Recognizing Simple Human Actions by Exploiting Regularities in Pose Sequences

by

Nikola Otasevic

S.B. Computer Science and Engineering, Massachusetts Institute of Technology(2012)
S.B. Economics, Massachusetts Institute of Technology(2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by .....
Professor Patrick H. Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by .....
Professor Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Recognizing Simple Human Actions by Exploiting Regularities in Pose Sequences

by

Nikola Otasevic

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

The human visual system represents a very complex and important part of brain activity, occupying a very significant portion of the cortex resources. It enables us to see colors, detect motion, perceive dimensions and distance. It enables us to solve a very wide range of problems such as image segmentation, object tracking, as well as object and activity recognition. We perform these tasks so easily that we are not even aware of their enormous complexity. How do we do that? This question has motivated decades of research in the field of computer vision.

In this thesis, I make a contribution toward solving the particular problem of vision-based human-action recognition by exploiting the compositional nature of simple actions such as running, walking or bending.

Noting that simple actions consist of a series of atomic movements and can be represented as a structured sequence of poses, I designed and implemented a system that learns a model of actions based on human-pose classification from a single frame and from a model of transitions between poses through time.

The system comprises three parts. The first part is the pose classifier that is capable of inferring a pose from a single frame. Its role is to take as input an image and give its best estimate of the pose in that image. The second part is a hidden Markov model of the transitions between poses. I exploit structural constraints in human motion to build a model that corrects some of the errors made by the independent single-frame pose classifier. Finally, in the third part, the corrected sequence of poses is used to recognize action based on the frequency of pose patterns, the transitions between the poses and hidden Markov models of individual actions.

I demonstrate and test my system on the public KTH dataset, which contains examples of running, walking, jogging, boxing, handclapping, and handwaving, as well as on a new dataset, which contains examples of not only running and walking, but also jumping, crouching, crawling, kicking a ball, passing a basketball, and shooting a basketball. On these datasets, my system exhibits 91% action recognition recall rate.

Thesis Supervisor: Professor Patrick H. Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

Acknowledgments

It was March 15^{th}, 2008. I was sitting in my room, nervously refreshing my browser. When I finally got the letter open, I shouted with excitement when I found out that I was accepted. I could not be more excited about attending the school that had been my dream ever since I first heard about it. The best engineering school in the world. I thought that I would get to listen to Nobel laureates and Turing award winners, work with the most advanced equipment, learn from the best curriculum and have the best facilities in the world.

Five years later, I have listened to Nobel laureates and Turing award winners, I have worked with the most advanced equipment, I have learned from the best curriculum and have had the best facilities. But five years later, I realize that all these things are not what made my education at MIT so great.

Education is a set of experiences and stories. The experiences that push you farther than you ever thought you could be pushed. And stories which inspire you to think deeper than you ever thought you could. Throughout my education, I have been lucky to always be surrounded by a group of people who bring together an enormous range of experiences and stories. And that is what MIT was about.

I worked and lived with some of the most amazing people I have ever met. It has never been about the Nobel laureates and Turing award winners, the most advanced equipment, the best curriculum or the facilities. My entire education has always been about the people. I owe tremendous gratitude to those without whom my experience would not have been nearly as rich:

To Professor Winston, the person who truly shaped my experience at MIT. I will be forever thankful for all the advice I have received during this research project and for all the freedom you gave me to explore the area of computer vision and artificial intelligence. But most of all, for letting me have the experience of being your student.

To my family, for all your care and support. For your constant emphasis on education throughout my life.

To Jovana, for always standing beside me, unconditionally.
To Ameya, Keshav and Latif, for all the great discussions and laughs. For all the amazing work we've done together. For friendship.
To all my friends in Genesis group, Sam, Hiba, Giuliano, Igor, Adam, Matthew, Sila, Mark, Dylan for great discussions that improved this work. But most of all, for all our discussion about other random things that made my experience so great.

To all my friends who helped me record the data I needed for this research.
To Ameya and Sam for suggesting many improvements on this thesis. Without you, this thesis would have been a mess.

This research was supported, in part, by the Defense Advanced Research Projects Agency, Award Number Mind's Eye: W911NF-10-2-0065.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"In general we are least aware of what our minds do best." -Marvin Minsky

The human visual system represents a very complex and important part of brain activity, occupying about 30% of the cortex resources. It enables us to see colors, detect motion, perceive dimensions and distance. It enables us to solve a very wide range of problems such as image segmentation, object tracking, as well as object and activity recognition. We perform these tasks so easily that we are not even aware of their enormous complexity.

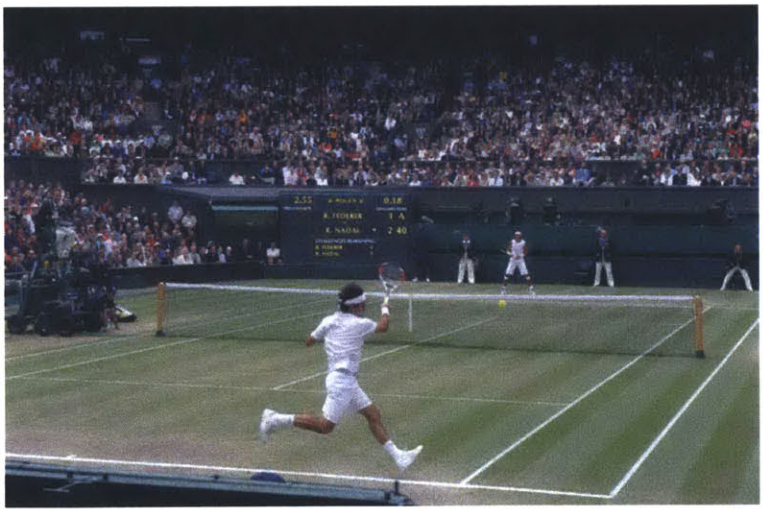

Figure 1-1: Wimbledon 2008 Final between Roger Federer and Rafael Nadal

The human vision system easily distinguishes between different moves in sports. The

single image in figure 1-1 already carries a lot of information. If we look at the video with a few frames, we say: "Federer served to Nadal's backhand, Nadal sliced the ball short, Federer cut off the shot and played a drop valley." This demonstrates the ability to recognize a wide range of very visually similar actions without too much effort. How do we do that? How can we build a system that can provide a play-by-play commentary of a sports game? How can we build a system that can detect security threats in real-time or solve crimes using the surveillance video data in a few minutes? These questions have motivated decades of computer vision research. In this thesis, I explore the problem of vision-based human action recognition by focusing on the recognition of low complexity actions such as running, walking and jumping. I implemented a system that can recognize human actions on a public dataset and a new dataset constructed primarily for this thesis. The system I built achieves an average of 91% recall rate on the public KTH dataset containing 6 actions(running, walking, jogging, handwaving, handclapping, boxing) and an average of 91% recall rate on the SHA dataset containing 8 actions(running, walking, jumping, crouching, crawling, kicking a ball, passing a basketball, shooting a basketball).

1.1 Problem Statement

The general task of recognizing human activity is very broad. For that reason the field has been conceptually divided and depending on the complexity of the activity we typically talk about action primitives, actions, activities, interactions and group activities [Aggarwal and Ryoo, 2011]. An action primitive is typically defined as an atomic movement such as "raising a leg" or "stretching an arm". Actions are made of action primitives and typically involve the movement of more parts of the body. Examples include running, walking, jumping etc. Activities could contain a few actions executed in a particular order, while interactions and group activities include two or more people. The activities and interactions very often entail a complex visual scenario including additional objects. Each of these categories requires a different approach to the problem. For example, in order to recognize an interaction such as

"two people fighting", the high level process would have to get an information about what is happening on the lower level such as action "punching" or even lower than that such as "extending an arm".

Problem Formulation

In this thesis, I focus on the level of action primitives and actions. Even at this level, there exist several categories of approaches, some of which I will discuss in chapter 2. I decided to formulate this problem as a so called single-layered approach. Single-layered simply means that a video input is considered one frame at a time. The motivation for this comes from the fact that the human vision system can very often infer an action from a single image or a very short sequence of frames.

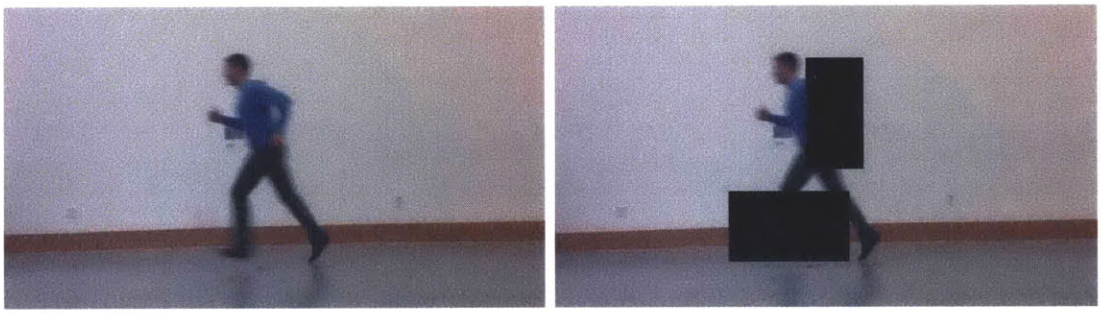

Figure 1-2: One frame example of a person running

Almost everyone understands that the person from the figure above is running even though they can see only one frame of the entire action. I hypothesize that this is because of an ability to infer an appearance of the same person in a set of frames before and after the given one by exploiting the structural constraints in the space of transitions between different poses. That is, even though this one frame could potentially be inserted into a set of frames that typically represent some other action (ex. jumping), we are able to estimate that this scenario is highly unlikely and therefore we label this as running despite the missing frames.

Furthermore, note that even under a partial occlusion one can infer enough information about the pose from the image. This suggests that the vision system has an ability to infer, with a very high certainty, a particular pose of the human body based

only on the fragments of the image.

I have mentioned the term pose a few times already, so I think that it would be good to introduce it in a more formal way.

Definition 1. *I define a* ***pose*** *to be a group of single-frame appearances of a human body where variations in viewpoint and visual appearance between the group members are minimal.*

Figure 1-3: Example of a group of single-frame appearances that can be described with one pose

This definition allows us to capture a representation of a human body in a particular configuration of its parts. Because of the fact that some variations within the same group are allowed, the pose space is discretized.

Definition 2. *I define an* ***action*** *to be a single-person activity that can be described by a small collection of poses arranged in a repetitive sequence.*

An action can contain multiple poses in its sequence and a pose can appear in multiple actions. Therefore, there is no one-to-one mapping from poses to actions or the other way around. However, in this work I exploit the fact that most actions are compositional and exhibit structured transitions between poses. This allows actions to be recognized based on the sequence structure they exhibit.

1.2 Framework

I started my research with an intent to recognize simple human actions from poses detected in single frames. This single-frame goal proved to be illusive, leading me to

use single-frame pose classification as an element in a larger system rather than as a sole determiner of action.

In this section I outline the framework for building a system that has an ability to infer actions using single frames and the knowledge about the pose transition space. I wish to explore how the appearance and the transition space associated with an action, play a joint role in the ability to correctly classify, recognize and imagine the action being performed.

For that purpose I developed a framework that comprises three stages. (1) First, I needed a system that is capable of inferring a human pose from a single frame. The system was developed by employing Ullman's intermediate features principle [Ullman et al., 2002] and developing hierarchical group classification tree. (2) Once the single-frame pose estimation was implemented, I could look at a sequence of the pose estimates. In particular, the transitions between these estimates are naturally constrained. I used that fact to develop a system modeled with a hidden Markov model that corrects some of the errors that the single-frame classifier makes. (3) In the third stage, after the best possible sequence is estimated, I developed a system that can recognize the sequence as an action.

Pose Classifier

The Pose Classifier is a system responsible for the classification of single frame images into one of the possible pose classes. I approach the training of the pose classifier in two stages. First, I develop a binary classifier for the case of two poses. Then, I combine the binary classifiers from all the poses into a hierarchical tree of groups of similar poses.

In developing binary classifier the goal is to understand what kind of features are present in one pose class and not in the other. For example, in figure 1-4, some features that are present in the pose on the left, and not in the one on the right, are shown.

For this purpose, a set of examples representing a pose class is fragmentized such that fragments at random locations on the image are extracted. The next step is to

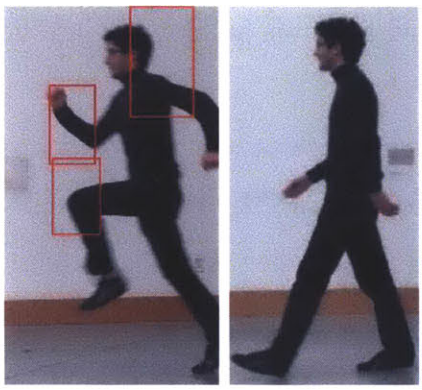

Figure 1-4: Example of a group of discriminative features

recognize which of those fragments are indicative of the pose class. These fragments are successively picked through a boosting method forming a group of fragments that best describe the pose class. After the binary classifiers for all the pairs are created, the poses that are closest to each other are successively joined into new pose groups. At each iteration, the two closest groups are joined forming a hierarchical group tree. This step reduces the complexity of the classifier and speeds up the classification process at run time.

The pose classifier is then represented through a decision tree structure. Given an image, the classifier looks for different fragments in their respective regions of interest, goes down the tree and eventually determines the pose class depending on which fragments it locates in the image.

Modelling Time-sequential Poses

Given a set of time-sequential frames, based on the previously described method, every frame can be evaluated for the most likely pose. That evaluation is done independently for each frame. However, the time-sequential nature of these frames can be very useful. For example, imagine that the sequence of poses as in figure 1-5 has been estimated.

This kind of sequence of estimates is possible simply because the previously described pose classifier has no knowledge of any constraints between the frames. So can we somehow understand that the pose classifier made a mistake given all the estimates?

Figure 1-5: Example of an unlikely sequence in which the transition model can help correct some mistakes made by a single-frame classifer

We could do that if we understood something about the structure of the transition space between the poses and how likely any given error is. For example, look at the poses in the middle of figures 1-6 and 1-5. For a moment, let me refer to them as pose "run" and pose "bend". If we know that pose classifier fairly often makes an error and confuses "run" for "bend", and if we know that transition from and to the poses surrounding the middle ones are fairly unlikely for the pose "bend", we could say with relatively high certainty that the middle pose estimate was an error and that it should really be estimated as "run" instead of "bend". I model the pose transitions using a hidden Markov model where states are poses.

Figure 1-6: Example of a likely sequence given the single-frame pose estimates as in figure 1-5

The structural constraints of human motion represent a powerful way to improve the single-frame pose classification. In the results chapter, I show that modelling structural constraints can improve the results of the pose classification by an average of 15%.

Sequence Recognition

Once the sequence of poses is estimated and corrected in the first two stages, the system needs to make a final decision on the action label. At this point in the process, the problem simplifies to the problem of discovering patterns in the text. That is, suppose that every pose is equivalent to a letter. Then every sequence is a word made of letters from the pose alphabet. The meaning of the word depends on the frequency of the letters, the way they transition from one to another and other letter patterns.

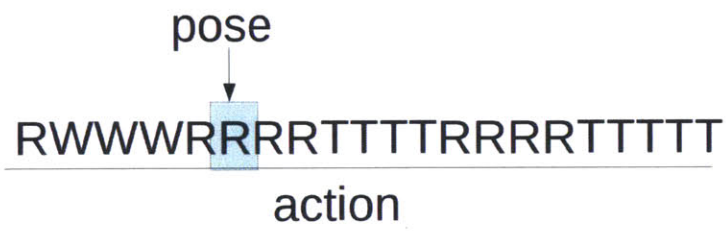

Figure 1-7: Example of a pose sequence seen as a letter sequence

The role of the action detection system is to classify these text patterns as one of the actions. I approach this problem in two ways:

1. Unsupervised hierarchical agglomerative clustering

2. Supervised training of HMM models, one per action class.

In the hierarchical clustering, at each stage, the algorithm joins the two closest clusters based on a certain predefined distance metric. For this purpose two distance metrics are used, the Levenshtein distance and the bag of features vector distance, which will be thoroughly examined in chapter 5. In the supervised method, I train a model for each action class, optimizing the parameters of the model based on the training sequences. Then I use these models to recognize the new, unseen, sequences.

The rest of this thesis is organized as follows: In chapter 2, I introduce the most influential and recent work in the subfields of computer vision related to action recognition. In chapter 3, I describe in detail the classification of poses using independent single-frame images. In chapter 4, I introduce hidden Markov model and explain how it is used to model the transitions between poses and improve the single-frame pose classification. In chapter 5, I describe the sequence recognition process. In chapter 6, I describe the design and implementation of the entire system. In chapter 7, I present and discuss my results. Finally, in chapter 8, I present my contributions and outline some of the steps for future work.

Chapter 2

Previous Work

Action recognition is an important area of computer vision research. Its applications include surveillance systems, self-driving vehicles, video search, sports analytics, medical monitoring systems etc. All of these applications require some level of understanding of ongoing activities which motivated the research in the area of action recognition. In this chapter I describe some of the most influential work in the domain of action recognition and other related domains of computer vision. It is not my intention to offer a comprehensive review of the action recognition field, but to summarize the most important recent results and approaches that influenced my work in this thesis. Depending on the part of the problem which a particular approach is attempting to solve as well as the structure of the solution, action recognition work can be categorized as in diagram 2-1, adapted from [Aggarwal and Ryoo, 2011].

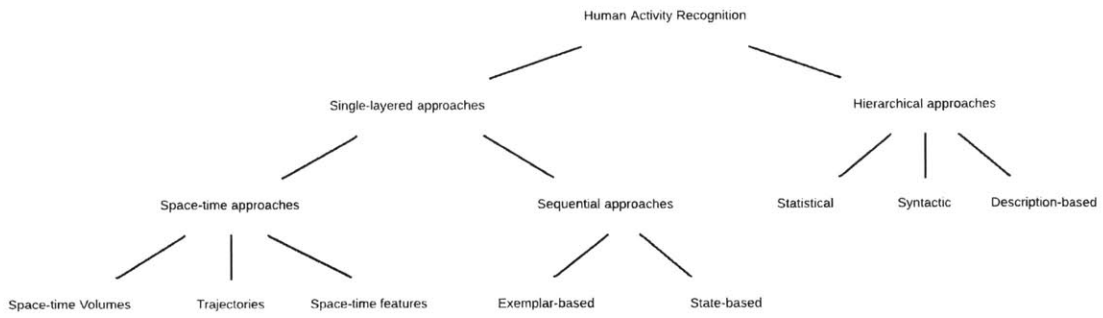

Figure 2-1: The hierarchical approach-based taxonomy adapted from [Aggarwal and Ryoo, 2011]

Activities can be categorized based on the complexity of the scene (one person or a crowd) as well as based on the complexity of the individuals' motion. For instance, [Aggarwal and Ryoo, 2011] divide activities into gestures (action primitives), actions, interactions and group activities. They define gestures to be "elementary movements of a person's body part such as stretching an arm and raising a leg". Actions are defined as "single-person activities that may be composed of multiple gestures." Examples include actions such as walking, running and punching. Interactions are "human activities that involve two or more persons and/or objects" For example, two people playing throw-catch is an interaction. Finally, group activity is defined as "activities performed by conceptual groups composed of multiple persons and/or objects such as marching, a group meeting or two groups fighting".

In my research, I focused on the actions performed by one person such as running, walking and jumping. Because of that I will focus on reviewing the literature approaches attempting to solve this particular problem, but I will also briefly describe what other approaches entail.

Hierarchical approaches are applied to higher complexity activities that typically exhibit a high level of variation in the visual domain such as following, chasing, fighting. These types of problems require an approach which builds a high-level model of how simpler actions make an activity. For example, activity such as "following" may contain multiple people or objects and may involve simpler activities such as running, walking or stopping. Hierarchical activity recognition systems make a representation of an activity that could potentially be performed in many visually completely different ways. A high level system typically relies on the low level action recognition systems to recognize simple actions which can then be used as building blocks.

Single-layered approaches consider a video as a sequence of images and an action as a particular configuration of the images in the sequence. This category of approaches is very successful when a particular sequential pattern that describes an action can be learned. [Aggarwal and Ryoo, 2011] divide this category into space-time and sequential approaches. Space-time approaches model a video in a 3D space by adding the

time dimension to 2D frames and are further divided based on the kind of features that are extracted from the video sequence. Sequential approaches analyze a video as a sequence of frames and the features that appear in individual frames. They are further divided based on the image representation that they use.

Action recognition field relies on good image features selection techniques and ways to classify these images and video representations. In the rest of this chapter, I first discuss different most popular and influential feature selection techniques and then, I review how they are used to classify images and video sequences.

2.1 Feature Selection and Image Representations

In this section, I discuss the work that relates to features that are typically extracted from image sequences. The basic requirement of any image representation is to be able to generalize over small changes in appearance, background and viewpoint. However, the representation needs to be rich enough to be able to robustly classify differences between classes of appearances. Poppe in his review [Poppe, 2010] divides image representations into global and local ones. With global representations, a region of interest is typically encoded as a whole through an image descriptor. In local representations, a region of interest is typically treated as a collection of independent patches that form a model of the appearance.

Global Image Representations

Global image representations typically encode a region as a whole resulting in a descriptor or a template. In early works on global image representations, people focused on silhouette extraction and motion information extraction typically encoding it with optical flow.[Bobick and Davis, 2001] form a binary motion energy image between any two successive frames in a sequence and compare them using Hu moments. Silhouettes can also be matched directly using a distance metric such as Euclidean or Chamfer distance [Weinland and Boyer, 2008].

The drawback of global image representations approaches is that they are usually sen-

sitive to changes in viewpoint or partial occlusions. In order to overcome these types of issues, people have over time developed so called grid-based global representations. Here, an image is divided into spatial regions that represent local observations.
The work that has highly influenced the research of feature selection in recent years is the development of histograms of oriented gradients (HOG) [Dalal and Triggs, 2005]. Dalal and Triggs made a significant improvement in the results of human detection by developing HOG descriptors in which they use fine-scale gradients, orientation binning, and local normalization.

Figure 2-2: An overview of the feature extraction and people detection process taken from [Dalal and Triggs, 2005]

Local Image Representations

Local image representations encode a region as a set of local patches or descriptors. These representations are typically less susceptible to occlusion or partial changes in viewpoint than the global representations.
The work that has had a great impact on my research in the domain of feature selection is the Ullman et al. work on the intermediate features principle. They introduce the intermediate features principle in [Ullman et al., 2002] where they show that features of intermediate complexity and scale are the most optimal for the visual classification. They show that intermediate features naturally arise as the most informative ones in the process of face detection. They also show that normalized cross-correlation performs as well as more complex features such as SIFT [Lowe, 1999]. In later publications ([Epshtein and Uliman, 2005] and [Epshtein and Ullman, 2007]), based on the principle of intermediate features, they develop a hierarchical representation of objects, which proves to be very effective for localization of object parts and subparts.

Most recently, the work of [Felzenszwalb et al., 2010] has had an important impact on the field of pose estimation. Their work on pose estimation serves as the starting point for many action recognition approaches. They develop a representation of an object using mixtures of deformable part models. This work has been widely adopted by the research community both because of its state of the art performance and easily accessible implementation.

2.2 Action Classification

In this section, I describe how image representations are used in the research on action recognition. I divided the section into two parts. First I describe the most influential work in the category of space-time approaches and then I introduce the category of sequential approaches.

Space-time approaches

In space-time approaches, a 3-D representation is used to represent actions. The volume is constructed by joining consecutive frames and extracting features in XYT space. Space-time approaches can be divided into three categories based on the kind of features that they extract.

1. In the space-time volume approaches the consecutive frames are stacked together and treated as a silhouette in the 3D space.

2. In the trajectories approaches, the trajectories of different body parts or other points of interest are used to represent actions.

3. In the space-time features approaches, the 3D features are extracted instead of simple volumes.

[Klaser and Marszalek, 2008] extend the HOG representation to 3D to form a descriptor based on the 3D spatio-temporal gradients. They achieve 91.4% accuracy on the KTH dataset.

[Wang et al., 2011] used dense trajectories representation to represent interest points in each frame and tracked them based on the information from a dense optical flow field. They achieve 94.2% accuracy rate on the KTH dataset. They also develop a novel descriptor which is robust to camera motion and can be used in more realistic environments.

[Bregonzio et al., 2009] represent actions as clouds of space-time interest points. They develop a technique for extracting interest points by avoiding the extraction in background and highly textured foreground which allows them to focus on the points that are the most important in describing the motion.

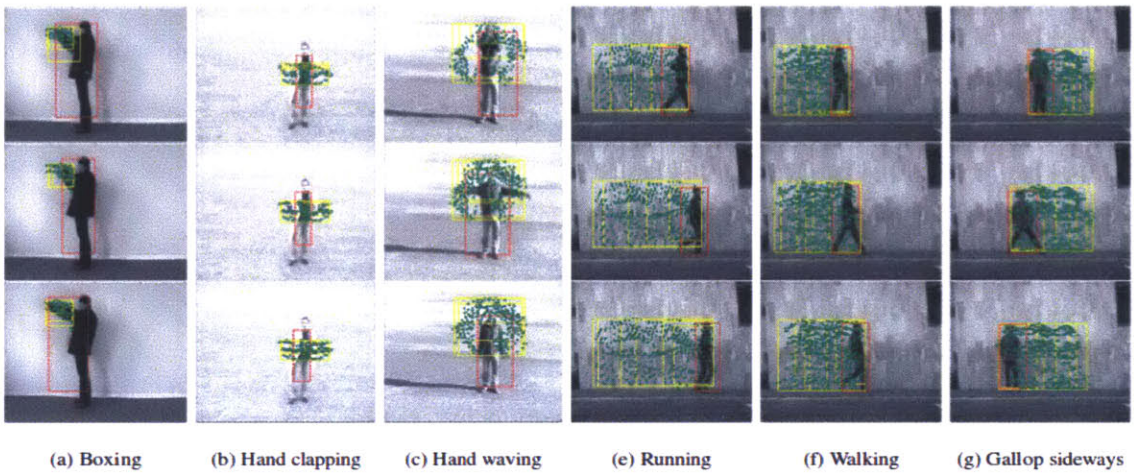

Figure 2-3: Examples of clouds of space-time interest points, taken from [Bregonzio et al., 2009]

Sequential approaches

In this section, I summarize the research that uses the sequential approaches, the analysis of sequence of single-frame observations.

[Liu et al., 2012] present an approach that is based on key-frame selection and pyramidal motion feature representation. They use this representation to select the most informative frames from a large pool of action sequences. They test the approach on the KTH dataset and obtain 95.5% accuracy.

The time-sequential approach that influenced my research directly is that of Yamato

et al. in [Yamato et al., 1992] where they use hidden Markov models for learning action representations from a set of time-sequential images. At each frame a vector is formed from a set of mesh features extracted from the image. Then, HMM is used to train one model per action category given the training sequences of vector observations.

[Lv and Nevatia, 2007] propose a framework called *Action Net* in which they perform single-frame silhouette matching using Pyramid Match Kernel algorithm and model transition between different actions.

Chapter 3

Pose Classification

The human body can find itself in different poses during its motion. At every frame, I attempt to classify a particular appearance of a human body into a pose category. In this chapter I discuss the problem of independent, single-frame pose classification which means that the pose classifier considers one frame at a time and has no knowledge about the relation between this single-frame and other frames that it sees before and after. The temporal relationship between the frames will be discussed in chapters 4 and 5.

3.1 Problem Formulation

Given a limited number of poses to choose from and an image, a pose classifier is required to output a pose that best describes the pose of the human in the image. The pose classifier is developed through training during which it learns about different poses in human motion. For that purpose, a set of examples is provided for each category of poses. The role of training is then to form a model of poses by picking features that can be used to distinguish different poses.

This is done in two stages. First, the classifier learns how to distinguish between any two poses. Second, it learns how to make a decision about the pose category based on the pairwise decisions made in the first stage.

I start by explaining the training in the simple case with only two poses where the

role of a classifier is to make a binary decision.

3.2 Case with Two Poses

Suppose for a moment that there are only two poses that we want to distinguish between, as shown in figure 3-1. The first one is very often found as a constituent element in walking while the second one is usually a constituent element in running. In the rest of this section, I will refer to them as pose W and pose R.

Figure 3-1: Example of two poses

The first goal is to learn the kind of features that are present in pose R and not in pose W.

1. First, I take examples of pose R and extract image fragments at many locations in the examples. An image fragment is a cropped piece of an image.

2. Then, I develop a mechanism to decide whether a particular fragment is present in an image.

3. Finally, I search for all the fragments in the positive and negative examples. Using a boosting algorithm, I select the fragments that are most indicative of pose R.

Fragment Extraction

For the purpose of fragment extraction, I adopted a framework of intermediate features first described by [Ullman et al., 2002] The idea behind intermediate features is that the features of intermediate scale and resolution are the most descriptive ones. Ullman et al. in their original paper describe the technique for the purpose of face detection, and find that features of small size (such as "one eye") are typically found in many places in an image, while features of large size (entire faces) are found rarely. Similarly, in the case of human poses, small size features (such as "top of the elbow") are found in many places in the image while entire body poses are rarely matched well. This result helps inform the process of feature selection. Image fragments are extracted from the given examples of pose R. This fragment extraction is done at multiple random locations across the image as well as at multiple scales. I used a simple sliding window technique with the step of 1/3 of the fragment size. The size of the fragments ranged between 0.1 and 0.5 of the size of the example. Once the fragments have been extracted, I compute the match between each fragment and all the positive (pose R) and negative (pose W) examples.

Measure of Fragment Match

Each of the extracted fragments is first matched against all the positive and negative examples. That means that the fragment is slid across a certain region of the image and the fragment match is computed. The maximum value of the match is then denoted as the match between the fragment and the image.

In the experiments carried out in this thesis, I use and compare two different matching techniques, normalized cross correlation and difference of sum of squares on histograms of oriented gradients [Dalal and Triggs, 2005]. The results obtained using these metrics will be discussed in chapter 7.

The size of the region of the image across which the fragment is slid and matched plays a very important role in the results. If the region of interest is too large, the

fragment loses its compositional meaning and its role in detection of the pose weakens. On the other hand, the region of interest cannot be too small either as the variations within the pose group would not be properly captured. For example, if you are looking for a part of an elbow across an entire image, you might find it at many places (ex. knee) and if you look for it at a very specific location, you might not find it due to a variation in pose.

Once the matches between the fragment and all the examples are computed, the fragment matching threshold needs to be determined. That is, in order to decide whether the fragment is present in an image, the matching value needs to be compared with a threshold. This threshold is found by maximizing the mutual information [Cover et al., 1994] between the fragment and the given class.

The mutual information between the two binary variables, $\varphi(\theta)$ and ζ, is defined as follows:

$$MI(\varphi_i(\theta_i);\zeta) \equiv \sum_{\substack{\varphi_i(\theta_i)=\{0,1\} \\ \zeta=\{0,1\}}} p(\varphi_i(\theta_i);\zeta) \log \frac{p(\varphi_i(\theta_i);\zeta)}{p(\varphi_i(\theta_i))p(\zeta)} \tag{3.1}$$

Note that the mutual information is a function of two random variables, $\varphi_i(\theta_i)$ and ζ. $\varphi_i(\theta_i)$ is a binary function of the fragment threshold parameter θ_i indicating whether the fragment is found in an image:

$$\varphi_i(\theta_i) = \begin{cases} 1 & M(I,fr) \geq \theta_i \\ 0 & otherwise \end{cases} \tag{3.2}$$

where $M(I,fr)$ is the best match between a fragment(fr) and an image(I). ζ simply denotes the example class. If the threshold parameter θ_i is too low, the mutual information between the fragment and the class is going to be low because the fragment will be found with high frequency in both positive and negative examples. Similarly, if the threshold is too high, the fragment will rarely be found even in the positive examples. Therefore, there exists an intermediate point at which mutual information

reaches a maximum, which is then the point picked for the fragment threshold.

$$\theta_i = \arg\max_{\theta_i} \sum_{\substack{\varphi_i(\theta_i)=\{0,1\} \\ \zeta=\{0,1\}}} p(\varphi_i(\theta_i);\zeta) \log \frac{p(\varphi_i(\theta_i);\zeta)}{p(\varphi_i(\theta_i))p(\zeta)} \tag{3.3}$$

Once the thresholds are determined, a binary decision on whether the fragment is present in the example image can be made for each fragment-example pair.
Now that the thresholds are computed, the system chooses the best set of fragments. This is done using a boosting algorithm [Freund and Schapire, 1997].

Boosting

The boosting algorithm comprises the following stages:

1. First, all the examples (positive and negative) are assigned weights. If there are p positive examples and n negative examples, each positive example is given the $weight = \frac{1}{2\times p}$, and each negative example is given the $weight = \frac{1}{2\times n}$, in order to account for possible bias in the number of examples on each side.

2. Then, for each fragment we compute the error, which is the sum of weights of all the positive examples in which fragment is not contained and all the negative examples in which the fragment is contained. The fragment with the minimum error is picked. The saved information includes the fragment, its region of interest, its threshold, and the parameter $\alpha = \frac{1}{2}\ln\frac{1-error}{error}$.

3. Then, all the example are reweighted such the weight of the examples that were classified correctly by the previous fragment is decreased, and the weight of the examples that were classified incorrectly is increased. Finally, the whole process is repeated with the new weights. The process can be repeated arbitrarily many times. In the system described in this thesis, I repeat the process ten times. I found that after ten iterations the error does not decrease significantly.

This process forms a classifier made up of the series of decisions made by individual

fragments. Each decision has a weight equal to the α parameter of the fragment. Sum of all the weights represents a number based on which the final decision can be made. Very often, the way the decision is made is by comparing the sum of weights with 0. In the system described here, I compute the decision threshold for the classifier by maximizing the mutual information between the classifier and the pose class. I found that this last step improves the classification.

At this point, a binary pose classifier is trained to estimate whether the pose looks more like pose R or pose W. The binary pose classifier consists of image fragments, their respective thresholds and regions of interest, the boosting weights for each of the fragments and the boosting decision threshold. An example of some fragments found when training pose R against pose W classifier is shown in figure 3-2.

Figure 3-2: Example fragments found in training binary poseR-poseW classifier

Figure 3-3 shows the complete process of developing a binary classifier.

In summary, after processing the incoming image, fragments of different scales and at different positions are extracted. Then, the match between fragments and the examples is computed in order to determine the threshold for each fragment. Then, the most informative fragments are picked through a boosting method. Finally, a binary classifier is formed based on the most informative fragments, the fragments' thresholds, the regions of interest and the boosting weights as well as an overall boosting decision threshold.

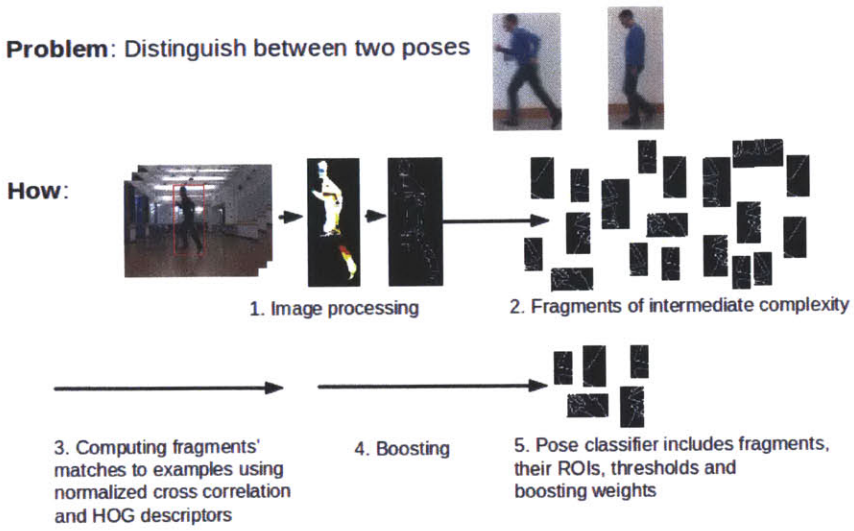

Figure 3-3: Summary Figure for Two Pose Classification

3.3 Case with N Poses

Everything discussed so far in this chapter represents the process of forming a binary classifier. However, in the case of N poses we need to form a pose classifier that can label the given image as any of the N poses. This section describes the second stage in developing a pose classifier.

From Binary Classifiers to Hierarchical Group Tree

In the first stage of the training, a binary classifier is trained for every pair of poses. This produces N^2 classifiers from N poses. Then, I take all the examples and evaluate them by each binary classifier. Based on that, each $pose_i - example_j$ pair gets a score equal to the number of binary classifiers that classified the $example_j$ as $pose_i$. The pose that has the highest score for a given example is picked as an estimate pose for that example. When all the examples are evaluated, an $N \times N$ matrix is computed where each cell (i, j) represents a number of hidden poses i (each example is an example of a particular "hidden" pose) that were estimated to be pose j.

For example, imagine that there are three poses (A, B and C) and twelve examples (4 example of each of the poses). Also, imagine that all examples of pose A were classi-

	Pose A	Pose B	Pose C
Pose A	4	0	0
Pose B	1	2	1
Pose C	2	1	1

Table 3.1: Example of $N \times N$ matrix where each cell (i, j) represents a percentage of hidden poses i (each example is an example of a particular "hidden" pose) that were estimated to be pose j

fied (correctly) as pose A, two examples of pose B were classified (correctly) as pose B and two were classified (incorrectly) as pose A and C (one of each), one example of pose C was classified (correctly) as pose C, two were classified (incorrectly) as pose A and one as pose B. Then, the 3×3 matrix would look like that in table 3.1.

Based on the $N \times N$ matrix of results, we can compute a *closeness* matrix. The closeness matrix represents the measure of closeness between any two poses or more specifically how difficult it is for a classifier to distinguish between any two poses. The element of the closeness matrix is defined as follows:

$$C[i,j] = \frac{R[i,j]}{\sum_j R[i,j]} + \frac{R[j,i]}{\sum_i R[j,i]} \tag{3.4}$$

where C is the closeness matrix, R the results matrix. The equation is symmetric for i and j, so $C[i,j] \equiv C[j,i]$.

Once the closeness matrix is computed, I group the two poses i and j that are closest to each other. Then I consider that group as one pose, leaving me with a total of $N-1$ poses. The binary classifiers between the other poses and this newly formed grouped have to be trained. For efficiency reasons they are trained using the fragments left from binary classifiers corresponding to poses i and j. The process is repeated until only two groups are left at which point a hierarchical binary group tree is naturally formed. Every non-leaf node in the tree represents a pose group and has associated with it a binary classifier that splits the group into two smaller groups. Figure 7-4 in chapter 7 shows an example of a hierarchical tree obtained during the training stage. The hierarchical group tree represents an N-pose classifier. When a new image is

ready for classification, it is passed through the nodes of the tree until it reaches the final pose. An alternative to this method would be to simply look at the results matrix and decide on the pose based on that.
However, the advantage of the hierarchical tree step is twofold:

1. First, at run-time, the hierarchical method takes much less computation time. If we label an average binary classifier computation time as B, In the case of a balanced hierarchical group tree, the computation time is of the order of $B \times \log N$ and in the worst case of the order of $B \times N$, while the computation time for the results matrix is of the order of $B \times N^2$.

2. Second, if two poses are close to each other, it is more reasonable to distinguish between them based on the result of the binary classifier between them and not based on the results of the classifiers that were trained between one of those poses and other poses.

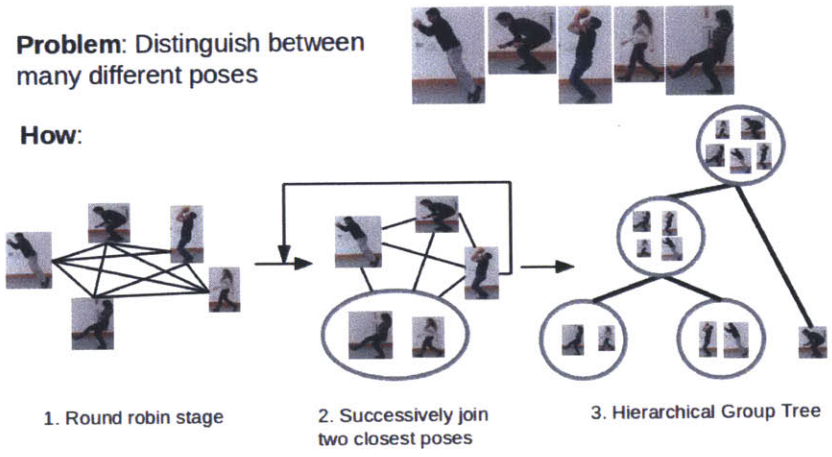

Figure 3-4: Stages in N-pose classification

Summary

In this chapter I showed how to develop a pose classifier in two stages:

1. First, I employed Ullman's intermediate features principle to develop a binary classifier for any two poses.

2. Second, I developed a technique for combining all the binary classifier into a hierarchical group tree that can be used for N-pose classification

In chapter 7, I present detailed results of the N-pose classification system obtained on two datasets, noting that I achieve 77% recall rate on my 10-pose dataset and 68% recall rate on the 9-pose KTH public data. I also show that hierarchical group tree representation is very effective in capturing differences between poses and discuss how this feature can be used to develop an automated pose selection mechanism.

Chapter 4

Hidden Markov Model for Exploitation of Structural Constraints

In chapter 3, I described the technique for the classification of independent single frame images. Now I turn to the temporal connection between frames. The temporal connection between the frames represents a very important aspect that can be leveraged not only for better action recognition, but also for an improved pose classification. Human motion is naturally constrained in a temporal domain. For example, it is highly unlikely that a human body can transition from a crawling pose to a running pose in one frame time. That kind of structural constraint in time can be used to make better estimates about the single frame poses. Furthermore, the structural constraints in human motion can be used to further improve the independent single-frame classifier by establishing a feedback training mechanism.

A natural way to model the transitions between poses is through a hidden Markov model. Hidden Markov model represents a stochastic state transit model [Rabiner and Juang, 1986] in which every state depends directly only on the state before that.

4.1 Hidden Markov Model

A hidden Markov model comprises a sequence of hidden states and a sequence of noisy state observables. The transitions between the states are described by a probability distribution that specifies the probability of transition from a state to any other state. The likelihood of observations is described by a probability distribution that specifies the probability of an observation given any hidden state. States in the HMM are not directly observable and can only be inferred from the sequence of noisy observations.

States

In the context of action sequences, the poses represent the states in the HMM. That is a very natural way to represent poses because the poses have all the characteristics of states in this problem. Poses are not directly observable, can be inferred only from the sequence of observations, and a pose at time t depends directly only on the pose at time $t-1$. Figure 4-1 shows a general structure of an instantiated HMM where each time point is represented through two random variables, pose[t] and O[t]. Pose[t] is a hidden state variable at time t and O(t) is observation at time t. The arrows in the graph represent conditional dependencies.

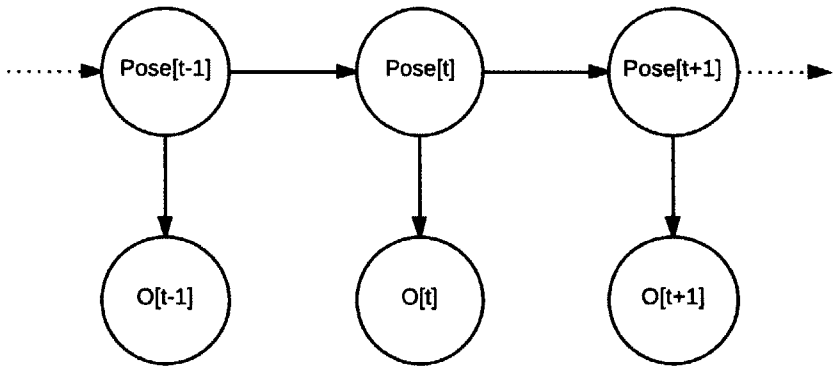

Figure 4-1: Hidden Markov model represented as a Bayesian net

Transition and Emission Probability Distributions

The hidden state space is made of N possible different states so the transition and emission probability distributions have to be described for this model to be instantiated. The emissions probability distribution specifies the likelihood that pose[t] will generate O[t] for all possible poses and observations.
That means that the pose classifier can, with a certain probability, estimate a hidden pose state to be any of the N poses. The emission probability distribution is estimated through training. After the pose classifier is trained, new single frame examples of each of the N classes are given to the classifier and the emission probability distribution is estimated directly in the following way:

$$P(O = o|Pose = p) = \frac{S(p)}{N} + (1 - S(p))\frac{P(o,p)}{P(p)} \tag{4.1}$$

where $P(o,p)$ is the number of observations o when p is the hidden pose, $P(p)$ is the total number of hidden poses p in the training set, $S(p)$ is a function of the number of training examples that ensures that probability distribution for any observation o does not go to 0. It is constructed such that the second term in equation 4.1 becomes more dominant as the number of training examples grows:

$$S(p) = a + \frac{b - a}{\sqrt{p}} \tag{4.2}$$

where $a = 0.05$ and $b = 0.5$ are the values used in the training, in the system described in this thesis.

The transition probability distribution specifies the likelihood of a transition from pose[t] to pose[$t + 1$] for all possible pairs of poses. Much like the emissions distribution, the transitions probability distribution is estimated through training. However, in this case, the training is done simultaneously with the pose classification training. The transitions for all pairs of poses are counted and similar equations to those in 4.1 and 4.2, substituting observations o for poses p', are applied. Figure 4-2 shows a

visualization of the transition probabilities (in black/heavy) and emission probabilities(in red/light) for the case of three poses.

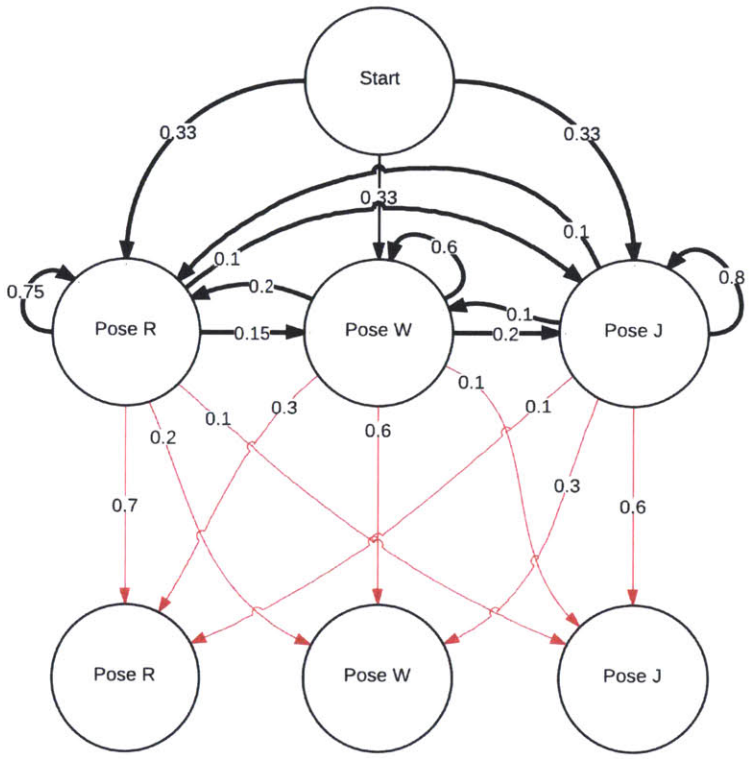

Figure 4-2: Visualization of the transition probabilities (in black/heavy) and emission probabilities(in red/light) for the example with three poses.

In this thesis, the transition and emission probability distributions are found through training and are not changed after that; however, it would be valuable to explore how performance of the system varies if we allow these distributions to be further trained online (during the run time of the algorithm).

4.2 Inference

Given the parameters of the model, the states, observations and all the probability distributions, the maximum likelihood sequence of hidden states is computed using the Viterbi algorithm [Viterbi, 1967].

Viterbi algorithm

The Viterbi algorithm finds the most likely sequence of hidden states given the sequence of observations. The algorithm is run iteratively over the hidden states (S) in the sequence. The recursive process can be defined formally as a maximization problem:

$$\begin{aligned} V_{t,s_i} &= P(o_t|s_i) \times max_{s_j \in S}(T_{s_j,s_i} \times V_{t-1,s_j}) \\ V_{1,s_i} &= P(o_1|s_i) \times \pi_{s_i} \end{aligned} \tag{4.3}$$

where $s_i \in S$, o_t is an observation at time t, T_{s_j,s_i} is an entry in the transition matrix that specifies the likelihood of transitioning from s_j to s_i, $P(o_t|s_i)$ is an entry in the emission matrix that specifies the likelihood of observing o_t when the hidden state is s_i, and V_{t,s_i} is probability of the most probable state sequence given the first t observations that has s_i as its final state.

The Viterbi algorithm at each iteration t examines all the possible states s and for each of them examines all the possible transitions that could have led from time $t-1$. It then determines, for each state s, the most likely transition based on the emission and transition matrices. Only the most likely transition is then stored (inside the V matrix) and affects the next step.

Given the complete V matrix, one can easily compute the most likely sequence of states. The last state is given by:

$$s^{end} = \arg\max_{s \in S} V_{T,s} \tag{4.4}$$

Given the last state, the most likely path of states can be reconstructed following the most likely transitions at each stage. This is typically stored in the process of computing the matrix V so that it can be easily restored.

As discussed in the results chapter later in this thesis, modelling the structural constraints significantly improves the results of the pose classification. This result can

be used in many different ways. First, instead of doing the correction, we can use prediction so as to reduce the amount of work that needs to be done by the pose classifier by reducing the pool of poses that can occur at any particular position.
More importantly, this result can be used to further improve the pose classifier by feeding the missed examples back to the pose classification training stage. The examples that are classified incorrectly are the most valuable examples for improving the classifier. They serve as near miss examples, useful to specialize or generalize the class representation. In his landmark thesis [Winston, 1970], Patrick Winston noted that becoming an expert requires repair as well as acquisition. He shows that exploiting situations when the already acquired knowledge makes a mistake, enriches the model.
In this thesis, I did not implement a framework that would feed near miss examples back to the training process, but I believe that development of such a framework can serve as a great starting point to some future work.

Summary

In this chapter I described the process of using a hidden Markov model to develop a model of transitions between poses in a video sequence. I explained how this model is used to correct the errors introduced through the pose classifier and showed how it can be used to further improve the results of the single-frame pose classification. In chapter 7, I demonstrate that modelling regularities in pose sequences improves the recall rate for pose classification by an average of 15%. I also discuss the implications of this result to online training and pose prediction/gap-filling.

Chapter 5

Sequence Recognition

In this chapter I present the system for action recognition based on the sequence of poses. After the single-frame pose estimation and sequence correction have been completed, the problem simplifies to that of recognizing textual patterns in a sequence. The incoming sequence of poses is first cut into smaller sequences that are then passed to the action detection component of the system. For this purpose, I adopt a sliding window technique that classifies all the possible sequences.

5.1 Problem

Once the single frame poses have been estimated and corrected where necessary to comply with the structural constraints of the human motion, each pose can be considered as a letter in a letter sequence. It is important to emphasize that there is no one to one mapping between poses and actions which implies that one pose can appear in many actions and one action typically contains multiple poses. However, it is the sequence in which these poses appear that determines the action. The poses can be thought of as elementary operations (building blocks) that, when combined together, produce different actions. For example, suppose that there are 3 letters in the pose alphabet, R, T and W. Then actions such as running and walking could possibly be built in the following way:

Running $\longrightarrow$ RRRTTWWRRRTTTRRR or RRRTTRRRRWWWRR

Walking $\longrightarrow$ TTWWWWWTTRRWWWWW or WWWWWWTTTWWWWW

In the example above the frequency of certain poses can be used to deduce the action class of the sequence. In running, we see more of R and T poses and in walking more of W and T poses. Furthermore, the transition between poses are different; while in one there are mostly transitions between R and T, in the other most of them occur between W and T. The transitions are important also because of the nature of simple action pose composition. That is, the speed with which different people go through the poses can vary significantly, while the structure typically does not vary too much (a person cannot just skip over a certain pose in running, but they can spend more or less time in it depending on the style).
This means that, in order to recognize actions, the sequence representation needs to be capable of capturing the transitions between poses.

5.2 Action Detection

In order to build a representation that can capture transitions between poses in a pose sequence, I looked at two approaches to action detection:

1. An unsupervised approach based on hierarchical clustering
2. A supervised approach based on learning a model of actions given the training examples.

5.2.1 Unsupervised Action Learning

In the unsupervised approach, the idea is to cluster incoming sequences based on the sequence patterns, without prior knowledge of the sequence action labels. In order to do this, I had to define a set of features that were extracted from the incoming sequences. The criteria for the feature selection is the ability to capture pose transitions well. In this work, I tested two different methods, the Levenshtein distance and the bag of features vector distance which will be described later in this section.
Without prior knowledge of the action labels, the system has no information about

the number of action categories. Because of that, I decided to apply the method of Agglomerative Hierarchical Clustering which does not assume a prior knowledge of the number of clusters.

Agglomerative Hierarchical Clustering

Agglomerative Hierarchical clustering is a bottom-up clustering method where each point starts off as its own cluster and at each iteration the closest two clusters are combined until only one big cluster is left. In figure 5-1 you can see the result of clustering a set of numbers $[1, 2, 4, 6, 10]$.

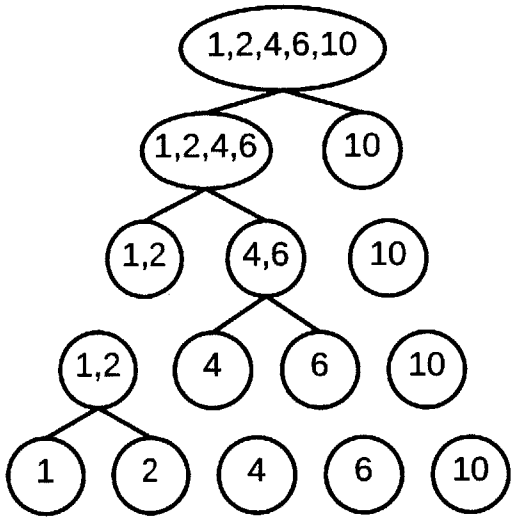

Figure 5-1: Example of levels in hierarchical clustering algorithm

As a result of hierarchical clustering, a binary tree of points is built in which similar clusters are successively merged, decreasing the number of clusters at each level. The only parameter needed for the clustering method besides the data points is the distance metric. Typically, the Euclidean distance between cluster centers is used for data that can be represented through vectors. However, in the case of actions represented through a sequence of letters, the Euclidean metric cannot be used. Instead, I use two alternative measures of distance between the data points:

1. Levenshtein string distance

2. Bag of features vector distance

Levenshtein distance is a measure of similarity between two strings that represents the number of single-character edits (insertion, deletion, substitution) required to change one string into the other. Formally:

$$LEV(S_1, S_2) = \begin{cases} |S_1| & |S_2| = 0 \\ |S_2| & |S_1| = 0 \\ min \begin{cases} LEV(S_1[:-1], S_2) \\ LEV(S_1, S_2[:-1]) \\ LEV(S_1[:-1], S_2[:-1]) + (S_1[-1] \neq S_2[-1]) \end{cases} & otherwise \end{cases} \tag{5.1}$$

where $|S|$ is the length of a string S, $S[:-1]$ is the string S without the last character and $S[-1]$ is the last character. Levenshtein distance represents a measure of similarity between two string sequences corresponding to two actions. However, in the case when sequence length varies substantially, Levenshtein distance turns out to be a very biased measure because it is not normalized in its original form.

Bag of features vector distance is similarly a measure of distance between two strings based on the strings' features. The features are simply frequencies of the patterns in the string. First a collapsed string is made out of the original string. A collapsed string is a string in which consecutive occurances of the same characters are replaced with one character. For example, string RRRRUURRNIIINNNG would be collapsed into RURNING. Now from both of these strings, the original and collapsed, substrings of different lengths (ranging from 1 to the length of the string) are extracted. I count the frequency of the substrings and that frequency map becomes the bag of features vector.

$$b_S(s_i) = \frac{count(s_i, S)}{|S|} + \frac{count(s_i, c(S))}{|c(S)|} \tag{5.2}$$

where $c(S)$ is S-collapsed string and $count(s_i, S)$ is the number of time substring s_i appears in string S.

Now that I explained how to extract the bag of features vector from a string, I describe

how to calculate a distance between two strings using their vectors. It is similar to a simple square distance between the vectors with the two added components accounting for the elements in a vector that are not present in the other one.

$$d(S_1, S_2) = \sqrt{\sum_{\substack{s_i | \\ s_i \in S_1 and s_i \in S_2}} (b_{S_1}(s_i) - b_{S_2}(s_i))^2 + \sum_{\substack{s_i | \\ s_i \in S_1 and s_i \notin S_2}} b_{S_1}(s_i)^2 + \sum_{\substack{s_i | \\ s_i \in S_2 and s_i \notin S1}} b_{S_2}(s_i)^2} \tag{5.3}$$

This distance measure has advantage of being normalized and applies equally well to the case when there are significant variations in the length of the sequences. Furthermore, use of collapsed strings helps emphasize the importance of transitions in sequence recognition. As mentioned in the introduction to this chapter, transitions between poses play a key role in understanding the action.

In addition to specifying the distance metric between two sequences, I also need to specify the distance between two clusters. For that purpose I define a cluster center to be the point whose sum of distances to other points in the cluster is minimum across all the points in that cluster.

$$C_c = \arg\min_{S_* \in C} \sum_{S_j \in C} d(S_*, S_j) \tag{5.4}$$

where C_c is the cluster center of cluster C. Then the distance between two clusters is simply the distance between their center points.

After the distance metric and a way of computing cluster centers are defined, the agglomerative hierarchical clustering can be applied. The clustering method outputs a tree that can be used to infer the quality of selected features for the sequence separation.

In summarry, the incoming sequences are first processed by adding their collapsed counterparts to the sequence set. Then, the string features are extracted. A feature is a frequency of a substring in the original string and its collapsed counterpart. Once the features are extracted, the distance metric between any two string can be computed. Finally, the clustering produces groups of similar sequences, which

are used to infer the quality of extracted features. In chapter 7, I show that bag of features is a representation that can capture most of the variability between the action sequences and seperate them well (73% accuracy on the set with 8 actions). However, the amount of captured information seems to be limited which suggests that supervised learning might be necessary.

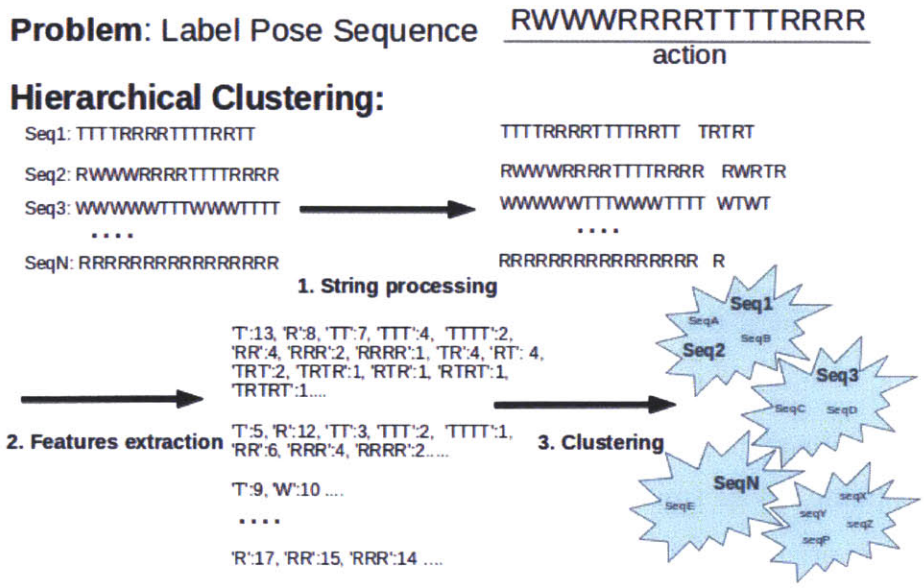

Figure 5-2: Summary of the unsupervised action detection method

5.2.2 Supervised Action Learning

In this section, I present the supervised method for sequence classificaton using the hidden Markov models. I first assume that there is one model for each of the action categories and show how to determine the most likely one given the new unlabelled sequence. I show that forward-backward algorithm can be effectively used for this task. Then, I explain how to train hidden Markov model to produce one model per action category using the Baum-Welch algorithm.

Estimating the most likely action model

First, assume that each action category has associated with it a set of parameters that define a hidden Markov model. Those parameters, as discussed in chapter 4, include a set of states, a set of observations and the probability distributions describing transitions between the states, the emissions of observations and the initial states. For the rest of this chapter I will refer to these parameters collectively as $\lambda = \{T, E, \pi\}$ where T, E and π represent the transitions, emissions and initial state probability distributions, respectively.

Given a sequence of observations O, the problem of recognizing an action category that these observations belong to can be formulated as:

$$C = \arg\max_i P(\lambda_i|O) \tag{5.5}$$

Term $P(\lambda_i|O)$ is, following the Bayes rule, proportional to $P(O|\lambda_i)$. Knowing the parameters of the model that best describes a particular set of actions, I need to estimate how likely a certain sequence is to be part of that model. This can be achieved by using a forward-backward algorithm which computes the likelihood of all the hidden states given a sequence of observations. The algorithm comprises three stages. First, in the forward stage, the probability of any given state is computed given the observations that preceded it. Then, in the backward stage, the probability of any given state is computed given all the observation following it. Finally, the smoothed probabilities are computed combining both forward and backward estimates.

Applying the forward-backward algorithm for a given sequence and all the models, allows me to estimate the model that best fits the observed sequence.

Baum-Welch algorithm for training action models

Now that I explained how to estimate which model is the most likely given the observation sequence, in the next paragraph I explain how to train the hidden Markov models, one for each class of actions.

The problem can be formulated as estimating the parameters most likely to generate

the training data. For the purpose of training the hidden Markov models I adopted the Baum-Welch algorithm [Baum et al., 1970]. The Baum-Welch algorithm is a special case of a more generalized expectation-maximization algorithm which solves the problem by finding a locally optimal solution over an infinite space. The Baum-Welch algorithm is particularly useful for the problem of action sequence recognition because of the variable nature of the length of the patterns within the same class of actions. Because the likelihood is calculated from the entire sequence, variations in length, shifts in sequence and small errors have little effect on the likelihood of a model given a set of observations.

The Baum-Welch algorithm is the process in which the parameters λ are optimized such that the quantity $p(O|\lambda)$ is maximized. For this purpose, two variables are introduced, $\xi_t(i,j)$ and $\gamma_t(i)$. $\xi_t(i,j)$ represents the probability of being in state i at $t = t$ and in state j at $t = t+1$.

$$\xi_t(i,j) = p(s^t = i, s^{t+1} = j|O,\lambda) = \frac{p(s^t = i, s^{t+1} = j, O|\lambda)}{p(O|\lambda)} \tag{5.6}$$

$\gamma_t(i)$ is the probability of being in state i at $t = t$, given the observation sequence and the model.

$$\gamma_t(i) = p(s^t = i|O,\lambda) = \frac{p(s^t = i, O|\lambda)}{p(O|\lambda)} \tag{5.7}$$

Using these two equations, the parameters λ are improved to λ' such that:

$$\begin{aligned} T'_{i,j} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ E'_i(s) &= \frac{\sum_{t\in\{t|O_t=s\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \\ \pi' &= \gamma_t(i) \end{aligned} \tag{5.8}$$

This process is then repeated until the difference between the quantities decreases to a certain preset value. At that point, we can be sure that we reached a local maximum. An example of the process that occurs in the iterations of Baum-Welch algorithm is shown in figure 5-3. The diagram demonstrates the steps in learning action "walk-

ing" from the training examples in SHA dataset containing 8 actions and 10 different poses. The strength of arrows is proportional to the probability of particular transitions. In illustration A, all the transitions are equally likely which represents the state when the model has not learned anything. As more iterations are performed, some transitions become less likely and some become more likely. Finally, when the process is finished, the model parameters are optimized to describe the training sequences. In illustration D, we can see the most likely pose and a few other poses that occur in "walking". The model that was learned using the Baum-Welch algorithm suggests that in "walking", the most time is spent in one pose (the one on top), with occassional drifts into three other poses. In the periods when the model is not in the pose on top, there is a strong tendency to come back to it.

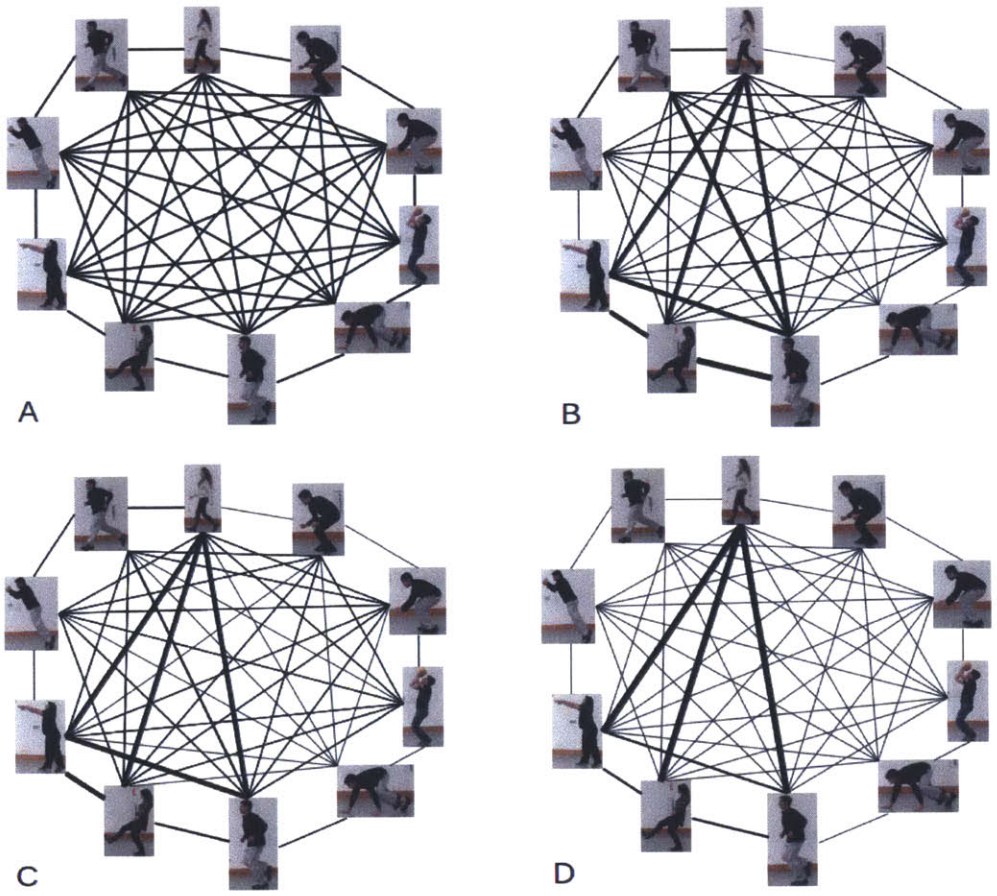

Figure 5-3: An example of different stages in the run time of the Baum-Welch algorithm during "walking" learning.

Figure 5-4 shows an example of learning "jumping" structure using the Baum-Welch algorithm. In contrast to "walking", the model of "jumping" is characterized by strong transitions, which indicate the path through poses. While in "walking" there is a tendency to come back to a single pose, in "jumping" we see a tendency to come back to a path.

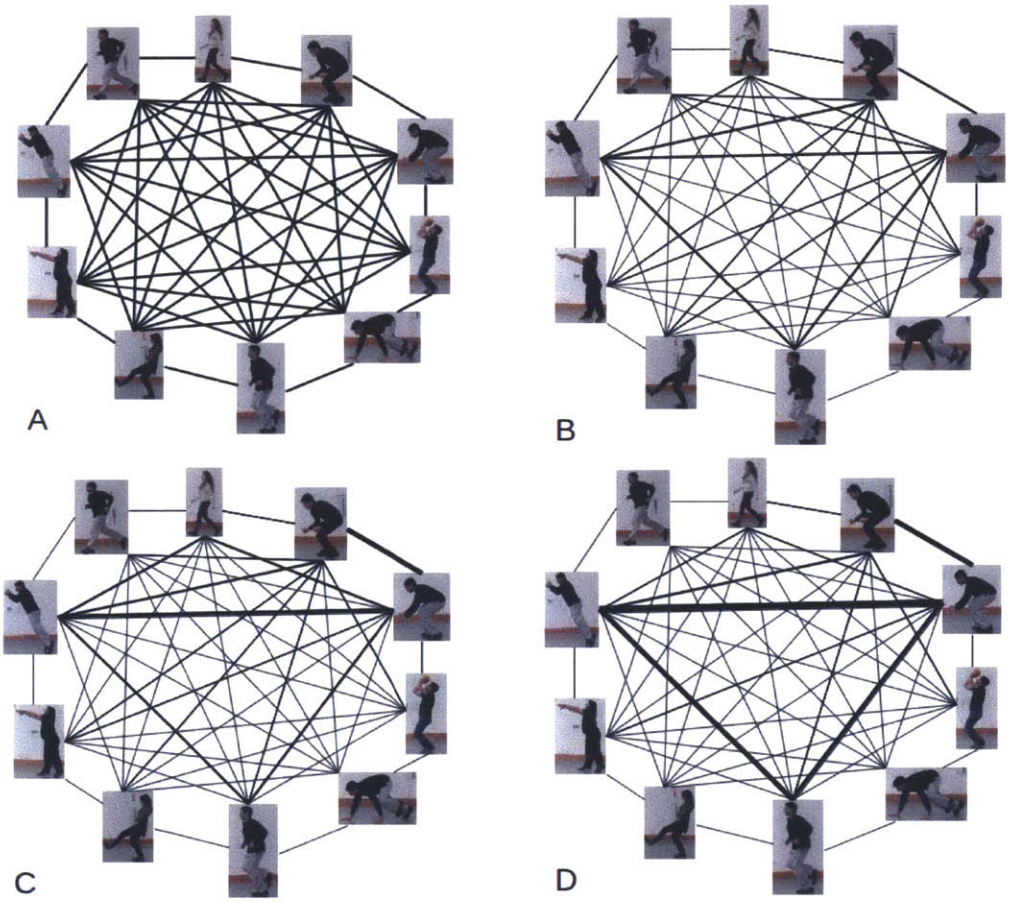

Figure 5-4: An example of different stages in the run time of the Baum-Welch algorithm during "jumping" learning.

In summary, the supervised action detection comprises the training stage and the classification stage. In the training stage, a set of sequences is given for each action category. Using the Baum-Welch algorithm, the system produces HMM models, one per action category. The parameters of these models are optimized to capture the representation of the category as described through the training examples. In the classification stage, a new unlabelled sequence is passed to each of the trained mod-

els. For each of the models, the forward-backward algorithm is used to compute the likelihood of the sequence given the model. The model that produces the maximum likelihood is then picked and its corresponding action category is used to label the sequence.

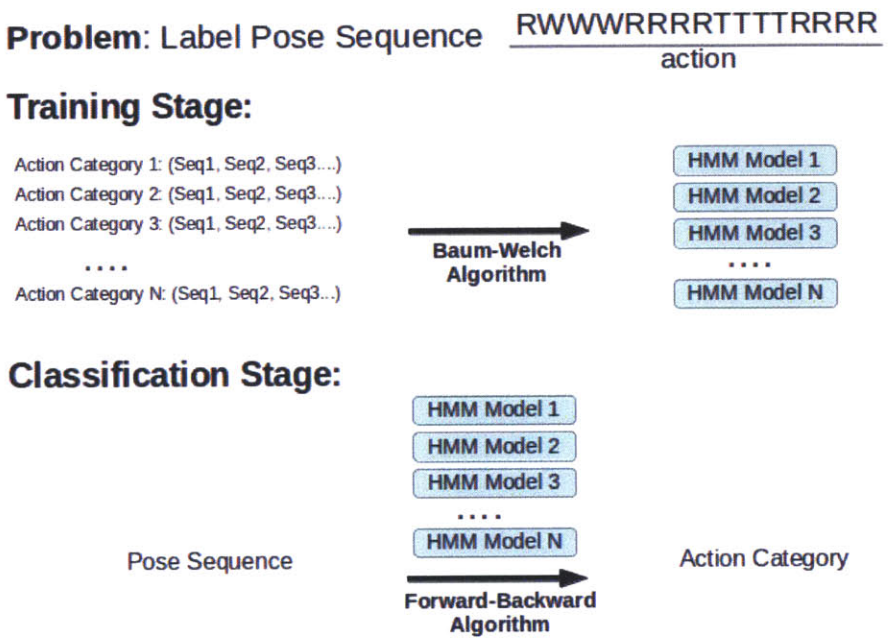

Figure 5-5: Summary of the supervised action detection method

Summary

In this chapter I reduced the action recognition problem to the problem of recognizing a sequence made of the letters from a finite alphabet. I showed how agglomerative hierarchical clustering can be applied to the problem using two different sequence distance metrics. I also showed how hidden Markov models can be trained for a specific class of actions and how such developed models can be applied in inferring the most likely action of the incoming sequence. In chapter 7, I present the results of the action recognition on two datasets and achieve 91% recall rate on the sets containing 6 and 8 actions.

Chapter 6

System Design and Implementation

In this chapter, I describe the problem more formally, in terms of system inputs and outputs and describe each of the components of the system separately. I concentrate on describing the flow of the system and how different components interface with each other. The internal algorithm implementations were previously discussed in chapters 3, 4 and 5 and will not be further explained in this chapter.

6.1 Problem Inputs/Outputs

The input to the problem is a sequence of video frames that contains human actions. The system output is a sequence of action labels together with their start and end indices. I do not consider the problem of finding a bounding box surrounding a person and assume that the coordinates of such a box are given as an input to the system. In my work, I used an open source tool [Vondrick et al.] to label the people in the sequences.

In figure 6-1, the top level system takes a video sequence as input and outputs a set of action labels with start and end times associated with every single label. In order to achieve this functionality, I split the problem into three smaller subsystems that

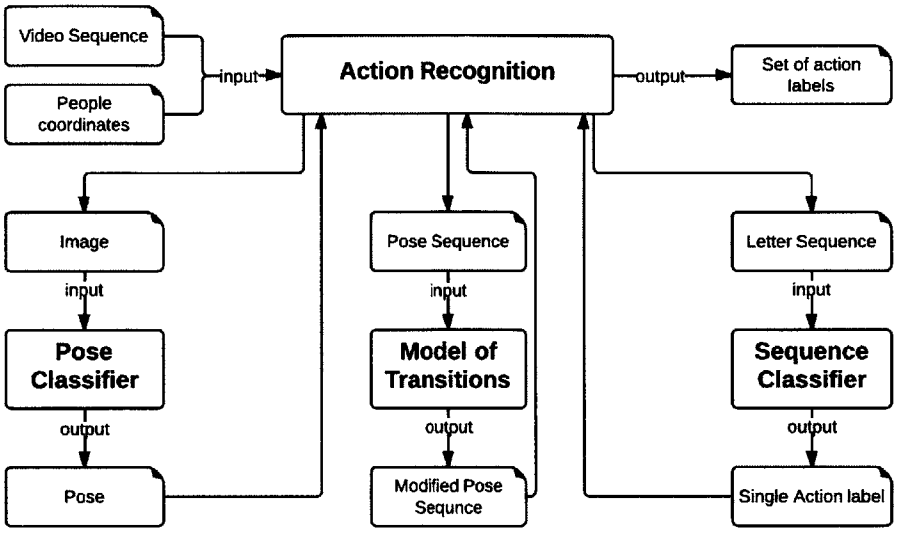

Figure 6-1: High level system design with inputs and outputs for each component

each have their own clearly defined inputs and outputs:

1. The Pose Classifier is a system that takes as input a single frame image and outputs the pose label that best describes a pose of a human in the given image. This system has two layers, the training layer and the classifier layer. Training layer produces a pose classifier based on a set of training inputs, as described in chapter 3. Classifier layer then uses the classifier to select the appropriate pose categories for the incoming images.

2. The Model of Transitions system is one whose role is to modify the sequence of pose labels such that it corresponds to the expected transitions between different poses. This component takes as input a sequence of pose labels and outputs a modified sequence of pose labels. This modified sequence contains the corrections of possible errors in pose estimates that were made by the independent, single-frame pose classifier.

3. The Sequence Classifier is a system required to output a single action label given the sequence of letters from the pose alphabet. The role of this component is to observe the sequence as a whole and find the action label that best describes the given sequence. The system is made of the training layer and the classifier

layer, as described in chapter 5. The training layer produces models of action categories that are then used by the classifier layer to select the most appropriate action label.

The three subsystems described above represent implementations of the work done in chapters 3, 4 and 5. The reason for such a division is to achieve high level of modularity that would allow the improvement of the individual components of the system without affecting the others. In the following sections, I first present the top level Action Recognition system and describe how it interacts with the three subsystems. Then I present the system design for each of the three subsystems separately.

6.2 Action Recognition

The Action Recognition component acts as a controller system for the entire project. It accepts video sequence and coordinates of people in the video sequence as inputs and produces action labels together with their start and end indices for all the actions in the given video. The dependency diagram for the Action Recognition system is shown in figure 6-2.

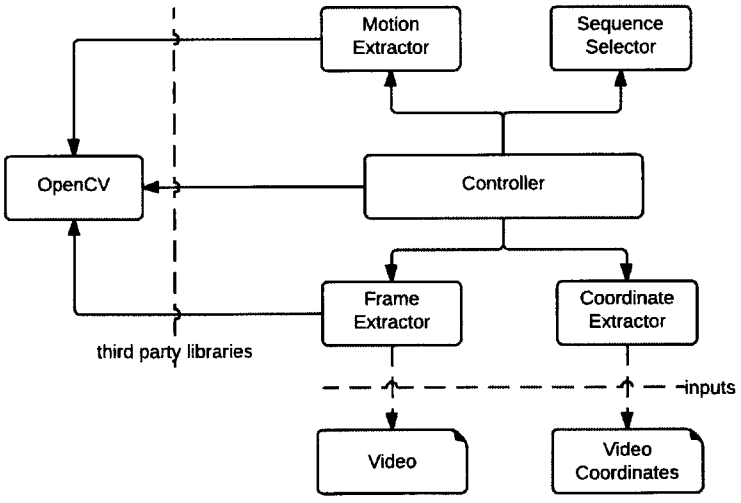

Figure 6-2: The dependency diagram for the Action Recognition system

The diagram shows major classes as they are implemented in the project. OpenCV [Bradski, 2000] is an open-source computer vision library that includes many useful image processing techniques and algorithms. It implements a custom image representation and many useful functions that significantly simplify image processing. Frame Extractor is a class that reads a video and provides frames on demand by frame number. Similarly, Coordinate Extractor is a class that reads a coordinates file and provides functions to get number of people as well as their coordinates in any frame of the video. Motion Extractor provides a motion map between any two frames. Finally, the Sequence Selector class selects sequence of frames to analyze for action. Currently, this class is implemented such that it simply returns all the frame sequences up to a certain prespecified length. Controller then analyzes the given sequences and outputs actions based on the output that it gets from the sequence classifier. All the sequences are analyzed simply because some actions can be contained in the others.

6.3 Pose Classifier

The Pose Classifier system consist of two layers, the classifier layer and the training layer. The dependency diagram for the classifier layer is shown in figure 6-3. The Classification Controller takes as input an image and instantiates an Edge/HOG Extractor in order to extract the proper features from the image. It then uses an instance of the N-pose Classifier, a class that stores the Hierarchical Group Tree representation of the classifier, to classify the input image. That is done by passing the image through a series of nodes in the tree, each of which contains a binary classifier, until a leaf-node is reached. The non-leaf nodes in the tree are represented by a group of poses and a binary classifier that splits the group into two smaller groups. Eventually, the image reaches a leaf node in the decision process. Each leaf node in the tree stores one pose category and the leaf node that is reached in the decision process becomes a designated classification pose label for the input image. An example of a hierarchical group tree is shown in figure 7-4 in chapter 7. The entire classification process is explained in more detail in chapter 3.

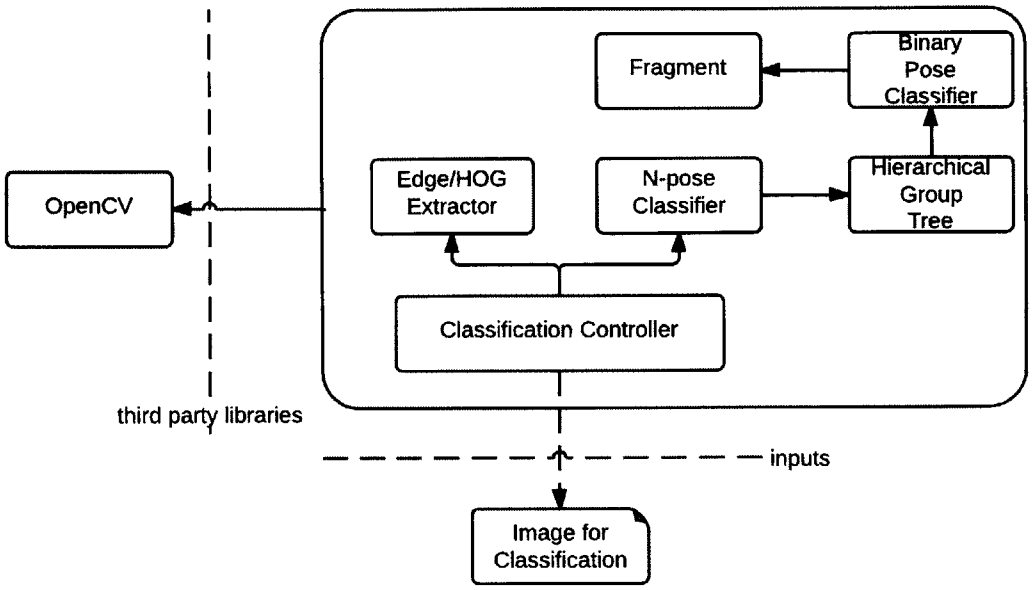

Figure 6-3: Dependency Diagram for the Pose Classifier Classification Layer

The training part of the system works offline and has a role of producing the N-pose Classifier for a set of N poses. The dependency diagram for the training system implementation is shown in figure 6-4.

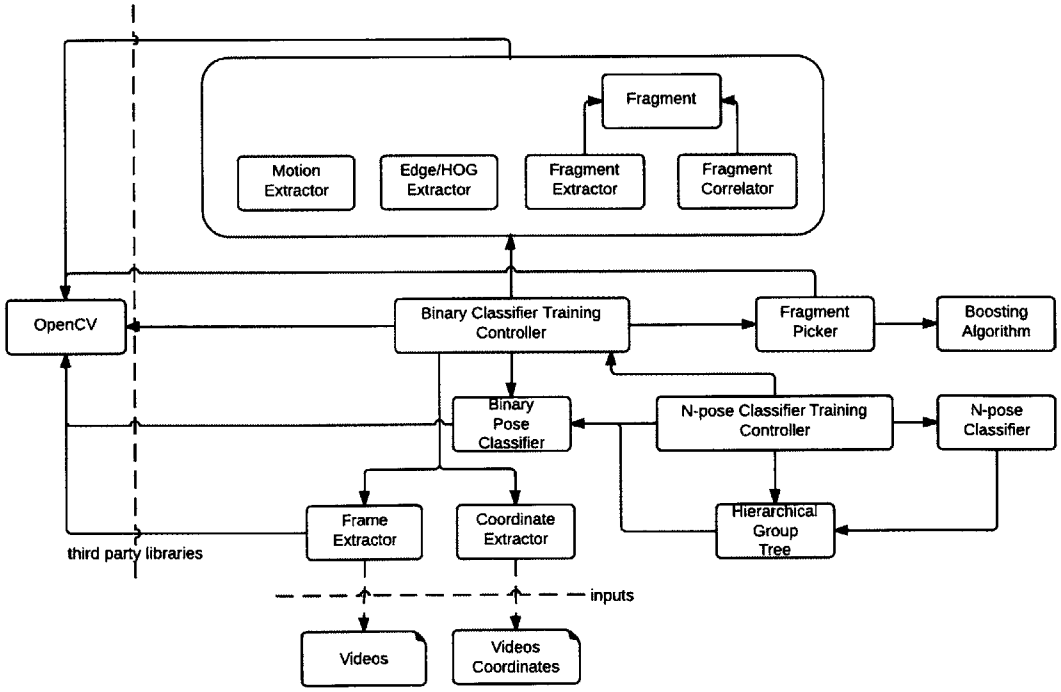

Figure 6-4: Dependency Diagram for the Pose Classifier Training System

Components such as Frame, Coordinate, Motion and Edge/HOG Extractors have already been explained in the previous sections. Their role is mostly to do the initial input reading and image processing.

N-pose Classifier Training Controller is a component which instantiates a Binary Classifier Training Controller for all N^2 pairs of poses. The Binary Classifier Controller forms an instance of a Binary Pose Classifier for each pair and passes it back to the N-pose Classifier Controller, which then forms a Hierarchical Group Tree representation and instantiates the N-pose Classifier as the final output.

Once the Binary Classifier Training Controller has read the input and processed the image, it forms fragments using an instance of Fragment Extractor and matches the fragments with the examples using the Fragment Correlator. Finally, the Fragment Picker is instantiated with all the fragments and the corresponding matches with all the examples. It then determines the appropriate fragment threshold and picks the fragments that are most indicative of a class of poses through a boosting method. Finally, an instance of Binary Pose Classifier is instantiated with the picked fragments. Such instance is then passed back to the N-pose Classifier Training Controller. As mentioned before, the internal details of different components in the Pose Classifier system are discussed in chapter 3.

6.4 Model of Pose Transitions

In this section I explain the system design of the Model of Pose Transitions. In the system described in this thesis, the model is trained offline. This implies that the emission and transition probability distributions are not changed after the training period (during the run time of the system). However, I designed the system such that it can be easily adapted to the online training in which case every new frame would potentially change the parameters of the model.

I use a hidden Markov model to model the pose transitions in the temporal domain. The HMM has two modes, the training mode and the classifier mode. In the training

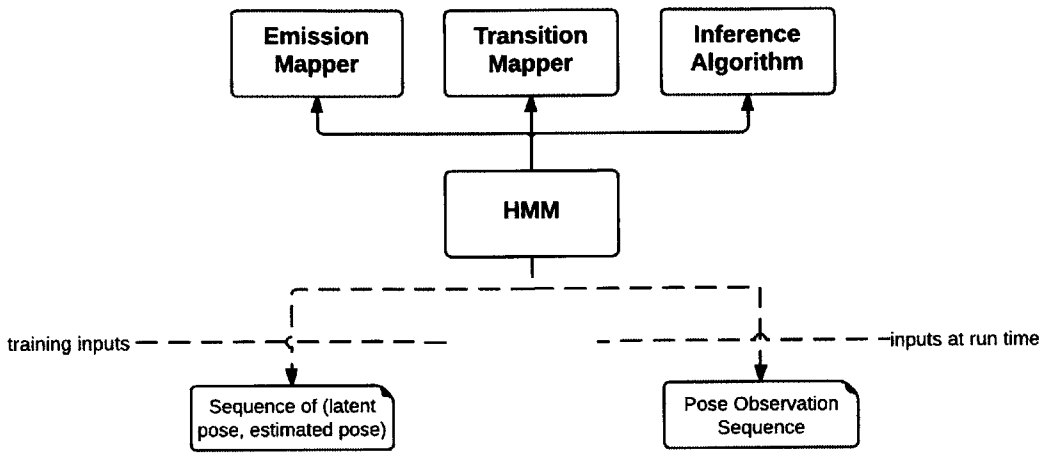

Figure 6-5: Dependency Diagram for the model of pose transitions

mode, it takes as input a set of pairs (latent pose, pose estimation) and uses those to train the Transition and Emission Mappers as described in chapter 4. In the classifier mode, it reads a sequence of pose estimates and relies on the pretrained mappers to get the necessary probability distribution parameters. Using these parameters, it instantiates an inference algorithm (the Viterbi algorithm) to produce a modified pose sequence that complies with the structural constraints of the domain.

6.5 Sequence Classifier

The Sequence Classifier comprises two separate methods for action inference, an unsupervised method based on clustering and a supervised method based on the pretrained HMMs. The Sequence Classifier, much like the other components consists of two modes, the training and the classification mode. In the training mode, the Controller is instantiated with a set of pairs (pose sequence, action label). Those pairs are then used to instantiate a Sequence object, which is further passed to the HMM module. The HMM module then uses the Inference Engine (the Baum-Welch algorithm) to form a model of actions. The Model simply stores a map between an action label and a set of HMM parameters. This process is shown in the dependency diagram in figure 6-6.

At run time, the Controller is instantiated with a pose sequence. It then forms a

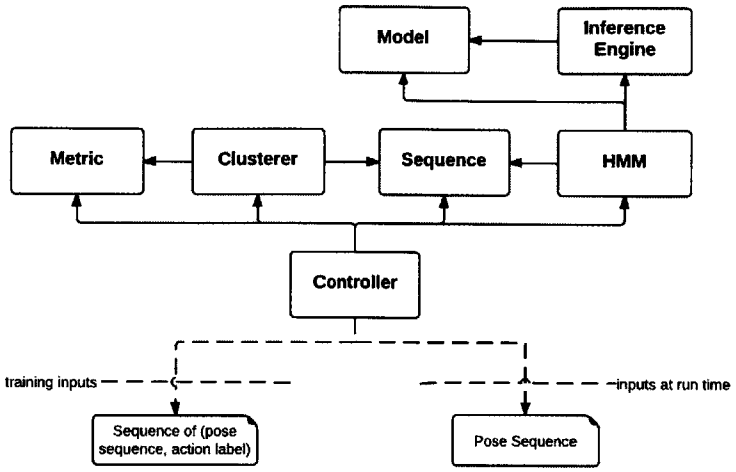

Figure 6-6: Dependency Diagram for the sequence classifier

Sequence instance and uses an HMM module to instantiate an Inference Engine. The Inference Engine uses the forward-backward algorithm to associate a given sequence with an action label whose model parameters best describe the given sequence. The other part of the sequence classifier is the unsupervised clustering module. The controller forms a Sequence and instantiates a Metric class. This class defines a distance measure that is used during clustering. Then, an instance of Clusterer is made with a set of sequences and a metric. The Clusterer then makes the hierarchy of clusters using the agglomerative hierarchical clustering algorithm. This hierarchy is then used to compute the accuracy which represents the amount of variability that can be captured with a set of features used in clustering.

Summary

In this chapter, I explained the system design. In particular, the system design comprises four different subsystems which are developed separately. I explained the interfaces between the subsystems and separately described each of them. In the next chapter I demonstrate the results that the system achieved on several recognition tasks.

Chapter 7

Results

In this chapter I present the results of my work. First I introduce the datasets that were used for system testing. Then, I show the results of the independent, single-frame pose classification. Next, I demonstrate that modelling the constraints of the human motion improves the results of single-frame pose classification. Finally, I present the results of action recognition. I present and discuss the results obtained on a public dataset and introduce a new dataset created primarily for the purpose of this thesis.

7.1 Datasets

In this thesis, two datasets are used:

1. **KTH Dataset**, introduced by [Schuldt et al., 2004], contains six different human actions (walking, jogging, running, boxing, hand-waving and hand-clapping) performed in four different scenarios (outdoors, outdoors with scale variation, outdoors with different clothes and indoors) by 25 subjects. All sequences were taken over homogeneous backgrounds with a static camera with 25fps frame rate. The sequences were downsampled to the spatial resolution of 160x120 pixels and have an average length of four seconds. Figure 7-1 shows an example of the different actions and scenarios in KTH dataset. Typically, the dataset is divided into three components, the training component(8 people), the validation component(8 people) and the testing component(9 people). In this

thesis, I use a sample training, validation and testing data (3 people in each part).

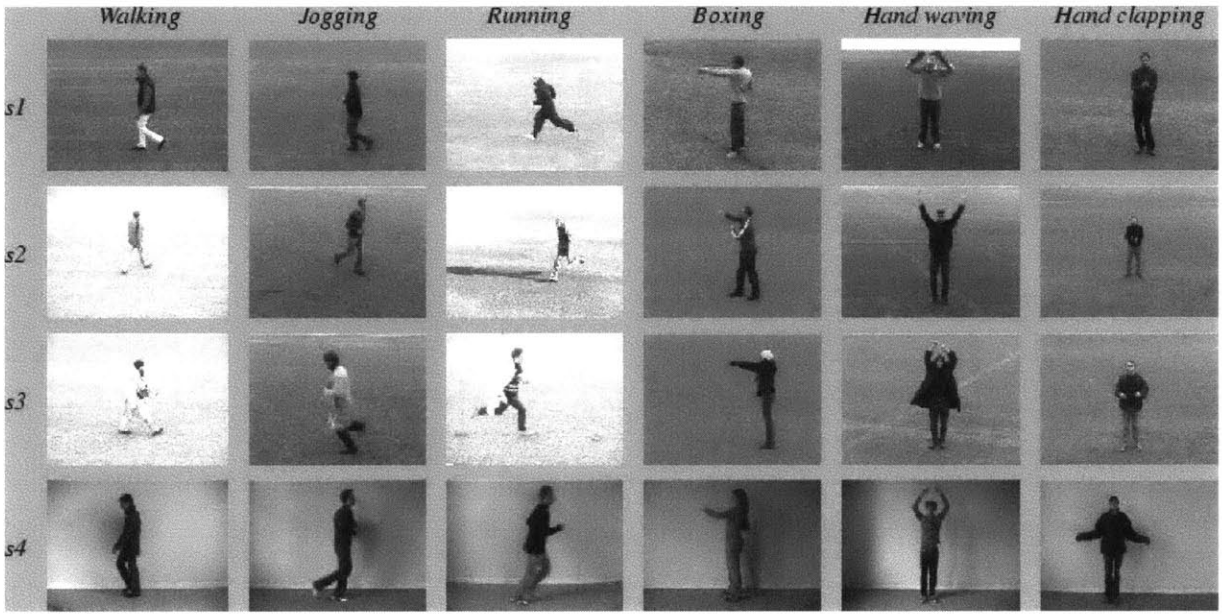

Figure 7-1: Examples of the different actions and the different scenarios in the KTH dataset. This image is taken from [Schuldt et al., 2004]

2. **SHA dataset (Simple Human Actions dataset)** - This is a new dataset that was recorded primarily for the purpose of this thesis. This dataset includes 8 different actions performed by 9 different people. The actions in this dataset are: running, walking, jumping, kicking a ball, passing a basketball, shooting a basketball, crouching and crawling. All sequences were taken over a relatively homogenous background with a static camera with a 30fps frame rate. The resolution at which the data has been recorded is 720x405 pixels. The dataset has been divided into training, validation and testing parts.

In the rest of this chapter I present the results for each of the major components of my work, namely the pose classifier, the model of transitions between poses and the sequence recognition. For each component I present and discuss the results separately for the two datasets.

7.2 Pose Classifier Results

In this section, I summarize the results of the independent, single-frame pose classification. The pose classifier is trained on a certain number of poses from each dataset using a method described in chapter 3.

First, I present the results of pose classification on the SHA dataset. As mentioned above, this dataset has been divided into training, validation and testing parts. The pose classifier was trained on 10 distinct poses (see figure 7-2) extracted from 8 actions.

Figure 7-2: Examples of 10 poses in the SHA dataset

For this dataset, three different image processing techniques and the corresponding fragment-example matching measures are grouped into three approaches:

Approach 1: The system extracts a motion silhouette from each frame. Then an edge map is extracted from the silhouette. A normalized cross-correlation is applied as a matching measure between the fragments and the examples.

Approach 2: The edge map is extracted directly from the image. The normalized cross-correlation is applied as a matching measure between the fragments and the examples.

Approach 3: The original frame image is not modified. HOG descriptor [Dalal and Triggs, 2005] is extracted for each fragment and the normalized square

distance between descriptors is used as a matching measure.

The pose classifier was separately trained for all the approaches listed above. By applying approach 1, I achieved an average of 69% recall rate across 10 poses with an average false positive rate of 2.7%. By applying approach 2, I achieved an average of 77% recall rate with an average false positive rate of 2.0%. Finally, using approach 3, I achieved an average of 61% recall rate with an average false positive rate of 3.9%. The complete results, for all the poses and all the approaches, are presented in appendix in table A.1.

KTH dataset is similarly divided into the training, validation and testing components. From the pool consisting of 6 actions, 9 different poses have been extracted. In the case of KTH dataset, two of the approaches listed above have been tested (Approach 2 and Approach 3). With approach 2, I achieved an average of 54% recall rate with an average false positive rate of 5.2%. Using approach 3, I achieved an average of 68% with an average false positive rate of 3.4%. The complete results, for all the poses and both approaches, are presented in appendix in table A.2.

Table 7.1 summarizes the average results of the pose classification system on the two datasets.

	Approach 1		Approach 2		Approach 3	
	TP*	FP**	TP	FP	TP	FP
SHA	69%	2.7%	**77%**	**2.0%**	61%	3.9%
KTH	N/A	N/A	54%	5.2%	**68%**	**3.4%**

Table 7.1: This table shows the summary of the results in independent, single-frame pose classification for SHA and KTH datasets and three different approaches; Note: TP*-true positives and FP**-false positives

In the pose classification, I tested three different approaches. There are two features that distinguish these approaches:

1. **The matching function**: In approaches 1 and 2 the matching function is the normalized cross correlation, while in the approach 3, the matching function

is the normalized square distance between the HOG descriptors. The normalized cross correlation is a very efficient operation, while the HOG descriptor extraction is a very slow. For this reason, the training using HOG descriptor is modified. The number of examples used in training was reduced, fewer fragments were extracted and the match was performed with 1/5 width of the sliding window. The results show that performance of these two approaches is similar and extracting HOG descriptors does not improve the results. I hypothesize that this is because differences between the poses come mostly from the outside shape of the body in which case edges adequately capture the representation.

2. **The image processing preparation**: In approach 1, I extract the motion silhouette of the person and then extract the edges. This means that mostly outside edges are present in the image that is further treated as an example. In approach 2, the edges are extracted directly from the image. This allows more internal edges to appear relative to the approach 1. The results show slight evidence that a better representation involves a combination of internal edges and outside ones.

The importance of hierarchical group tree representation

The pose classification results on the SHA dataset appear better than the results on the KTH dataset. In particular, slightly lower results on the KTH dataset may be attributed to pose selection. Some of the 9 pose categories that I selected were very similar, while this was not the case to a such degree in the SHA dataset. For example, four different pose categories, shown in figure 7-3, exhibit minimal variation between them. Thus, they are very hard to distinguish.

That simply suggests that the initial pose selection was too broad - 9 poses were selected out of 6 actions in the KTH dataset. However, this broad pose selection can be effectively discovered with the hierarchical group tree representation described in chapter 3.

Figure 7-3: Example of very similar pose categories in the KTH dataset

To illustrate my point, I consider the hierarchical group tree generated in the training stage of pose classification on the KTH dataset using the approach 2 (see figure 7-4)

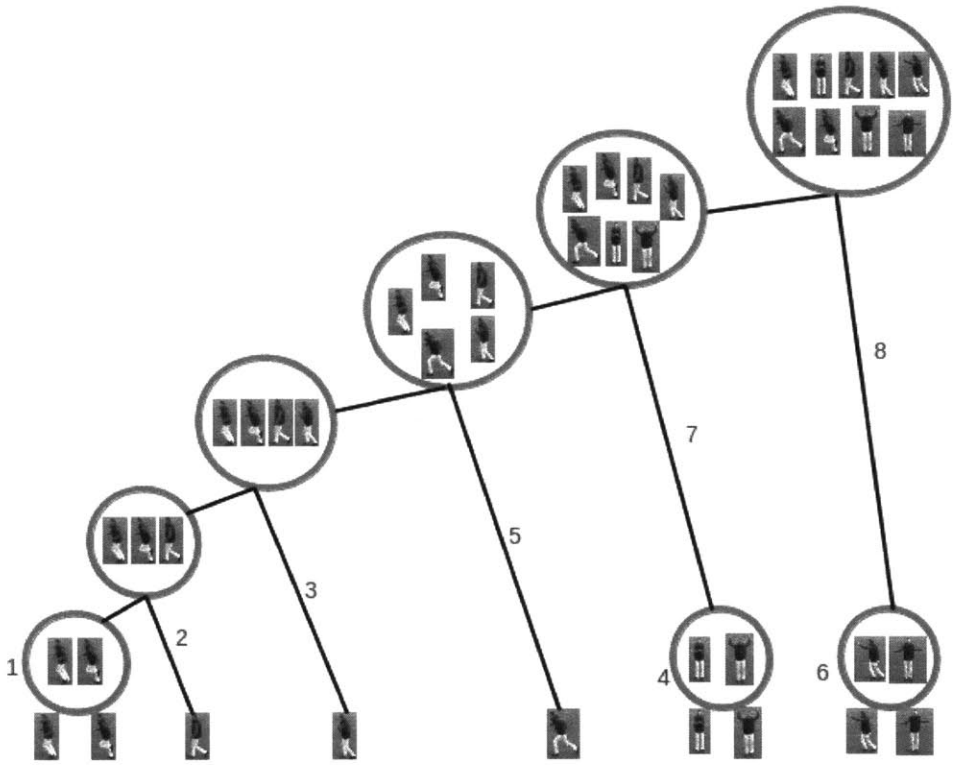

Figure 7-4: Hierarchical group tree generated in the training stage of pose classification on the KTH dataset using the approach 2

For this particular tree, the recall rate improves from 54% to 59% after the first grouping, and then to 67% after the second grouping. These groupings correspond to the reduction in number of poses from 9 to 7 poses. By reducing the pose set to 5 poses, the recall rate improves to 82%. This result demonstrates the hierarchical group tree representation's ability to guide the process of pose selection by selecting

the appropriate number of poses. It also introduces the possibility of building an automated pose selection mechanism which could be a very valuable extension of the work done in this thesis.

The reason that this broad pose selection occurs is because the poses were selected manually. What if a system could select the poses automatically? How would the system know which poses to select? I think that this problem requires a system in which a decision process flows both ways, top-down and bottom-up. The pose selection inherently depends on the problem that the system is trying to solve. For example, imagine that there are two sets of actions: {"running", "swimming", "diving"} and {"running", "jogging", "walking"}. In the first set, it is sufficient to define only three poses to achieve a near perfect action recognition rate because the actions in the set are very visually distinct. Moreover, the three poses can be defined with a substantial amount of variation within a pose category. However, in the second set, more poses have to be defined and they have to be defined with less variation within a pose category because the actions are visually similar. Addition of the automatic pose selection to the system would make it more flexible by enabling the top-down communication during the discovery stage.

7.3 Model of Transitions between Poses Results

In this section, I present the improvements to the pose classification results made by the application of hidden Markov model to the original pose sequence. The same frames that were used to generate the results for pose classification were used again, after the HMM has been applied to the sequence. The results, similar to the previous section, are presented separately for the two datasets in appendix in tables A.3 and A.4.

In the SHA dataset, the average recall rate improved from 69% to 89%, from 77% to 89% and from 61% to 80% for approaches 1, 2 and 3, respectively, while improving the false positive rate from 2.7% to 1.0%, from 2.0% to 1.1% and from 3.9% to 2.4%. That represents an average improvement of 17% in the recall rate across the

three approaches. The comparison of the complete results for the SHA dataset, obtained through independent pose classification and pose classification after the HMM correction is applied, is shown on the graphs in figure 7-5.

For the KTH dataset the average recall rate improved from 54% to 69% and from 68% to 82% for approaches 2 and 3, respectively, while improving the false positive rate from 5.2% to 3.6% and from 3.4% to 1.9%. That represents an average improvement of 14.5% in the recall rate across the two approaches. The comparison of the complete results for the KTH dataset, obtained through independent pose classification and pose classification after the HMM correction is applied, is shown on the graphs in figure 7-6.

In the figures above we can see that some poses' recall rates improve more than the others. For example, if we look at poses 2 and 9 in approach 2 in figure 7-5, we see that pose 2 recall rate was improved much more than pose 9 recall rate. The independent, single-frame pose classification recall rate is similar for the two poses. However, the effectiveness of temporal constraints model on them is very different. While pose 2 recall rate improves by more than 40%, the recall rate for pose 9 improves just over 5%. Why is that so? I think that in order to answer this question we need to look at the kind of mistakes that the independent single-frame classifier makes on both pose categories. In the case of pose 2, the classifier confuses it dominantly with pose 1 and sometimes with pose 5. In the case of pose 9, the classifier confuses it very often with pose 7, pose 5 and pose 1, and it sometimes makes mistakes and estimates it as a few other poses. The distribution of mistakes on the validation set directly influences the emissions probability distribution. This means that emissions probability distribution for pose 2 is very narrow (most probability is concentrated around poses 1 and 5), and for pose 9 it is spread out across many different poses. When a misclassification happens, it is much easier for the inference algorithm to infer the proper correction in the case when the mistake is common.

In this section, I have demonstrated that results of independent single-frame pose classification are significantly improved when the transitions between poses in the human motions are modelled with a hidden Markov model. As discussed in the end

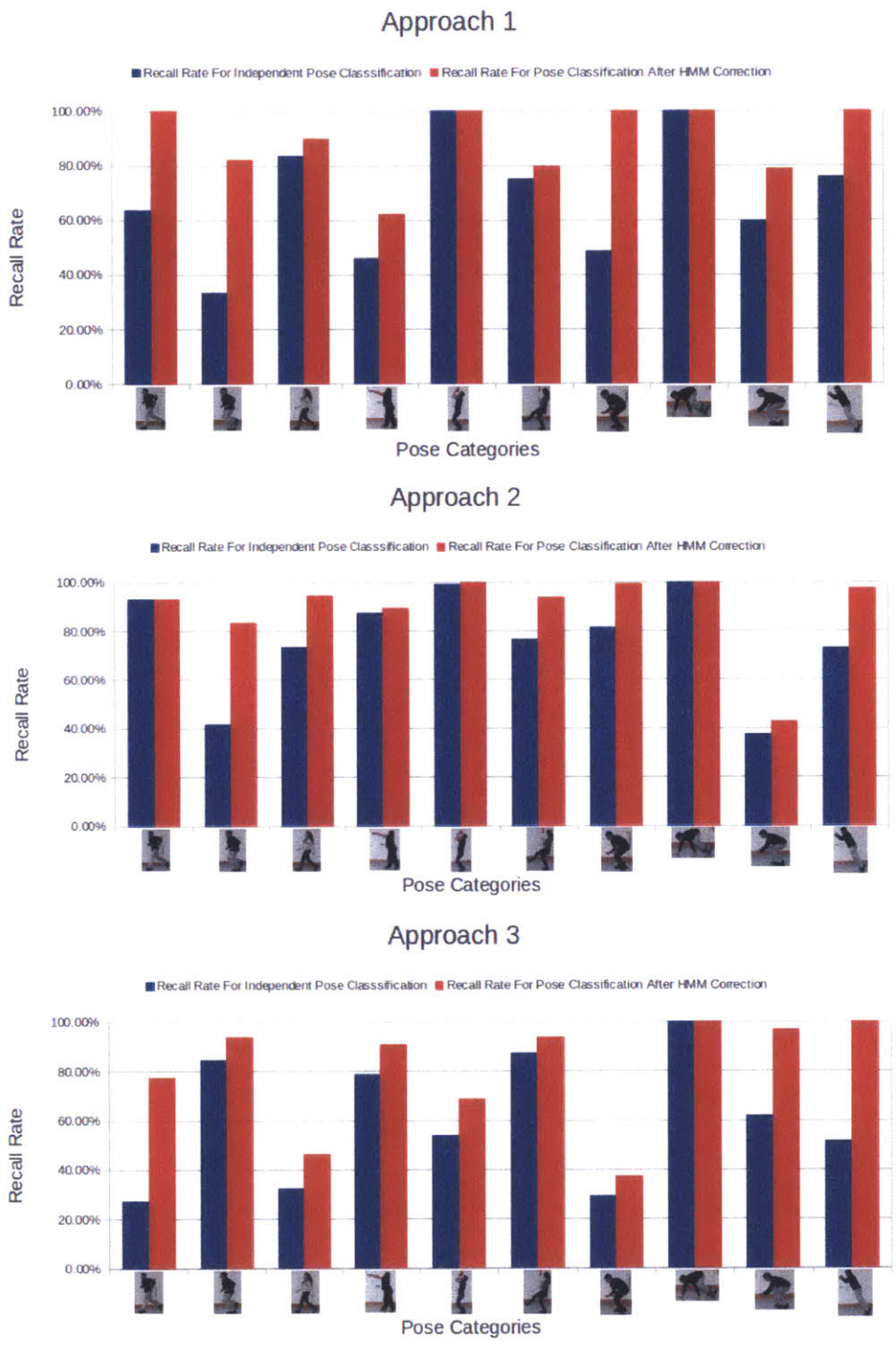

Figure 7-5: Comparison of independent single-frame pose classification and pose classification after HMM correction is applied for the SHA dataset

of chapter 4, this result has multiple implications. First, it directly affects action recognition by improving the pose sequence. Second, based on this result, I have

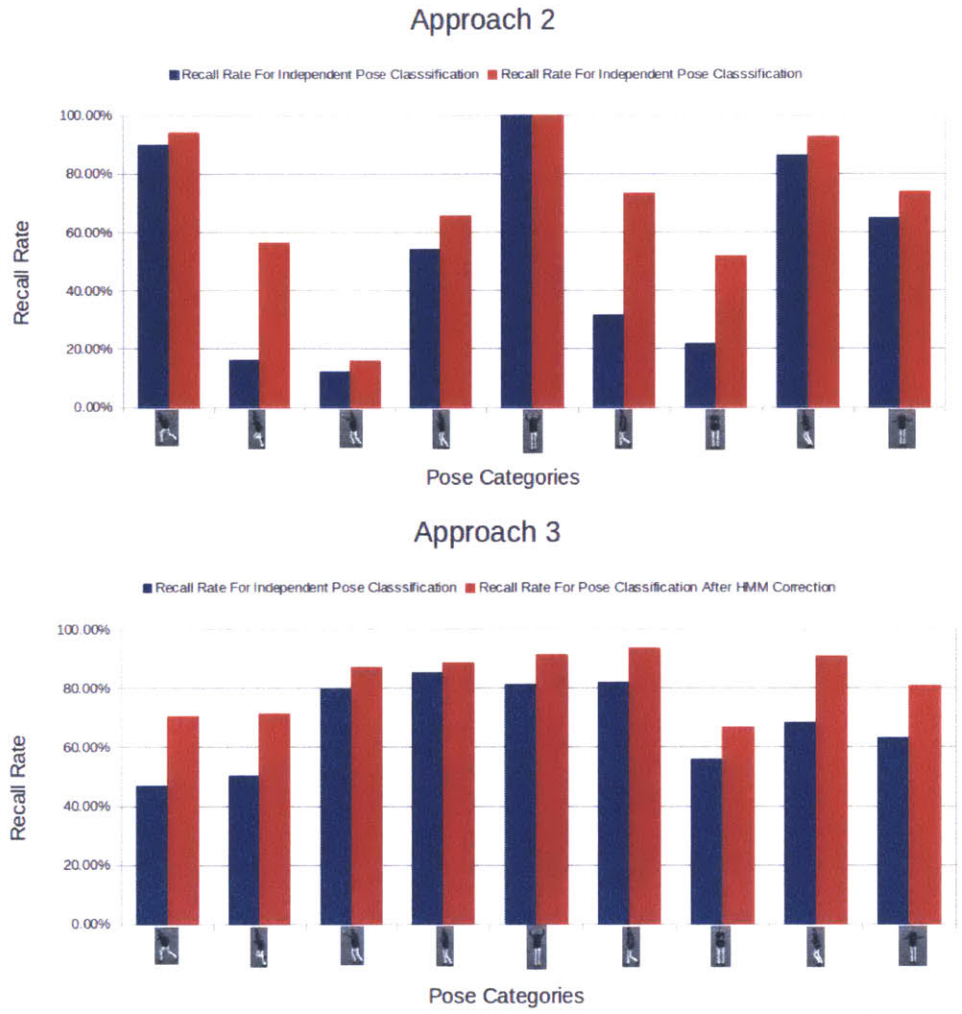

Figure 7-6: Comparison of independent single-frame pose classification and pose classification after HMM correction is applied for the KTH dataset

confirmed that structural constraints can be used as a powerful feedback mechanism to the pose classifier. The examples that are corrected are the ones that are currently present in the model of a pose and should not be there. Therefore, these examples can be used to prune the pose models and improve them online, during the runtime of the system.

7.4 Sequence Recognition Results

In this section I present the results of the action recognition based on a sequence of poses. I approached the problem of sequence recognition using two methods, an unsupervised method based on hierarchical agglomerative clustering and a supervised method based on training of HMM models for different action categories.

Unsupervised learning

In the unsupervised method, I apply agglomerative hierarchical clustering to the sequences of letters in the pose alphabet. At every level of the hierarchy, I measure the accuracy of the clustering method as the percentage number of actions that were put into a cluster where they are a majority.

$$accuracy = \frac{\sum_{a \in A} 1_{c(a)=a}}{|A|} \tag{7.1}$$

where A is a set of all actions, and $c(a)$ is the cluster action label into which action label a was put.

I measured the accuracy of the clustering algorithm for the sequences produced by three different methods and two different datasets. The graphs showing the accuracy obtained on the SHA dataset are shown in figure 7-7.

Similarly, the graphs showing the accuracy obtained on the KTH dataset are shown in figure 7-8.

The results of the unsupervised clustering prove that most of the sequence structure can be captured by a simple bag of words model. At number of clusters equal to the number of action categories (8 for SHA and 6 for KTH), the accuracy is 72% and 74% for SHA and KTH datasets, respectively. However, these figures also show that there is a limit to the amount of variability in the pose sequence that can be captured using the unsupervised learning. As the number of clusters grows, the accuracy increases, but seems to approach an asymptote.

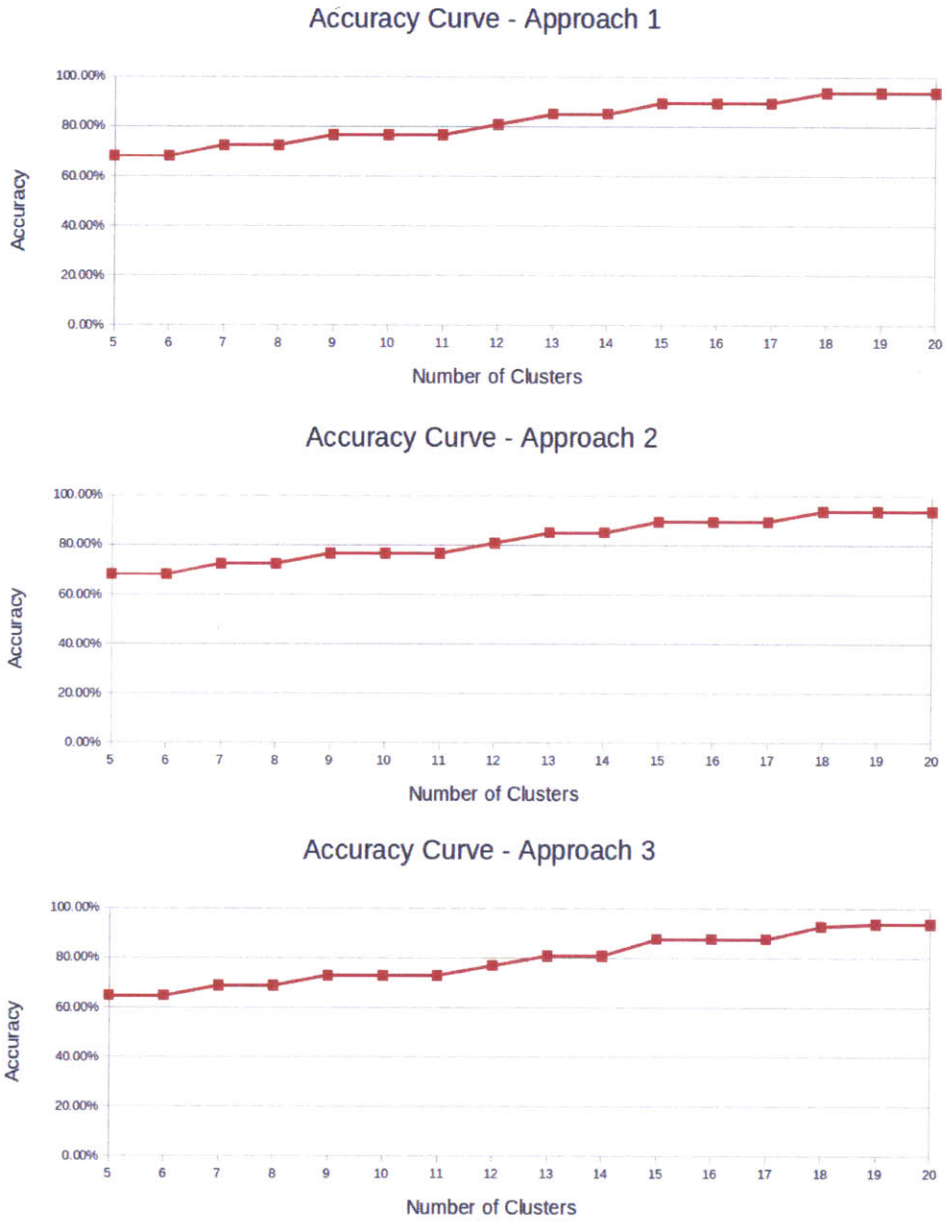

Figure 7-7: Clustering Accuracy for the SHA dataset

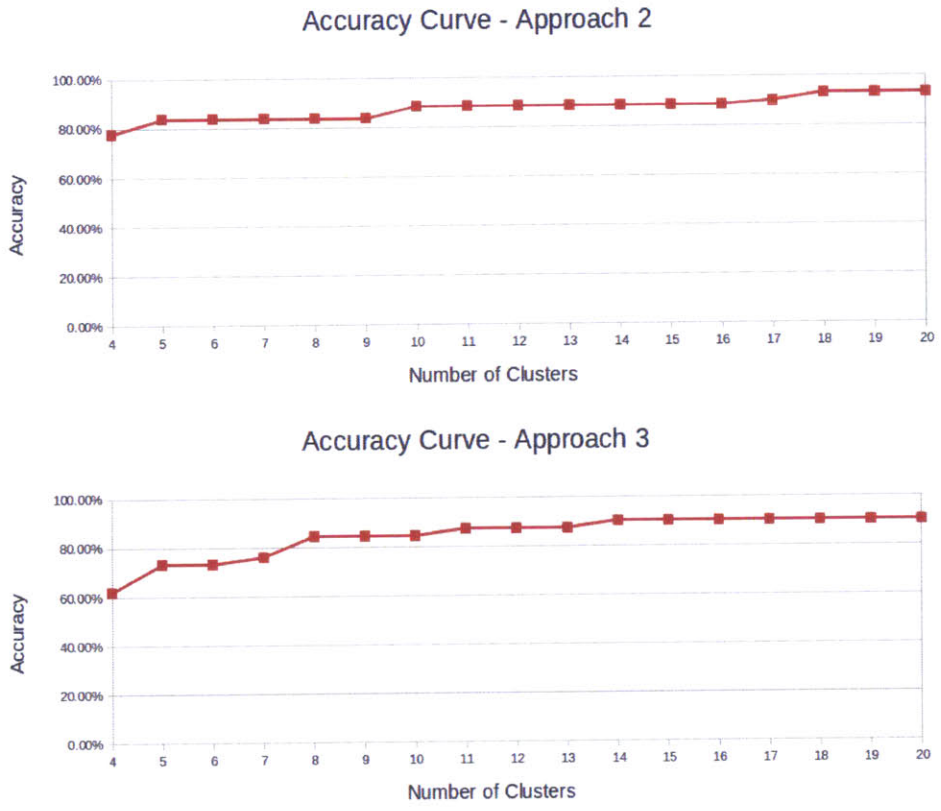

Figure 7-8: Clustering Accuracy for the KTH dataset

Supervised learning

In the supervised method, I model action sequences with a hidden Markov model, one per action category, using the Baum-Welch algorithm. Each model parameters are optimized such that they best describe the training sequences from the associated action category. I then test these models on the unseen sequences of actions by feeding the sequences and outputting the model that is most likely to produce such a sequence.

The sequences in the SHA dataset were tested in three different scenarios and sequences in the KTH dataset were tested in two different scenarios, corresponding to the previously described approaches in pose estimation. Table 7.2 summarizes the average results of the action recognition system on the KTH and SHA datasets.

	Approach 1		Approach 2		Approach 3	
	TP*	FP**	TP	FP	TP	FP
SHA	86%	0.9%	**91%**	**0.6%**	88%	0.8%
KTH	N/A	N/A	86%	1.1%	**91%**	**0.9%**

Table 7.2: This table shows the summary of the results in action recognition for SHA and KTH datasets and three different approaches; Note: TP*-true positives and FP**-false positives

Using approach 2 for pose classification, the system achieved 91% average recall rate on the SHA dataset containing 8 actions. It is interesting to notice that the system achieved a slightly higher action recognition recall rate when approach 3 was used (88%) for the pose classification instead of approach 1 (86%). This is interesting because independent single-frame pose classification with approach 1 achieved a higher recall rate (69%) compared to that with approach 3 (61%). The data on the pose classification suggests that this is simply because approach 3 was more successful in recognizing poses that turned out to be crucial in the action recognition stage. For example, one of the reasons that approach 3 pose classification recall rate was slightly lower is that the recall rate for a bending pose found in "crouching" was very often confused with the bending pose found in "jumping". However, when it comes to action recognition, the distinction between these two is not so important. This represents another example of a grouping effect that I mentioned earlier in the pose classifier section of the results. As mentioned before, hierarchical group tree representation can be a very useful mechanism for deciding which poses are key poses for action recognition.

Results for all the actions in the SHA dataset are summarized in figure 7-9.

In figure 7-9, we can see that actions running, walking and jumping have a consistently lower recall rate than others. The reason for this is that the poses in these actions are very often confused. That sometimes leads to model selecting a wrong action, for example walking instead of running. Sometimes the training sequences contain a mix of the poses that participate in all three actions which leads to a weak separation between the models. In that case, even sequences which can be clearly classified

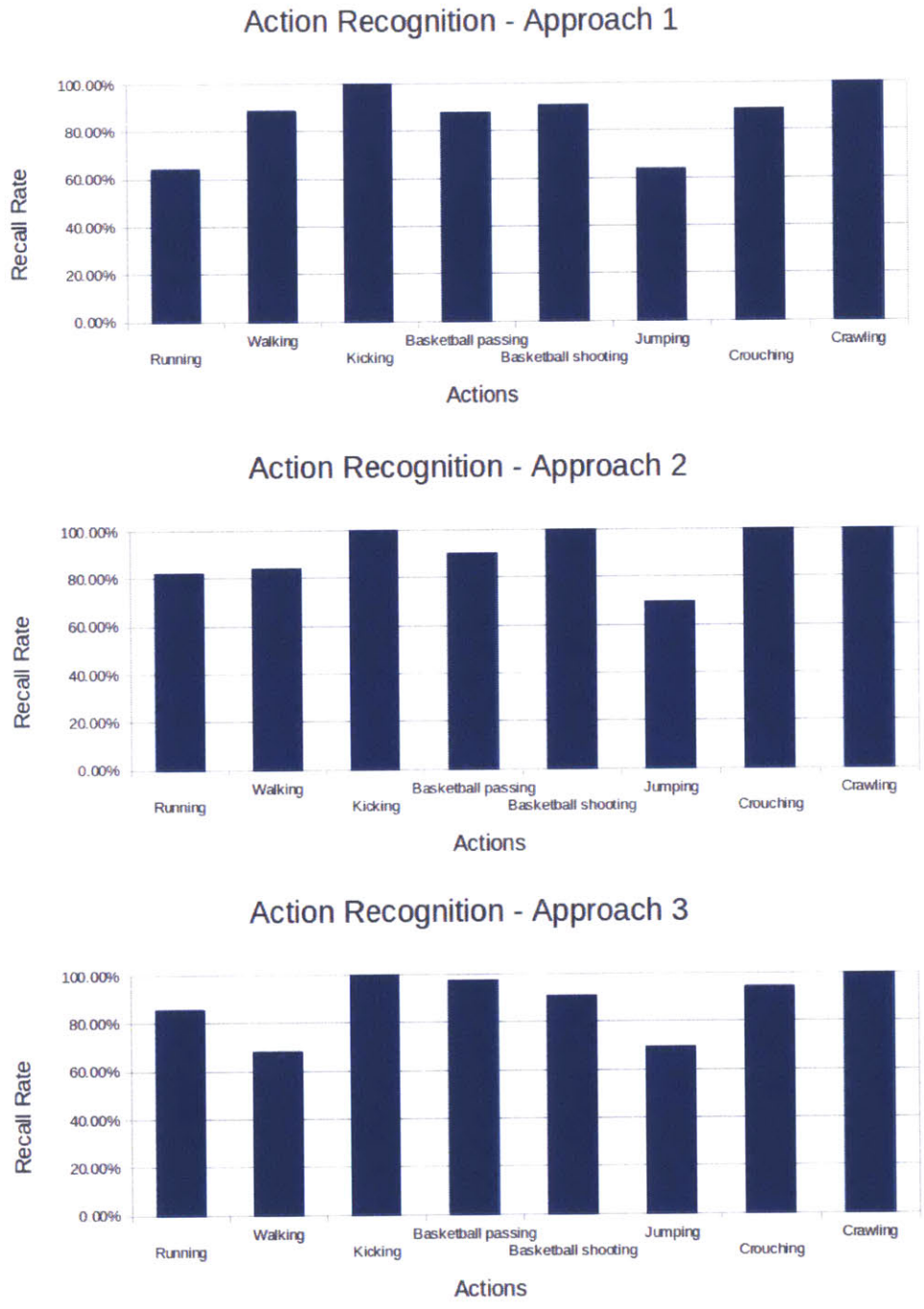

Figure 7-9: Action Recognition Recall Rate for the SHA dataset

correctly are misclassified. This suggests that there is a room for improvement in the detection rate that can be achieved with a more careful selection of training sequences.

Using approach 3 for pose classification, the system achieved 91% average recall rate on the KTH dataset containing 6 actions. This result is competitive with the state of the art approaches described in chapter 2, which achieve between 90% and 96% recall rate.

Results for all the actions in the KTH dataset are summarized in figure 7-10.

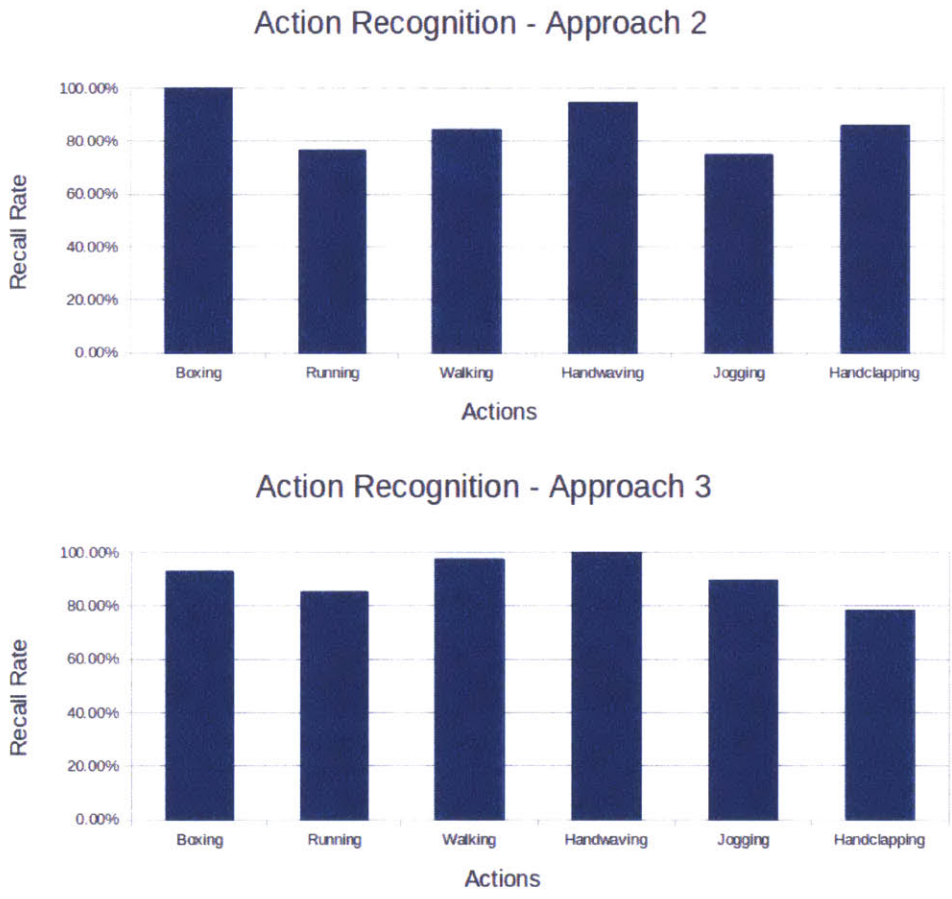

Figure 7-10: Action Recognition Recall Rate for the KTH dataset

Summary

In this chapter I presented results for three major components of my work, the single-frame pose classification system, the system for modelling the transitions between the poses and the action recognition system. I tested the systems on two datasets, the SHA dataset and the public KTH dataset.

The action recognition system as a whole achieves results that are competitive with the state of the art systems. I also showed that modelling transitions between poses based on the constraints in human motion significantly improves the results of single-frame pose classification. Finally, I showed that treating actions as a sequence of poses is an effective method in action recognition.

Chapter 8

Conclusion

In this final chapter I review the goals of this thesis. I state what has been done toward achieving these goals and outline some steps for the future work.

The goal of action recognition is to achieve a state where everyday activities can easily be described by an automated process. In order to build systems that can give you a live, play-by-play commentary of a sports game or detect security threats, we first need to understand how to recognize actions performed by humans on an everyday basis. In this thesis, I took a step toward achieving this goal by focusing on the problem of recognizing simple human actions. I approached this problem in three stages: single-frame pose classification, modelling of structural constraints in human motion and action recognition from the sequence of letters in pose alphabet. I demonstrated high-reliability action recognition results on two datasets. The results that I obtained are competitive with the results in other, state of the art approaches presented in chapter 2.

Furthermore, my approach is unique in the following key aspects:

1. I develop a 3-tier structure that combines the constraints of single-frame appearance on one level and the temporal constraints imposed by physics, human body physics in particular, on two higher levels. My approach enables relative strength on one level to compensate for relative weakness on another and provides a foundation for cooperative learning in future work.

2. I model actions as sequences of tentatively identified poses, and I think of pose sequences as having a story like quality, with one such sequence telling a walking story and another, a jumping story. This point of view encouraged unusual emphasis on single-frame processing in my work and provides a foundation for dealing with more complex actions and actions achieved in multiple ways, such as drinking, in future work.

8.1 Future work

Online training of the pose classifier system can gradually improve the results. One of the most challenging tasks in training a pose classification system is feature selection. This is typically done offline: a set of positive and negative examples is created and features that are the most descriptive are picked. However, very often we end up picking examples that do not allow the training process to capture the essential features of a certain class. This is because very often we do not know what kind of examples are good to pick. So it would be ideal if we can build a system that can automatically pick these examples and grow our training set. This is possible only in the case when there are two independent decision processes that can make an inference about a class. This is the case with the pose class estimation. On one side, there is a model of a pose built in the pose classifier. On the other side, there is a model of transitions between those poses. As I demonstrated in the results section, the model of transitions corrects a significant number of errors. However, those errors are still part of the pose classifier model. Now this error example can be used as a near miss example in order to prune the pose classifier model. Eventually when the model is pruned and the classifier runs into a similar example again, it will classify it correctly. Then the accuracy of the classifier will gradually improve which will lead to more robust action recognition.

Automated pose selection would be a great extension to the system presented in this thesis because it would allow top-down expansion to many other actions.

Currently, the pose selection is done manually which often leads to selecting pose categories that are too similar or not selecting an important pose category. Hierarchical group tree system represents a good starting ground for development of the system that would automatically select poses because the tree provides a similarity metric between different groups of poses. The next step would be to feed the system with many different actions and optimize the parameters of the system based on the group tree such that it picks the poses that best describe actions in the system.

8.2 Contributions

In this thesis, I make four distinct contributions:

1. First, I successfully integrated Ullman's Intermediate Features principle into the pose classification system. On top of it I developed a hierarchical group tree structure which serves as a representation of similarity between different pose groups.

2. I introduced the model of regularities in pose sequences. I showed that it significantly improves the results of single-frame pose classification and enables reliable pose correction. On the dataset with 10 poses, the model improved the results by an average of 17% and on the dataset containing 9 poses, it improved the results by 14.5%.

3. I developed a model of actions that learns simple human actions based on the sequences of poses. Using the Baum-Welch algorithm, the model learns the parameters of a hidden Markov model for a specific action category. The model is then used for action recognition.

4. I demonstrated the strength of the overall system by successfully learning and recognizing actions in the 6-action public KTH dataset and the 8-action newly introduced dataset. On both datasets, the system achieved 91% action recognition recall rate. These results demonstrate that simple actions can be effectively represented using sequence of poses.

Appendix A

Tables

	Approach 1		Approach 2		Approach 3	
	TP*	FP**	TP	FP	TP	FP
Pose1	0.64	0.019	0.93	0.052	0.27	0.020
Pose2	0.33	0.008	0.42	0.003	0.84	0.054
Pose3	0.83	0.019	0.73	0.012	0.32	0.003
Pose4	0.46	0.002	0.87	0.004	0.79	0.047
Pose5	1.00	0.072	0.99	0.077	0.54	0.094
Pose6	0.75	0.002	0.77	0.002	0.87	0.020
Pose7	0.49	0.000	0.81	0.024	0.29	0.027
Pose8	1.00	0.065	1.00	0.003	1.00	0.028
Pose9	0.60	0.077	0.38	0.023	0.62	0.080
Pose10	0.76	0.009	0.73	0.007	0.51	0.023

Table A.1: This table shows complete results of the independent single-frame pose classification on the SHA dataset; Note TP*-true positives and FP**-false positives

	Approach 2		Approach 3	
	TP*	FP**	TP	FP
Pose1	0.90	0.111	0.47	0.016
Pose2	0.16	0.038	0.50	0.025
Pose3	0.12	0.007	0.80	0.006
Pose4	0.54	0.134	0.85	0.066
Pose5	1.00	0.041	0.81	0.046
Pose6	0.31	0.083	0.82	0.081
Pose7	0.22	0.004	0.56	0.028
Pose8	0.86	0.019	0.68	0.035
Pose9	0.65	0.037	0.63	0.002

Table A.2: This table shows complete results of the independent single-frame pose classification on the KTH dataset; Note TP*-true positives and FP**-false positives

	Approach 1		Approach 2		Approach 3	
	TP*	FP**	TP	FP	TP	FP
Pose1	1.00	0.009	0.93	0.029	0.77	0.008
Pose2	0.83	0.003	0.83	0.012	0.94	0.036
Pose3	0.90	0.011	0.95	0.008	0.46	0.000
Pose4	0.62	0.000	0.89	0.003	0.91	0.050
Pose5	1.00	0.030	1.00	0.028	0.69	0.036
Pose6	0.80	0.001	0.94	0.002	0.94	0.014
Pose7	1.00	0.000	0.99	0.026	0.37	0.000
Pose8	1.00	0.046	1.00	0.000	1.00	0.026
Pose9	0.79	0.000	0.43	0.002	0.97	0.069
Pose10	1.00	0.007	0.97	0.005	1.00	0.003

Table A.3: This table shows complete results of the pose classification, on the SHA dataset, after the regularities in pose sequences have been modelled by an HMM.; Note TP*-true positives and FP**-false positives

	Approach 2		Approach 3	
	TP*	FP**	TP	FP
Pose1	0.94	0.069	0.70	0.005
Pose2	0.56	0.058	0.71	0.012
Pose3	0.16	0.001	0.87	0.002
Pose4	0.66	0.040	0.89	0.040
Pose5	1.00	0.023	0.91	0.029
Pose6	0.73	0.089	0.93	0.048
Pose7	0.52	0.009	0.67	0.015
Pose8	0.93	0.010	0.91	0.021
Pose9	0.74	0.026	0.81	0.000

Table A.4: This table shows complete results of the pose classification, on the KTH dataset, after the regularities in pose sequences have been modelled by an HMM.; Note TP*-true positives and FP**-false positives

Bibliography

JK Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Matteo Bregonzio, Shaogang Gong, and Tao Xiang. Recognising action as clouds of space-time interest points. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1948–1955. IEEE, 2009.

Thomas M Cover, Joy A Thomas, and John Kieffer. Elements of information theory. *SIAM Review*, 36(3):509–510, 1994.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Boris Epshtein and S Uliman. Feature hierarchies for object classification. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 220–227. IEEE, 2005.

Boris Epshtein and Shimon Ullman. Semantic hierarchies for recognizing objects and parts. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Alexander Klaser and Marcin Marszalek. A spatio-temporal descriptor based on 3d-gradients. 2008.

Li Liu, Ling Shao, and Peter Rockett. Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition. *Pattern Recognition*, 2012.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Fengjun Lv and Ramakant Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

Shimon Ullman, Michel Vidal-Naquet, and Erez Sali. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, 2002.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, pages 1–21. ISSN 0920-5691. URL `http://dx.doi.org/10.1007/s11263-012-0564-1`. 10.1007/s11263-012-0564-1.

Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

Daniel Weinland and Edmond Boyer. Action recognition using exemplar-based embedding. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE, 2008.

Patrick H Winston. Learning structural descriptions from examples. Technical report, DTIC Document, 1970.

Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE, 1992.