots, N_A\}$, find the feature vector $\vec{x}_{f_c^*}$ such that $\vec{x}_{f_c^*} \cdot \theta_c$ 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

$$c^* = \arg \max_c \vec{x}_{f_c^*} \cdot \theta_c. \quad (5.6)$$

For evaluation, we used a set of data gathered from 16 people (as discussed in Section 5.3). All of our experiments are implemented using 4-fold cross-validation.

Figure 5.1 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 Figure 5.1 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.

5.4.1 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 Equation 5.5.

5.4.1.1 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 [9]. While Zhao et al. [37] propose extensions for the BoW model that use key frames, we use the original BoW model because recent work [38] has shown that in direct comparisons on KTH, the original BoW outperforms the variant proposed in [37].

In our baseline, the BoW employs the same distances described in Section 5.3, 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 a Support Vector Machine with Histogram Intersection Kernel (SVM-HIK).

5.4.1.2 Linear Chain CRF Model

The Conditional Random Field (CRF) model [71] 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 Equations (5.2) and (5.4). However, in this case, the probability is computed using a function $C_k(\vec{y}; \vec{x})$ that expresses the cost of a sequence of hidden states, expressed in the vector \vec{y} , given the observation \vec{x} .

Following Section 5.2.2, the probability of a particular class is expressed as

$$p[l = T|x] = \frac{\exp \left\{ \min_y (-C_T(y; x)) \right\}}{\sum_k \exp \left\{ \min_y (-C_k(y; x)) \right\}} \quad (5.7)$$

$$\approx \frac{\exp \left\{ -\log \sum_y \exp (C_T(y; x)) \right\}}{\exp \left\{ -\log \sum_k \sum_y \exp (C_k(y; x)) \right\}}. \quad (5.8)$$

The function $C_k(\vec{y}; \vec{x})$ is constructed to be a typical chain-structured CRF model, with pairwise Potts model potentials [72] 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 pose. While it could be expected that the more detailed CRF model could

Table 5.2: A tabular representation of the data from Figure 5.2. 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

lead to better recognition performance, our results clearly demonstrate the advantages of focusing on a single, reliably occurring, highly discriminative pose.

5.4.2 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

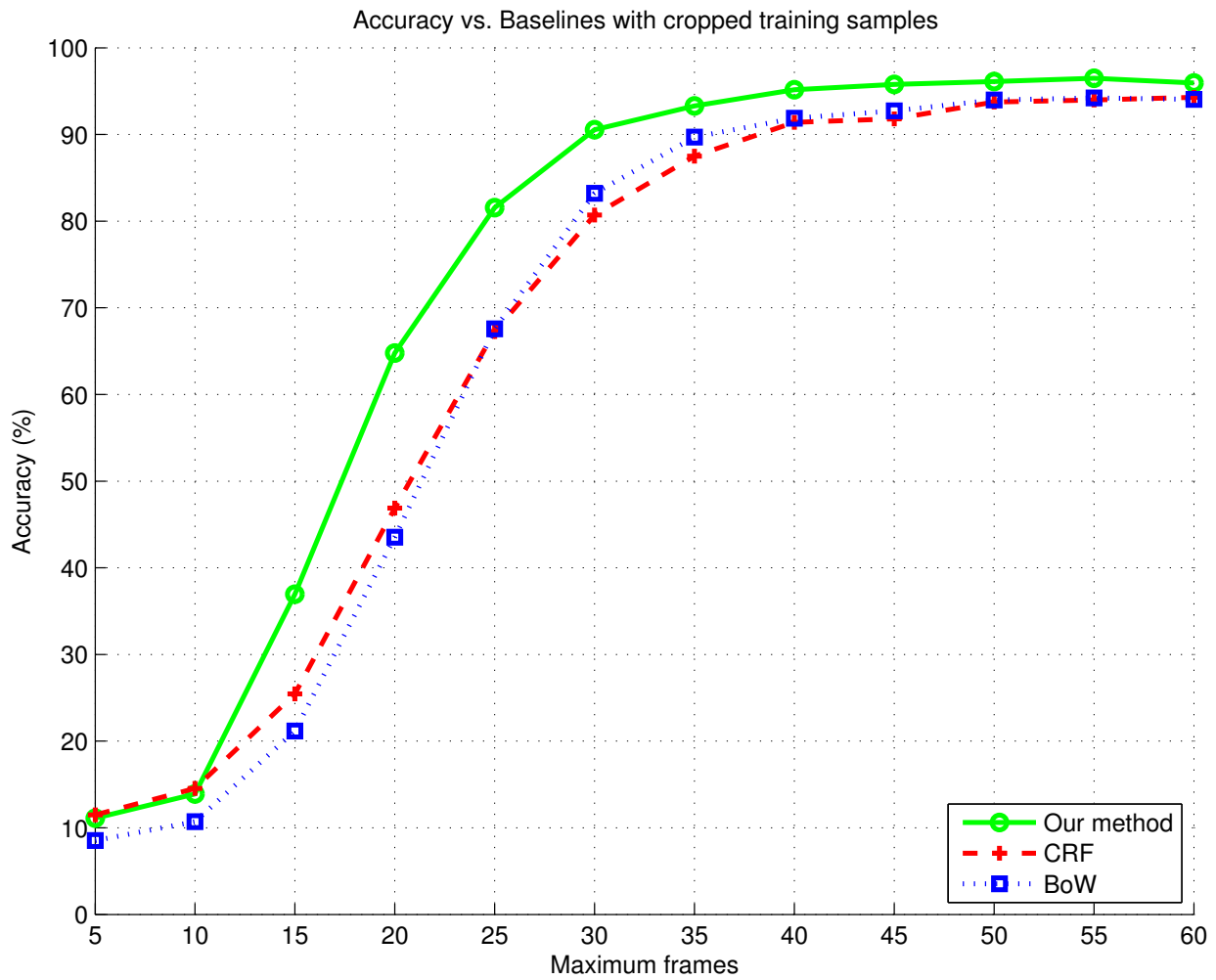


Figure 5.2: Accuracy vs. Bag of Words and CRF over videos truncated at varying maximum lengths. The pose-based classifier proposed here achieves higher accuracy with less observations.

Full Video Temporally Segmented Classifier confusion matrix

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

Figure 5.3: Confusion matrix for full video temporally segmented classification. Results shown are from uncropped action data. Overall accuracy achieved is 95.94%.

three frames. On the other hand, the BoW and CRF classifiers use the feature vectors from all *maxFrames* frames.

As shown in Figure 5.2, 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 5.2 shows numerical results at several points along the curves in Figure 5.2. 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.

Figure 5.3 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 5.2 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 5.4 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 caused by the low number of videos that have such a large number of frames.

Figure 5.2 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.

5.4.3 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 [67]. In this work, we use a soft approximation of the max operator during the training process while the training system in [67] 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 Section 5.2, we also implemented a coordinate descent approach, similar to [67], 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 Equation (5.4). 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

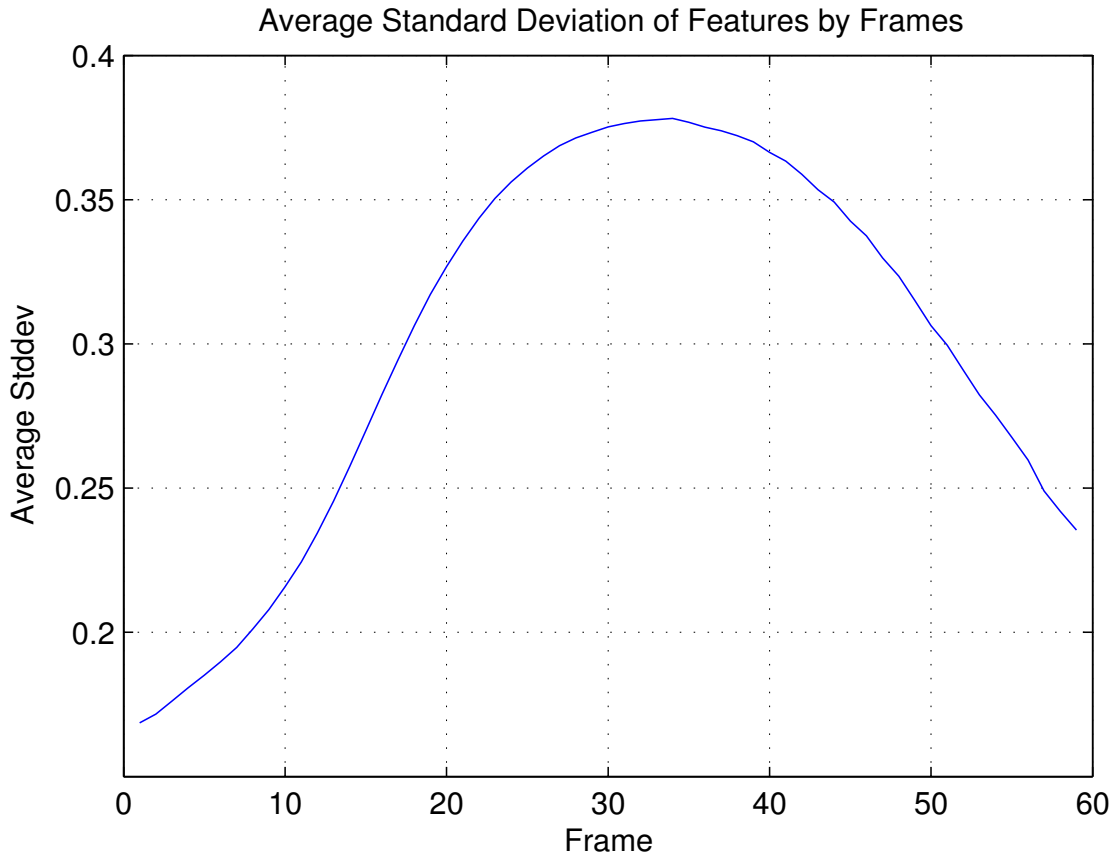


Figure 5.4: The standard deviation aggregated over all features per frame. On average, the most informative frame is 30 frames into the action. Our online classifier can accurately recognize actions an average of 10 frames before this peak.

f_1, \dots, f_{N_A} , where N_A is the number of actions. With these frames, the negative log-loss learning criterion becomes

$$L = -\vec{x} \cdot \theta_T + \log \left(1 + \sum_c \exp(\vec{x}_{f_c} \cdot \theta_c) \right). \quad (5.9)$$

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 Section 5.4.2, 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 Section 5.2.2, 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 [67], can perform similarly to the soft approach used in this paper, but is much more sensitive to the initialization used.

5.5 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 Equation (5.4). The weights, θ , are the identical weights learned for the experiments in Section 5.4. 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.

5.5.1 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 Section 5.4.2 with *maxFrames*. This is combined with the original loss to create the learning criterion for online detection, denoted as $L_{Online}(\cdot)$,

$$L_{Online}(\theta) = L_{Full}(\theta) + \sum_{m \in M} (\gamma \cdot m) L_M(\theta) + R(\theta), \quad (5.10)$$

where $R(\theta)$ is the regularization term from Equation (5.5).

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.

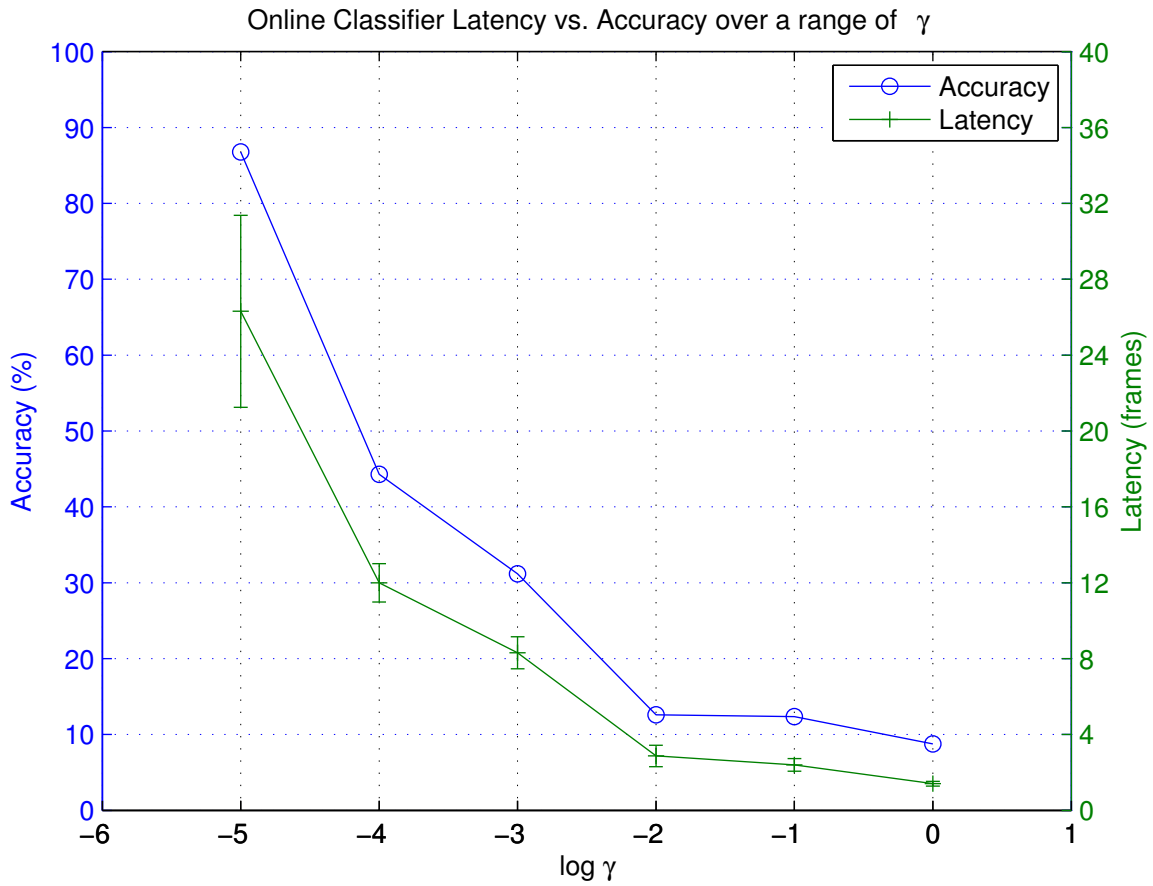


Figure 5.5: Latency compared with accuracy, evaluated on the testing set, for different values of γ . Action accuracy is maximal at the 26th frame on average. As γ increases, latency is gradually reduced at the cost of decreased accuracy.

5.5.2 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 5.5 shows the relationship between the observational latency and system accuracy on the testing set for different values of γ in Equation (5.10).³

Figure 5.5 shows that as γ rises, the accuracy of the online detection system decreases along with the latency. This indicates that the learning criterion in Equation (5.10) 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 Section 5.4.2 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.

The reason for the drop in classification accuracy can be seen in Figure 5.6, 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 5.7 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.

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

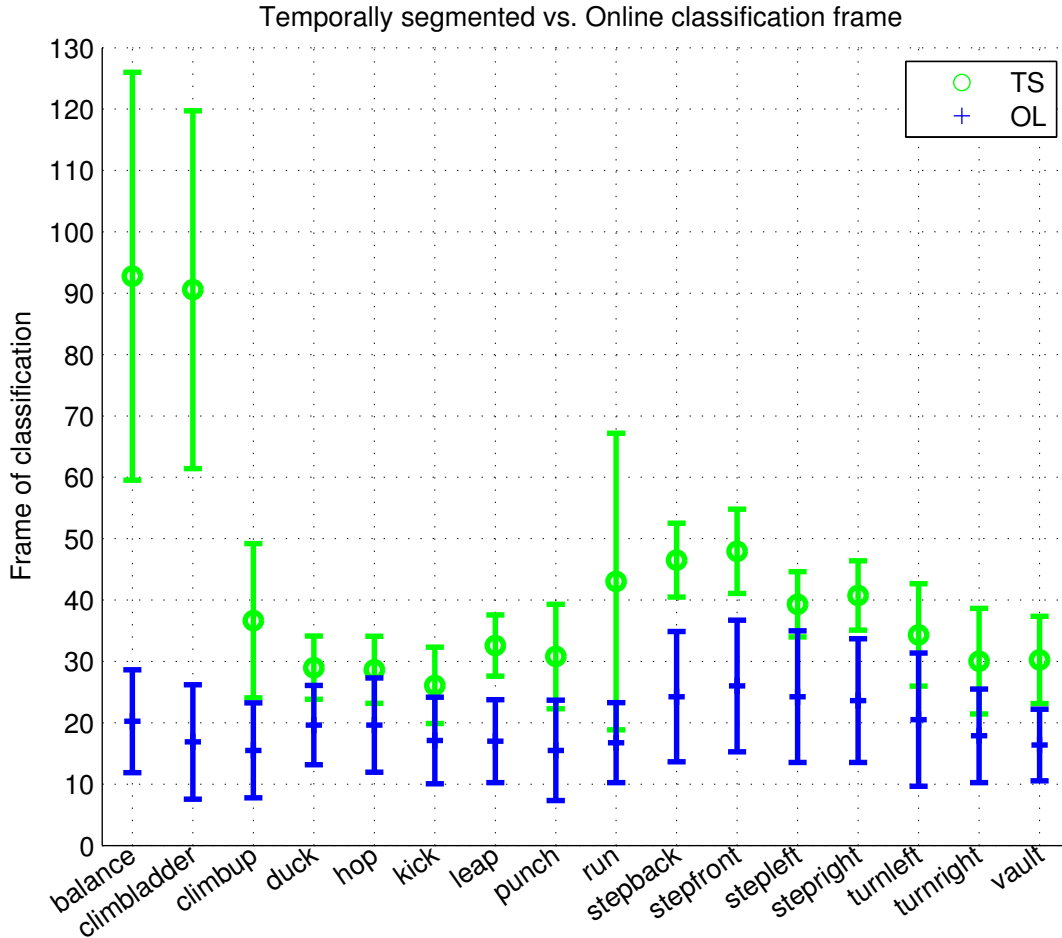


Figure 5.6: Comparison of frame of highest response from full video TS classifier with frame of classification from OL classifier. The value of γ for the results shown is 0. The error bars depict the std. deviation of the frame of classification. Recall that the TS classifier must look at the entire 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.

Online Classifier confusion matrix

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	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	
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	
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	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	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	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	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	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	
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	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	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	
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	
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	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	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	
	balance	climbladder	climbup	duck	hop	kick	leap	punch	run	stepback	stepfront	stepleft	stepright	turnleft	turnright	vault	unassigned

Figure 5.7: 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 Figure 5.3 to 85.78% above), we are able to achieve a significant drop (approx 66%) in classification latency.

Table 5.3: 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 2775, 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%

5.5.3 Reducing Latency

Figure 5.5 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 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.

Table 5.4: GentleBoost vs All Features for online (OL) classification. We see the same trend as observed for the temporally segmented videos in Table 5.3. 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%

5.6 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 [70] technique to greedily select a set of features. This algorithm operates through a stage-wise minimization of the negative log-likelihood of Equation (5.4). 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 5.3 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 5.3) but with greatly improved efficiency. The same trend is observed for online classification for the best possible γ value (refer to Table 5.4).

5.6.1 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 Section 5.3, 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 5.8 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.

5.7 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 Figure 5.5, the action that was most often confused with other actions was removed. The action sets chosen are shown in Table 5.5. 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.

Figure 5.9 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 Section 5.5, we then evaluated the system across variety of thresholds, T in Section 5.5. Figure 5.9 shows the best accuracies achieved for different latency values. All accuracies are averaged in the same four-fold cross-validation framework described in Section 5.4.

As Figure 5.9 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

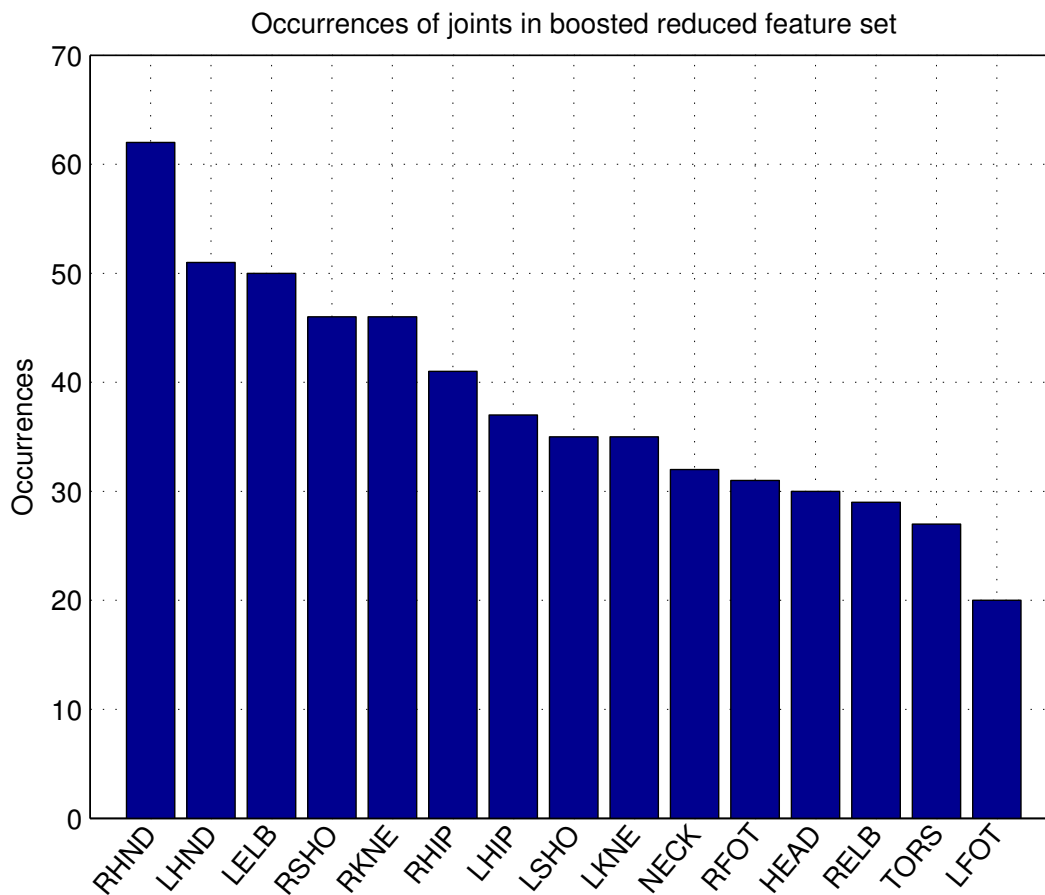


Figure 5.8: List of joints by occurrence in the 300 feature set found by boosting. Notice that the right and left hands are the two most commonly used joints. Less articulate joints, such as the head and torso, as well as less used joints, such as the left and right foot, are used less often.

Table 5.5: 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	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
4 actions	balance hop stepleft stepright	balance climbladder stepleft stepright	balance climbup stepfront stepright	balance leap stepfront stepright	balance leap stepfront stepright	balance leap run stepfront	duck leap run stepfront
8 actions	climbup kick twistright vault	climbup leap twistright vault	duck kick stepback stepleft	duck run stepback stepleft	duck kick stepback stepleft	duck stepback stepleft stepright	balance climbup stepleft stepright
12 actions	duck leap stepback twistleft	hop run stepback twistleft	hop leap twistright vault	climbladder climbup hop kick	climbup hop run vault	climbup hop kick vault	hop kick punch stepback
16 actions	climbladder punch run stepfront	duck kick punch stepfront	climbladder punch run twistleft	punch twistleft twistright vault	climbladder punch twistleft twistright	climbladder punch twistleft twistright	climbladder twistleft twistright vault

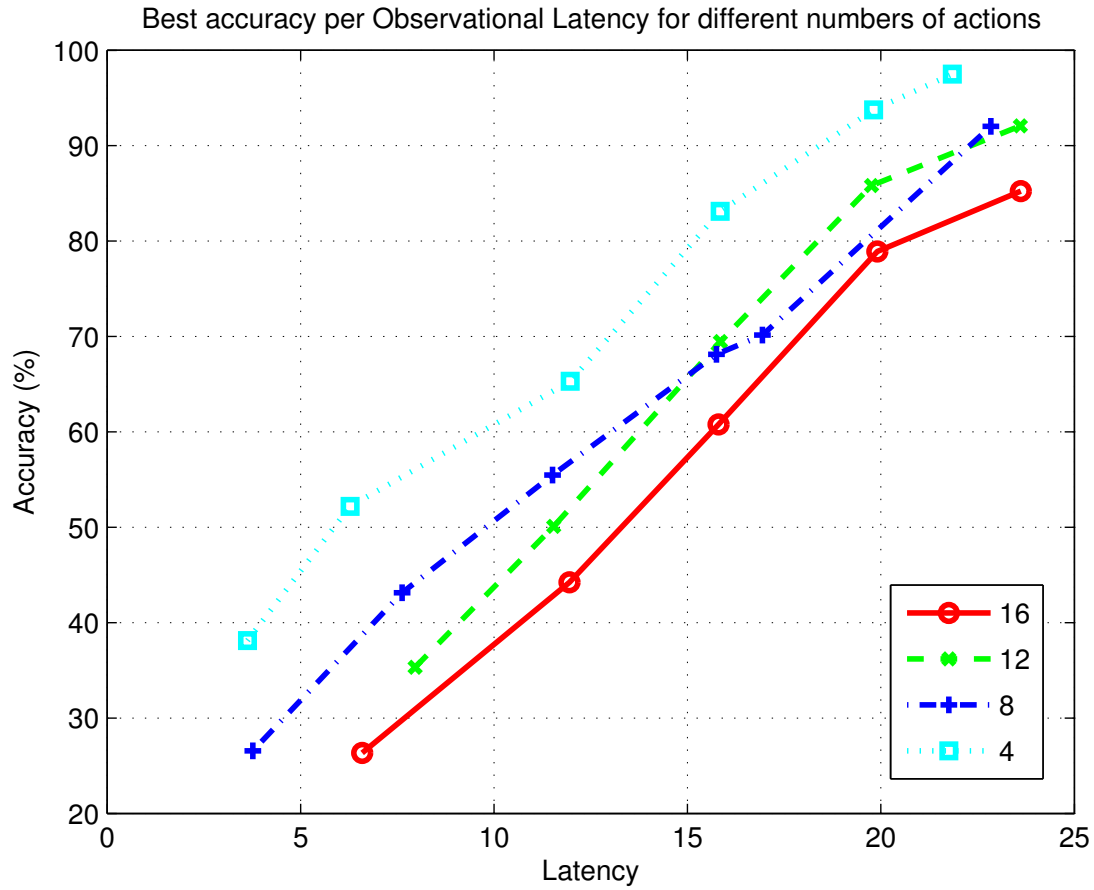


Figure 5.9: These curves show how the accuracy and latency in recognition changes as actions are eliminated. As actions that are difficult to recognize are greedily eliminated, the recognition rate at different latencies rises.

the 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*.

5.8 Accuracy Results on Additional Datasets

We also validated our system in terms of recognition accuracy using two additional datasets — the MSR Action3D dataset from [2] and the MSRC-12 Kinect Gesture dataset [14]. 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.

5.8.1 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 [14]. 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 [14] 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 Section 5.4, 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 Section 5.4, 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 Section 5.4 to balance limitations in both our methodology and the protocol in [14]. While the experiments in [14] 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.

5.8.2 Results of MSR Action3D Dataset

The MSR Action3D dataset from [2] consists of a set of temporally segmented actions, so we followed the experimental methods outlined in [2]. Table 5.6 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.

Table 5.6: This table compares the recognition achieved with our system against previous work on the MSR Action3D dataset from [2]. 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 [73]	42.5%
Dynamic Temporal Warping [74]	54%
Hidden Markov Model [75]	63%
Our Approach	65.7%
Action Graph on Bag of 3D Points [2]	74.7%
Actionlet Ensemble [76]	88.2%

Insight into our system’s performance can be gained by examining recognition accuracy for specific action classes. Table 5.7 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 5.7: This table shows the accuracy of the five least-recognized actions in the MSR Action3D dataset [2] 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%

CHAPTER 6 CONCLUSION

This work argues the importance of localizing actions in video sequences for the task of recognition. Not only accurate localization is important for improving recognition accuracy but, as we have shown, the proper application of localization information is highly essential especially for complex scenes. To support our claim, we introduced a new synthesized, complex dataset which we argue is better suited for analyzing how recognition is affected by background complexity. We show how a change from simple to complex background significantly affects the performance of traditional recognition tools. Using our new synthesized complex dataset, we established that drop in accuracy is directly related to localization and its application in eliminating background information from the recognition pipeline. A detailed analysis of the new dataset is presented, with special emphasis on the impact of factors such as background gradients, background motion and action localization on the recognition results. In light of the analysis, we show how person localization combined with removing cuboid corruption helps tackle the background complexity problem and thus substantially improve the overall recognition results. We show how 'proper' use of localization for interest point pruning and cuboid modification leads to a substantial increase in performance accuracy on both the synthesized and realistic datasets. An automatic localization method is also presented which is shown to outperform the baseline approach. Results are shown with ground-truth masks to show how near-perfect localization helps in improving the recognition accuracy.

The above approach, though useful, is hampered by the fact that we rely heavily on the availability of ground-truth silhouette information. Obtaining them is a manual and highly cumbersome process for large scale datasets. Since the success of the above system is constrained on accurate localization, reliance on human detector [55, 56] and saliency methods [23] as a means of localization is not an ideal solution. Additionally, for most realistic datasets like UCF Sports or UCF11, the background is highly discriminative making it impossible to achieve results better than the state-of-the-art even if we use near-perfect localization. To address the above concerns, we formulated a two-stage system that is designed to:

1. localize the action using a sub-window search learning framework over the entire video.
2. represent an action video sequence using information extracted from the localized region followed by non-linear classification.

Using this proposed model, we were able to show the significance of localization for action recognition. Unlike [1, 24, 25, 26, 27, 28, 30, 6], our system is unique in that it is independent of the requirement of manual ground truth annotations of the actor for the training process. We showed how a system, given the task of selecting a subregion within a video that best represents an action, generally localizes on the actor performing the action. We presented results on two well known complex datasets, namely UCF Sports and UCF11.

Another aspect of the action recognition task that has gained considerable attention recently is determining how quickly an action can be recognized. In other words, what is the minimal observation required by the system to classify an action with reasonably high accuracy. This is a major concern for Human Computer Interaction (HCI) environments where even a small latency is extremely undesirable. Using the probabilistic model proposed above, we modified it so that instead of determining *where* an action is occurring spatially,

we can address the issue of *when* an action happens temporally. For this task, we collected an skeletal action dataset using Microsoft Kinect. The goal of our approach was to learn a discriminative canonical body pose for each action class. 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.

LIST OF REFERENCES

- [1] T. Lan, Y. Wang, and G. Mori, “Discriminative figure-centric models for joint action localization and recognition,” in *ICCV*, 2011.
- [2] 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*, pp. 9–14, 2010.
- [3] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, 2008.
- [4] J. Liu, J. Luo, and M. Shah, “Recognizing realistic actions from videos ”in the wild”,” *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 461–468, 2009.
- [5] M. Bregonzio, S. Gong, and T. Xiang, “Recognizing action as clouds of space-time interest points,” in *CVPR*, 2009.
- [6] N. Ikizler-Cinbis and S. Sclaroff, “Object, scene and actions: Combining multiple features for human action recognition,” in *ECCV*, 2010.
- [7] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *VS-PETS*, pp. 65–72, 2005.
- [8] M. Marszalek, I. Laptev, and C. Schmid, “Actions in context,” *IEEE Conf. Computer Vision and Pattern Recog*, 2009.
- [9] J. Liu and M. Shah, “Learning human actions via information maximization,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, 2008.
- [10] J. Liu, Y. Yang, and M. Shah, “Learning semantic visual vocabularies using diffusion distance,” *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 461–468, 2009.
- [11] A. Kläser, M. Marszalek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *British Machine Vision Conference*, pp. 995–1004, sep 2008.
- [12] metacritic, “Fighters uncaged critic reviews.” <http://www.metacritic.com/game/xbox-360/fighters-uncaged/critic-reviews>, February 2011.
- [13] S. Carlsson and J. Sullivan, “Action recognition by shape matching to key frames,” in *Workshop on Models versus Exemplars in Computer Vision*, 2001.

- [14] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin, “Instructing people for training gestural interactive systems,” in *CHI* (J. A. Konstan, E. H. Chi, and K. Höök, eds.), pp. 1737–1746, ACM, 2012.
- [15] D. Weinland, R. Ronfard, and E. Boyer, “A survey of vision based methods for action representation, segmentation and recognition,” in *CVIU*, 2011.
- [16] R. Poppe, “A survey on vision-based human action recognition,” in *IVC*, 2010.
- [17] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *British Machine Vision Conference*, p. 127, sep 2009.
- [18] M. Marszalek, I. Laptev, and C. Schmid, “Actions in context,” in *CVPR*, 2009.
- [19] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *ICPR*, 2004.
- [20] L. Ballan, M. Bertini, A. D. Bimbo, L. Seidenari, and G. Serra, “Effective codebooks for human action categorization,” in *ICCV workshop on Video-oriented Object and Event Classification (VOEC)*, 2009.
- [21] A. Kovashka and K. Grauman, “Learning a hierarchy of discriminative space-time neighborhood features for human action recognition,” in *CVPR*, 2010.
- [22] P. Felzenswalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multi-scale, deformable parts model,” in *CVPR*, 2008.
- [23] S. Goferman, L. Zelnik-Manor, and A. Tal, “Context-aware saliency detection.,” in *CVPR*, pp. 2376–2383, IEEE, 2010.
- [24] A. Klaser, “Learning human actions in videos,” in *PhD thesis, Universit de Grenoble*, 2010.
- [25] S. Z. Masood, A. Nagaraja, N. Khan, J. Zhu, and M. F. Tappen, “Correcting cuboid corruption for action recognition in complex environment,” in *ICCV workshop on Video Event Categorization, Tagging and Retrieval for real-world applications (VECTaR)*, 2011.
- [26] A. Yao, J. Gall, and L. V. Gool, “A hough transform-based voting framework for action recognition,” in *CVPR*, 2010.
- [27] M. R. Amer and S. Todorovic, “A chains model for localizing participants of group activities in videos,” in *ICCV*, 2011.
- [28] J. Yuan, Z. Liu, and Y. Wu, “Discriminative subvolume search for efficient action detection,” in *CVPR*, 2009.
- [29] H. Boyraz, M. F. Tappen, and R. Sukthankar, “Localizing actions through sequential 2d video projections,” in *Fourth IEEE Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB)*, 2011.
- [30] W.-L. Lu, K. Okuma, and J. J. Little, “Tracking and recognizing actions of multiple hockey players using the boosted particle filter,” in *IVC*, 2009.

- [31] S. Ali and M. Shah, “Human action recognition in videos using kinematic features and multiple instance learning,” vol. 32, pp. 288–303, 2010.
- [32] Y. Shen and H. Foroosh, “View-invariant action recognition from point triplets,” vol. 31, pp. 1898–1905, 2009.
- [33] W. Yang, Y. Wang, and G. Mori, “Recognizing human actions from still images with latent poses,” pp. 2030–2037, 2010.
- [34] Z. L. Liangliang Cao and T. S. Huang, “Cross-dataset action detection,” *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2010.
- [35] N. Cuntoor and R. Chellappa, “Key frame-based activity representation using antieigenvalues,” in *ACCV*, 2006.
- [36] L. Shao and L. Ji, “Motion histogram analysis based key frame extraction for human action/activity representation,” in *CRV*, 2009.
- [37] Z. Zhao and A. Elgammal, “Information theoretic key frame selection for action recognition,” in *Proceedings of the British Machine Vision Conference*, pp. 109.1–109.10, BMVA Press, 2008.
- [38] S. Masood, A. Nagaraja, N. Khan, J. Zhu, and M. Tappen, “Correcting cuboid corruption for action recognition in complex environment,” in *The 3rd IEEE Workshop on Video Event Categorization, Tagging and Retrieval for Real-World Applications (VEC-TaR2011), ICCV Workshops*, 2011.
- [39] A. Vahdat, B. Gao, M. Ranjbar, and G. Mori, “A discriminative key pose sequence model for recognizing human interactions,” in *Eleventh IEEE International Workshop on Visual Surveillance*, pp. 1729–1736, 2011.
- [40] C. T. S. Cheema, A. Eweiwi and C. Bauckhage, “Action recognition by learning discriminative key poses,” in *ICCV Workshops*, 2011.
- [41] F. Lv and R. Nevatia, “Single view human action recognition using key pose matching and viterbi path searching,” 2007.
- [42] M. Narasimhan, P. A. Viola, and M. Shilman, “Online decoding of markov models under latency constraints.,” pp. 657–664, 2006.
- [43] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, “A unified framework for gesture recognition and spatiotemporal gesture segmentation,” vol. 31, no. 9, pp. 1685–1699, 2009.
- [44] J. W. Davis and A. Tyagi, “Minimal-latency human action recognition using reliable-inference,” *Image Vision Computing*, vol. 24, no. 5, pp. 455–472, 2006.
- [45] D. Ramanan, D. A. Forsyth, and A. Zisserman, “Strike a pose: Tracking people by finding stylized poses,” pp. 271–278, 2005.
- [46] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, “Efficient regression of general-activity human poses from depth images,” 2011.

- [47] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from a single depth image,” 2011.
- [48] M. Raptis, D. Kirovski, and H. Hoppe, “Real-time classification of dance gestures from skeleton animation,” in *Symposium on Computer Animation*, pp. 147–156, 2011.
- [49] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” in *The Tenth IEEE International Conference on Computer Vision (ICCV’05)*, pp. 1395–1402, 2005.
- [50] A. Gilbert, J. Illingworth, and R. Bowden, “Fast realistic multi-action recognition using mined dense spatio-temporal features,” in *IEEE 12th International Conference on Computer Vision (ICCV)*, 2009.
- [51] A. Levin, D. Lischinski, and Y. Weiss, “A closed-form solution to natural image matting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 228–242, 2008.
- [52] K. Schindler and L. van Gool, “Action snippets: How many frames does human action recognition require?,” *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 1–8, 2008.
- [53] P. V. Gehler and S. Nowozin, “Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 06 2009.
- [54] Z. Jiang, Z. Lin, and L. S. Davis, “A tree-based approach to integrated action localization, recognition and segmentation,” in *Third Workshop on Human Motion (In Conjunction with ECCV)*, 2010.
- [55] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, 2010.
- [56] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, “Discriminatively trained deformable part models, release 4.” <http://people.cs.uchicago.edu/~pff/latent-release4/>.
- [57] I. Laptev, “On space-time interest points,” in *IJCV*, 2005.
- [58] N. Schraudolph, “Local gain adaptation in stochastic gradient descent,” in *International conference on Artificial Neural Networks*, pp. 569–574, 1999.
- [59] C. Lampert, M. Blaschco, and T. Hofmann, “Beyond sliding windows: Object localization by efficient subwindow search,” 2008.
- [60] M. Rodriguez, J. Ahmed, and M. Shah, “Action mach: A spatial-temporal maximum average correlation height filter for action recognition,” in *CVPR*, 2008.
- [61] L. Yeffet and L. Wolf, “Local trinary patterns for human action recognition,” in *ICCV*, 2009.

- [62] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, “Action recognition by dense trajectories,” in *CVPR*, 2011.
- [63] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, “A large-scale benchmark dataset for event recognition in surveillance video,” in *CVPR*, 2011.
- [64] P. Guan, A. Weiss, A. O. Blan, and M. J. Black, “Estimating human shape and pose from a single image,” pp. 1381–1388, 2009.
- [65] 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*, pp. 155–162, 2010.
- [66] L. Sigal, A. Balan, and M. J. Black, “HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion,” vol. 87, no. 1-2, 2010.
- [67] P. F. Felzenszwalb, R. B. Girshick, and D. Mcallester, “Cascade object detection with deformable part models,” pp. 2241–2248, 2010.
- [68] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” p. 511, 2001.
- [69] P. Viola and M. Jones, “Robust real-time object detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.
- [70] J. Friedman, T. Hastie, and R. Tibshirani, “Additive Logistic Regression: a Statistical View of Boosting,” *The Annals of Statistics*, vol. 38, no. 2, 2000.
- [71] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” 2001.
- [72] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” vol. 23, no. 11, pp. 1222–1239, 2001.
- [73] J. Martens and I. Sutskever, “Learning recurrent neural networks with hessian-free optimization,” 2011.
- [74] M. Müller and T. Röder, “Motion templates for automatic classification and retrieval of motion capture data,” in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, (Aire-la-Ville, Switzerland, Switzerland), pp. 137–146, Eurographics Association, 2006.
- [75] F. Lv and R. Nevatia, “Recognition and segmentation of 3-d human action using hmm and multi-class adaboost,” 2006.
- [76] J. Wang, Z. Liu, Y. Wu, and J. Yuan, “Mining actionlet ensemble for action recognition with depth cameras,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.