Doctoral Thesis

Vision-Based Human Motion Analysis

Author(s):
Yao, Angela

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007563521

Rights / License:
In Copyright - Non-Commercial Use Permitted
Vision-Based Human Motion Analysis

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences (Dr. sc. ETH Zürich)

presented by
Yingjie Angela Yao
MSc ETH in Biomedical Engineering
born February 18, 1983, citizen of Canada

accepted on the recommendation of
Prof. Dr. Luc Van Gool, ETH Zürich and KU Leuven, examiner
Dr. Cordelia Schmid, Inria Grenoble, co-examiner

2012
Abstract

Interpreting human activity from video is at the core of a wide spectrum of applications such as content-based indexing, intelligent surveillance, human-computer interfacing and sports video analysis. Cheap hardware and growing storage capacity has led to an explosion of video data and there is a critical need for machine vision algorithms that automatically analyze video content.

This thesis provides a collection of methods for video-based human action recognition, i.e. the application of semantic labels to a person’s movements over time in a video sequence. We present two approaches for this task, one appearance-based and one pose-based. The appearance-based method uses no structural modeling of the human body and relies only on the statistical distribution of appearance features such as edges, shapes and flow to classify actions. The pose-based method, on the other hand, explicitly estimates a 3D articulated pose of the body and classifies the action based on geometric relations between specific joints in a single pose or a short sequence of poses.

In addition to determining action labels, we examine how action recognition could be leveraged to help with the closely related task of human pose estimation. We integrate action recognition and pose estimation into a single system, taking output from appearance-based action recognition as a prior for 3D pose estimation. The estimated poses are then used to for pose-based action recognition to refine the action label. Finally, we examine the temporal aspect of labeling actions and propose a method to both segment and classify actions from a continuous stream of body poses.
Zusammenfassung


Da die Schätzung der menschlichen Pose und die Erkennung der menschlichen Tätigkeit eng miteinander verbunden sind, untersuchen wir, inwieweit das Lösen der einen Aufgabe hilfreich für die andere ist. Hierzu vereinen wir Lösungsansätze für beide Probleme in ein einziges System und verwenden die Resultate der bildbasierten Methode zur Tätigkeitserkennung als apriorisches Wissen für die Poseschätzung. Die hiermit ermittelten Posen verwenden wir wiederum, um die annotierten Tätigkeiten mit Hilfe des posenbasierten Ansatzes zu verbessern. Zum Schluss stellen wir eine Methode vor, die nicht nur die Tätigkeiten klassifiziert, sondern auch die Länge der Tätigkeit genau segmentiert.
Acknowledgements

This thesis would not have been possible without the support of a number of people and I would like to extend my sincerest gratitude to each one.

First and foremost, I would like to thank Professor Luc Van Gool for giving me the opportunity to be a part of the Computer Vision Lab at ETH Zurich and the freedom to explore my own research interests. My thanks also go to Dr. Juergen Gall, who has shaped the way that I approach research and has taught me, by example, how to be a proficient and prolific researcher.

A portion of this thesis was the result of a collaboration with Professor Raquel Urtasun. Many thanks for hosting me at the Toyota Technology Institute at Chicago and for the numerous Skype conferences about any and everything.

To all the members of BIWI, past and present, and especially to the girls of my office, thank you for making me feel so welcome these last four years and for creating such a friendly atmosphere in the lab.

I am fortunate enough to have supervised several students during my time at BIWI. They deserve much credit and have taught me how to mentor and be a supervisor. Particular thanks goes to Dominique Uebersax and Daniel Waltisberg, whose hard work became integral parts of my own research.

Work on this thesis was funded by the EU project IURO, the Swiss National Foundation NCCR project IM2 and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Lastly, I would like to thank my family and friends. My parents’ love, support and encouragement along this long path of education has made me the person that I am today. And to Dani, thanks for persevering with me, for helping me keep things in perspective and for loving me.
# Contents

1 Introduction  
1.1 Terminology ................................................. 4  
1.2 Contributions .................................................. 6  
1.3 Organization .................................................... 6  

2 Related Works  
2.1 Introduction .................................................... 9  
2.2 Actions as Sequences of Poses ................................. 9  
2.3 Actions as Space-Time Volumes ............................... 10  
2.4 Actions as Collections of Local Spatio-Temporal Features .. 11  
2.5 Actions as a collections of pose-derived features .......... 12  
2.6 Actions With Context ......................................... 12  
2.7 Appearance- versus pose-based action recognition .......... 12  

3 Appearance-based Action Recognition  
3.1 Introduction .................................................... 15  
3.2 Related Works ................................................. 18  
3.3 Voting Framework for Action Recognition .................... 19  
3.3.1 Training ....................................................... 19  
3.3.2 Classifying and Localizing Actions ......................... 21  
3.3.3 Building Action Tracks ..................................... 22  
3.4 Experiments ..................................................... 23  
3.4.1 Datasets ....................................................... 23  
3.4.2 Evaluation Measures ........................................ 23  
3.5 Results ......................................................... 24  
3.5.1 Benchmarks: Weizmann and KTH ............................. 24  
3.5.2 Broadcast Sports: UCF Sports ............................... 25  
3.5.3 Surveillance: UCR Videoweb Activities .................... 27  
3.5.4 Aerial Footage: UT Tower Dataset ......................... 28  
3.5.5 Inhouse Monitoring: TUM Kitchen Dataset ................. 28  
3.5.6 Dense patch sampling ........................................ 32  
3.6 Discussion and Outlook ....................................... 32
4 Leveraging Actions for Pose Estimation

4.1 Introduction ................................................. 33
4.2 Related Work .............................................. 34
4.3 Framework .................................................. 35
4.4 Action Recognition ......................................... 35
4.5 Optimizing Over a Set of Manifolds ...................... 37
  4.5.1 Action-Specific Manifolds ............................... 38
  4.5.2 Theoretical Discussion ................................. 39
  4.5.3 Algorithm ............................................... 39
4.6 Experiments ................................................ 43
  4.6.1 Datasets ............................................... 43
  4.6.2 Action Recognition ..................................... 44
  4.6.3 3D Pose Estimation ................................... 46
4.7 Conclusion .................................................. 53

5 Pose Estimation of Complex Activities .................... 57

5.1 Introduction ................................................ 57
5.2 Stochastic learning ....................................... 59
  5.2.1 GPLVM Review ........................................ 59
  5.2.2 Stochastic Gradient Descent ......................... 60
  5.2.3 Incremental Learning .................................. 62
  5.2.4 Tracking Framework .................................... 63
5.3 Experimental Evaluation .................................. 64
  5.3.1 Neighbourhood Type and Size ....................... 64
  5.3.2 Monocular Tracking .................................... 66
  5.3.3 Online Tracking ....................................... 67
  5.3.4 Multi-view Tracking on HumanEva .................. 67
5.4 Conclusion and Future Work .............................. 70

6 Pose-based Action Recognition .............................. 73

6.1 Introduction ................................................ 73
6.2 Related Work .............................................. 74
6.3 Methods ..................................................... 75
  6.3.1 Appearance Features .................................. 76
  6.3.2 Pose Features ......................................... 76
  6.3.3 Combined Features .................................... 79
  6.3.4 Dataset and Experimentation ......................... 79
6.4 Results ..................................................... 80
  6.4.1 Appearance-based Features ......................... 80
  6.4.2 Pose-based Features .................................. 80
  6.4.3 Combined Appearance- and Pose-Based Features .... 81
6.5 Conclusion ................................................ 84
7 Integrated Action Recognition and Pose Estimation ........................................ 85
   7.1 Introduction ......................................................................................... 85
   7.2 Pose-based action recognition ............................................................ 87
   7.3 Closing the Loop ................................................................................ 91
   7.4 Conclusion ......................................................................................... 91

8 Online Action Segmentation and Classification .............................................. 93
   8.1 Introduction ......................................................................................... 93
   8.2 Related Works .................................................................................... 94
   8.3 Approach ............................................................................................. 95
      8.3.1 Pose and Action Representation ................................................... 95
      8.3.2 Comparing Two Actions .............................................................. 96
      8.3.3 Vocabulary of Actions .................................................................. 97
      8.3.4 Online Processing ....................................................................... 98
   8.4 Experiments ......................................................................................... 99
      8.4.1 Datasets ....................................................................................... 99
      8.4.2 Classification ............................................................................... 101
      8.4.3 Segmentation and Classification ................................................... 102
   8.5 Conclusion ......................................................................................... 104

9 Conclusion .................................................................................................. 107
   9.1 Contributions .................................................................................... 107
   9.2 Perspectives ....................................................................................... 108

A Tracking People in Broadcast Sports ............................................................. 113
   A.1 Introduction ....................................................................................... 113
   A.2 Related Works ................................................................................... 115
   A.3 Sports Tracker ................................................................................... 115
      A.3.1 Tracking Overview ...................................................................... 116
      A.3.2 Vote-Based Confidence Map ...................................................... 116
      A.3.3 Appearance Model ..................................................................... 117
      A.3.4 Dynamical Model ...................................................................... 117
   A.4 Experiments ....................................................................................... 118
      A.4.1 Datasets ..................................................................................... 118
      A.4.2 Evaluation .................................................................................. 118
   A.5 Results ............................................................................................... 119
   A.6 Conclusion ....................................................................................... 122

B Facial Expression Recognition ........................................................................ 123
   B.1 Introduction ....................................................................................... 123
   B.2 Related work ..................................................................................... 124
   B.3 Voting Framework for Facial Expression Recognition ......................... 125
      B.3.1 Training ...................................................................................... 126
### B.3.2 Facial Expression Classification 127

### B.4 Building the Expression Tracks 129
  **B.4.1 Feature Extraction** 129

### B.5 Experiments 130

### B.6 Conclusions 134

### C Group Action Recognition 135

#### C.1 Introduction 135

#### C.2 Hough-Voting Framework 135
  **C.2.1 Training** 136
  **C.2.2 Classifying and Localizing Actions** 136

#### C.3 Combining Classifiers for Group Action Recognition 137
  **C.3.1 Decision Profiles** 137
  **C.3.2 Extension for Asymmetric Group Actions** 138

#### C.4 Experiments 138
  **C.4.1 Single Person Actions** 139
  **C.4.2 Group Interactions** 139

#### C.5 Discussion 139
Introduction

Understanding human activity from video is a central problem in computer vision. Movements of the body can be interpreted on a physical level through pose estimation, i.e. localizing the individual’s body parts, or on a higher semantic level through action recognition, i.e. understanding the movements over time.

Progress in the field has been driven by a multitude of potential applications which rely on interpreting human motions:

Content-Based Video Indexing Cheap consumer hardware to produce video along with fast internet access and growing storage capacity has led to an explosion of video data easily accessible on the web. At the time of writing, YouTube has 48 hours of worth video being upload every minute, equivalent to almost 8 years worth of content every day, while 3 billion videos are being watched every day\(^1\). Automated indexing, search and retrieval that does not rely on manual annotation such as user-provided descriptions and tags is in high demand.

Intelligent Surveillance We are living in a "surveillance society" - closed-circuit television cameras are constantly being added to urban, suburban and even rural areas throughout the world. Despite all of the recording activity, analysis of the footage still lags behind and is heavily dependent on manual effort. The sheer number of cameras makes it impossible for human operators to watch or survey everything, hence the need for machine vision algorithms to detect abnormal or suspicious activity. Such algorithms are also highly applicable for home monitoring to support geriatric independent living.

Human-Computer Interfacing Touch free gesture-based controllers are no longer a thing of science fiction movies\(^2\) but a reality that is easily accessible to the general public, particularly with the release of the Microsoft Kinect. As more and more computer systems are being embedded into our ambient environment, the ability to interact with these systems without external equipment will become very popular.

\(^1\)http://www.youtube.com/t/press_statistics, retrieved 03.02.2012

\(^2\)Recall the famous scene of Tom Cruise manipulating computer screens with his fingers.
Sports Video Analysis  Athletes and coaches have long used video footage as a training tool, not only for analyzing individual and team performance but also to develop strategy against opponent teams. Vision algorithms can automate many tasks such as tracking the athletes, annotation, statistical tabulation of plays and generation of semantic descriptions.

Motion Capture and Animation  Recording the movements of actors and recreating computerized models has made animated characters come to life on the big screen with impressive visual effects. The film and entertainment industry has pushed development on this front though motion capture is also widely used in video game development, virtual reality applications, and biomechanics and gait analysis.

Despite the extensive research activity in the field in the past decade and a half, efforts to interpret human motion in video are still being challenged by the complexity and variability of real-world footage. Variation in viewpoint and scale, occlusions and background clutter, as well as variation in people’s size, appearance, speed and style of movement are some examples of why the problem remains difficult. The aim of this thesis is to provide a collection of methods which can overcome some of these challenges to analyze and understand human motion.

The primary focus of this thesis is on action recognition from video, i.e. applying a semantic label to a person’s movements over time in a video sequence. In the current thesis, we differentiate between two settings for action recognition, that of unconstrained video, where sequences are drawn from monocular footage taken from television, films and or YouTube, usually with moving backgrounds, and that of constrained video, in which (multiple) cameras are mounted in fixed locations in a studio setting. Unconstrained video is a more challenging form of data and action labels are more geared towards content-based indexing applications. Constrained video offers the opportunity to study human actions on a more detailed level and is more applicable for surveillance or human-computer interfacing.

Note that the task of action recognition can share significant information overlap with the task of pose estimation. On one hand, pose information can be a very strong indicator of actions and on the other hand, action labels can be determined from as little as a single pose. As such, a secondary focus of the thesis is on human pose estimation, particularly where action recognition and pose estimation overlap.

1.1 Terminology

The use of terminology in the vision community for action recognition is not always consistent; action recognition is often used interchangeably with activity recognition. In
A hierarchy was proposed of action primitives, actions, and activities. Action primitives are atomic entities out of which actions are built, which in turn constitute an activity. Action primitives are generally limited to the movements of specific limbs, while actions should have some semantic meaning attached. Activities are a higher level combination of actions, usually with some temporal relationships attached to the individual actions. For example, playing soccer could be considered an activity which could be decomposed into actions such as kicking or running. Relevant action primitives would then be movements such as moving the right leg forward or swinging the left arm back. In this thesis, we use the provided class labels in publicly available datasets, which unfortunately contain a mix between all three levels of the Moeslund’s hierarchy. For the most part, we classify actions, such as walking, running, and waving, as per the benchmark datasets KTH [Schuldt et al., 2004] and Weizmann [Blank et al., 2005], though there are also some activities, such as boxing in KTH, diving in UCF Sports [Rodriguez et al., 2008] and some primitives, such as reach and release grasp in the TUM Kitchen Dataset [Tenorth et al., 2009].

In most cases, the recognition in the action recognition problem is used to refer only to the classification aspect, i.e. determining the semantic label. Closely related is action detection, i.e. determining the semantic label as well as the spatial and or temporal extent of an action. Detection is typically considered more challenging than classification, although the two problems are not always distinctly separable, with results of one closely relying on another. This is the case for the methods developed in this thesis and we will use the term recognition as an umbrella term to refer to both classification and detection.

Recovering people’s body configurations from video is referred to both as (articulated) pose estimation and articulated tracking, often interchangeably. Strictly speaking, pose estimation is a more general problem, since articulated tracking can be cast as a pose estimation problem over multiple frames. However, pose estimation is typically regarded as a more challenging problem and leveraging the coherency that exists in video data through tracking can help simplify the problem in the video domain. In this thesis, we primarily use the term pose estimation, even though tracking plays a critical role in determining pose.
1.2 Contributions

Specific contributions of this thesis are listed as follows:

1. We present an action recognition system using a Hough-transform based voting framework which can handle both appearance-based and pose-based features and is applicable under a wide range of action recognition scenarios.

2. Using the same framework of contribution 1, we perform a comparison of low-level appearance-based features such as edges, shapes and flow and high-level pose-based features which rely on structural modeling of the human body for the application of human action recognition.

3. We demonstrate how action recognition can help with pose estimation task and propose a system which can integrate the action recognition results from the system in contribution 1 as a prior distribution for pose estimation. We show the effectiveness of including such a prior, especially under limited computational resources.

4. We demonstrate the feasibility of using estimated poses for pose-based action recognition. The quality of poses estimated from the system in contribution 3, with an average error of 42-70mm, is sufficient for reliable action recognition.

5. We present an integrated framework which couples action recognition and pose estimation by combining the systems in contributions 1 and 3. The framework integrates an appearance-based action recognition system as a prior for pose estimation and then refines the action labels using pose-based action recognition based on the extracted poses.

6. We present an online method for simultaneous classification and temporal segmentation of actions. Motivated by the release and success of the Microsoft Kinect, this method was designed especially with such a system in mind, in which an incoming stream of poses are available.

7. We demonstrate an alternative method for learning low-dimensional manifolds to encode human body poses which could be applied to pose estimation and tracking.

1.3 Organization

This thesis is organized as follows:

In Chapter 2, Related Works, we provide a short literature review of vision-based action recognition to give the reader an overview of the field. Highly relevant works will be discussed in more detail in each of the individual chapters.
In Chapter 3, Appearance-based Action Recognition, we present a method to classify and localize human actions in video using a Hough transform voting framework. To facilitate the recognition, we first track the human using a detection-based particle-filter. Random trees are then trained to learn a mapping between densely-sampled feature patches and their corresponding votes in a spatio-temporal-action Hough space. Using low-level features such as gradients and optical flow, we demonstrate that Hough-voting can achieve state-of-the-art performance on several datasets covering a wide range of action-recognition scenarios. This chapter is based on research originally presented in [Yao et al., 2010a] and [Gall et al., 2011b].

In Chapter 4, Leveraging Actions for Pose Estimation, we present an algorithm that integrates the results of a 2D action recognition system as a prior distribution for particle-based optimization for 3D pose estimation. Low-dimensional manifolds are often used to simplify 3D pose estimation, but the complexity of the embeddings increases with the number of actions. Separate, action-specific manifolds seem to be more practical; here, we adapt a particle-based annealing optimization scheme [Gall et al., 2008b] to jointly optimize over the action-specific manifolds and the human poses embedded in each of the manifolds. The approach scales in the worst case linearly with the number of manifolds but can be made much more efficient with an action prior. Our experiments demonstrate that the optimization can handle an 84D search space and provides already competitive results on HumanEva with as few as 25 particles. This work was originally presented in [Gall et al., 2010b] and [Yao et al., 2012].

In Chapter 5, Pose Estimation of Complex Activities, we take an aside to discuss an alternative method for learning low-dimensional embeddings. Learned low-dimensional manifolds are commonly used to simplify the pose-estimation problem but can be poor at generalization; models which are more expressive are more difficult or inefficient to learn. In this chapter, we present an efficient stochastic gradient descent algorithm that is able to learn probabilistic non-linear latent spaces composed of multiple actions. We also derive an incremental algorithm for the online setting which can update the latent space without extensive relearning. We demonstrate the effectiveness of the manifold learning technique on the task of monocular and multi-view tracking. The work of this chapter was originally presented in [Yao et al., 2011b].

In Chapter 6, Pose-based Action Recognition, we present demonstrate the robustness of using relational pose features [Müller et al., 2005] for pose-based action recognition. In this chapter, we address the question of whether pose estimation is useful for action recognition or if it is better to train a classifier only on low-level appearance features drawn from video data. We compare pose-based, appearance-based and combined pose and appearance features for action recognition in a home-monitoring scenario. Our experiments show that pose-based features outperform low-level appearance features, even when heavily corrupted by noise, suggesting that pose estimation is beneficial for the action recognition task. The comparison of the different types of features was originally presented in [Yao et al., 2011a].
In Chapter 7, *Integrated Action Recognition and Pose Estimation*, we present an integrated framework, coupling together the individual systems presented in Chapters 3, 4 and 6. We transition from action labels to pose estimates and back to actions again. This work was originally presented in [Yao et al., 2012].

In Chapter 8, *Online Action Segmentation and Classification*, we explore a relatively neglected aspect of action recognition - temporal segmentation. Previous action recognition algorithms almost always assume that a temporal segmentation of input data is already available. We believe this to be unrealistic for most real-life applications and focus our attention on simultaneous action recognition and segmentation. Taking as input a continuous stream of body poses, we propose an efficient framework for online analysis. To study and evaluate the performance of our algorithm we have tested it on both markered motion capture data and less robust markerless pose estimates collected in real-time from an RGB-depth sensor.

Finally, we conclude the thesis in Chapter 9 by discussing the open problems and future research directions. For completeness, we have also included appendices which are extensions and modifications of the presented work in the thesis. In Appendix A, *Tracking People in Broadcast Sports*, we elaborate on the tracker discussed in Chapter 3 and include results from extra experiments; work here was originally presented in [Yao et al., 2010b]. In Appendix B, *Facial Expression Recognition*, and Appendix C, *Group Action Recognition*, we apply the appearance-based action recognition method of Chapter 3 to classify facial expressions and group actions respectively, as presented originally in [Fanelli et al., 2010] and [Waltisberg et al., 2010].

We have tried to make each chapter as independent as possible. Chapters 3, 4, 6 and 8 can be read independently, while Chapter 7 describes the framework which integrates the various systems presented in Chapters 3, 4 and 6.
2

Related Works

2.1 Introduction

Action recognition is a rich field of research supported by a huge corpus of literature. For a more comprehensive overview, the reader is referred to the survey papers [Turaga et al., 2008; Poppe, 2010; Aggarwal & Ryoo, 2011; Weinland et al., 2011]. Discussion of related works here focus predominantly on methods applicable to video and those which address general movements of the body. Closely related but beyond the scope of current work are the specialized fields of facial expression and gesture recognition.

Some of the earliest works in action recognition focused on tracking body parts and classifying the joint movements (Section 2.2). Rather than tracking the individual limbs, an alternative line of work has treated actions as space-time volumes occupied by the body (Section 2.3). More recently, focus has shifted towards direct classification of actions with abstract, low-level spatio-temporal features (Section 2.4).

In fact, the general trend of vision-based action recognition over time has been to decrease the amount of modeling (and thereby constraints) with respect to the human body. Low-level spatio-temporal features alone, however, offer little intuition with regards to the actor performing the action, much less the various poses that constitutes the action itself. In an attempt to bring back the “human” to human action recognition, a new line of work has tried to couple person-detectors with the action recognition task and focus on features which are related to the human pose even though the pose is not solved for explicitly (Section 2.5). In addition, there has been increased interest in going beyond the movements of the actor to the context of the action (Section 2.6), taking into account details such as the scene, objects or interactions with other people.

2.2 Actions as Sequences of Poses

Early works in recognising human motions relied on recovering articulated poses from frame to frame and then linking together either the poses or pose-derived features into a
sequence, or a spatio-temporal trajectory. Pose information was typically obtained from motion capture systems [Campbell & Bobick, 1995], a generate-and-test strategy [Gavrila & Davis, 1995] or segmentation [Gavrila & Davis, 1995; Yacoob & Black, 1999; Rao et al., 2002]. The sequences or trajectories were then classified either through template matching [Gavrila & Davis, 1995; Yacoob & Black, 1999; Rao et al., 2002] or with state-space models such as HMMs [Campbell & Bobick, 1995].

These initial approaches stem directly from the definition of an action as a sequence of articulated poses and are the most straightforward and intuitive. However, they can be highly dependent on obtaining correct pose estimates. More recent works which have followed the idea of limb tracking have resorted to either hand labeling the joints [Yilmaz & Shah, 2005; Ali et al., 2007], or using 2D pose estimates which are not view-point invariant [Ramanan & Forsyth, 2003; Tran et al., 2011].

### 2.3 Actions as Space-Time Volumes

Actions have also been represented as space-time volumes of the whole body [Bobick & Davis, 2001; Weinland et al., 2006; Gorelick et al., 2007], parts of the body [Ke et al., 2010] or areas of high motion correlation [Shechtman & Irani, 2007; Rodriguez et al., 2008]. At the heart of each of these methods is matching a template volume generated from training examples to a query volume generated from the test example. [Bobick & Davis, 2001] used a time-weighted projection to collapse the space-time volume into a motion-history image. [Weinland et al., 2006], using silhouettes from multiple views, extended the concept to motion-history volumes in order to achieve view-point invariance. [Gorelick et al., 2007] considered the intensity of each pixel in the space-time volume of the body volume as a solution to the Poisson equation solution; properties of the solution were then used to characterize the shape of the space-time volume. In each of these approaches, a reliable silhouette must be extracted, making them unsuitable in environments with background clutter or when parts of the body are obstructed. To deal with such problems, [Ke et al., 2010] used over-segmentation to generate spatio-temporal superpixels, of which subsets were then matched to template models according to shape and flow. [Shechtman & Irani, 2007] and [Rodriguez et al., 2008] eschewed segmentation altogether by directly searching for regions of high correlation in the space-time volume, according to optical flow [Shechtman & Irani, 2007; Rodriguez et al., 2008], and temporal derivatives [Rodriguez et al., 2008].
2.4 Actions as Collections of Local Spatio-Temporal Features

With the rise of publicly available video content on the web, focus has shifted towards the analysis of more "natural" videos taken outside of a constrained laboratory setting. In sequences taken from television [Patron-Perez et al., 2010], feature films [Laptev et al., 2008; Marszalek et al., 2009; Kuehne et al., 2011], broadcast sports [Rodriguez et al., 2008; Niebles et al., 2010] and YouTube [Liu et al., 2009b; UCF50], efforts have moved away from modeling the human body towards direct classification of actions with local spatio-temporal features. Features are either densely sampled from sequence, e.g. Gabor filter responses [Jhuang et al., 2007; Schindler & Van Gool, 2008] and optical flow [Efros et al., 2003] or extracted in a sparse manner, e.g. in the case of spatio-temporal interest points such as saliency [Oikonomopoulous et al., 2006; Rapantzikos et al., 2009], cuboids [Dollar et al., 2005], 3D Harris corners [Laptev & Lindeberg, 2003; Schuldt et al., 2004], 3D SIFT [Scovanner et al., 2007], 3D HOG [Klaeser et al., 2008] and 3D Hessians [Willems et al., 2009]. Most of the local spatio-temporal features are extensions of their 2D counterparts used in object detection and their usage follows a traditional object detection approach. After interest point detection at multiple scales, feature descriptors are computed, clustered, and assigned to a code-book to be used in some bag-of-features representation [Laptev et al., 2008; Dollar et al., 2005; Liu et al., 2009b].

In the majority of the works which have adopted a bag-of-features model, the global spatio-temporal aspect of the action is largely ignored [Dollar et al., 2005; Scovanner et al., 2007; Niebles et al., 2008], akin to the spatial-layout of words being ignored in the original bag-of-words model for object detection. For the majority of action recognition tasks, a complete or relaxed independence assumption [Laptev et al., 2008; Savarese et al., 2008; Kovashka & Grauman, 2010] has proven to be sufficient. For example, [Laptev et al., 2008] created a rough spatio-temporal histogramming of the features over the video sequence; [Savarese et al., 2008] looked at the spatial and temporal correlation of the features, while [Kovashka & Grauman, 2010] used a hierarchical approach with varying sizes of spatio-temporal neighborhoods for detecting interest points. A few works which do address the spatio-temporal relationships inherent in actions more directly [Wong et al., 2007; Oikonomopoulous et al., 2009] have used an implicit shape model [Leibe et al., 2008], or have looked at the temporal decomposition of an action into simpler motion segments [Niebles et al., 2010; Gaidon et al., 2011].

---

1In [Rapantzikos et al., 2009], the title "Dense saliency-based spatiotemporal feature points for action recognition" is a misnomer, as their feature points are only "denser" than other methods but is still sparse.
2.5 Actions as a collections of pose-derived features

In many action recognition applications in which the scenario is slightly more constrained, \textit{i.e.} home monitoring or surveillance, it is both natural and helpful to infer activity from the actor and his movements rather than from only a set of low-level spatio-temporal features. [Thurau & Hlavac, 2008; Ikizler-Cinbis et al., 2009; Nazli & Duygulu, 2009; Klaeser et al., 2010b] all applied person-detectors to narrow the focus on features which are related to the human body. The features used in [Thurau & Hlavac, 2008; Nazli & Duygulu, 2009] even aim at encoding the pose information, though pose is never solved for explicitly. [Thurau & Hlavac, 2008] represented poses as a weighted sum of pose primitives and actions as a histogram of the pose primitives associated with the actions’ poses. Similarly, [Nazli & Duygulu, 2009] represented pose as a collection of oriented rectangles and actions as a histogram of the oriented rectangles.

2.6 Actions With Context

In addition to identifying the actions of the main actor in a sequence, many works are trying to incorporate contextual information. The term \textit{context} has been interpreted with differing levels of abstraction, varying from an abstract feature level, such as Gist features [Ikizler-Cinbis & Sclaroff, 2010] and low-level statistics affiliated with detected interest points and key point trajectories [Sun et al., 2009; Wang et al., 2011b] to high-level nameable attributes such as scene, objects and other humans. Adding scene and object information into action recognition can be particularly useful since location is a strong indicator of the actions to be expected, while an object interaction can often be a defining characteristic of an action. [Marszalek et al., 2009] used film scripts to retrieve scene information for labeling actions from films. A number of works have addressed human-object interactions [Moore et al., 1999; Han et al., 2009; Gupta et al., 2009; Ikizler-Cinbis & Sclaroff, 2010], both with [Moore et al., 1999; Han et al., 2009; Gupta et al., 2009] and without explicit object detection [Ikizler-Cinbis & Sclaroff, 2010]. Finally, context has also been addressed in terms of other humans present in the scene, either interacting with each other [Patron-Perez et al., 2010; Ryoo & Aggarwal, 2009] or participating in group activities [Choi et al., 2009, 2011; Lan et al., 2011].

2.7 Appearance- versus pose-based action recognition

Vision-based action recognition techniques can be loosely categorized as appearance-based or pose-based. Appearance-based action recognition uses little to no structural
modeling of the human body and relies on the statistical distribution of appearance features such as edges, shapes and flow to classify actions. The methods discussed in Sections 2.3 and 2.4 fall into this category. Note that the works in Section 2.3 are sometimes referred to as pose-based approaches, in reference to the extracted silhouettes of the human body. However, we consider silhouettes to be a specialized appearance feature, since the overall shape of the body offers little interpretation of the individual body parts. Pose-based methods are those which either implicitly or explicitly solve for the body pose. The works mentioned in Section 2.2 are pose-based methods.

One of the main advantages of appearance-based methods is that they require little to no high-level processing and can bypass the difficulties of pose estimation. They can already take some contextual information into account (e.g. background), since the features are not restricted to the human body. And despite having to deal with great intra-class variations, such as human appearance, background clutter and differing viewpoints, appearance-based systems are applicable in scenarios where pose estimation is difficult (e.g. monocular views) or even impossible (e.g. very low resolutions [Efros et al., 2003]).

Pose-based methods have received little attention in recent years due to the inherent difficulty of extracting human pose, particularly under realistic imaging conditions. Despite requiring more initial processing, pose-based approaches have several advantages. First, pose representations suffer little of the intra-class variances that plague appearance-based systems. In particular, 3D skeleton poses are viewpoint and appearance invariant, such that actions vary less from actor to actor. Secondly, using pose representations greatly simplifies the learning for the action recognition itself, since the relevant high-level information has already been extracted. Given the great progress in pose estimation over the past few years [Bandouch & Beetz, 2009; Gall et al., 2010a; Taylor et al., 2010; Li et al., 2010; Gall et al., 2010b], along with the exploding popularity of the Microsoft Kinect, pose-based action recognition is once again experiencing revived interest.
3

Appearance-based Action Recognition

3.1 Introduction

Recognizing human actions in unconstrained video has garnered growing interest in the computer vision community due to its potential for a multitude of applications. Early works in action recognition focused on classifying video sequences of single persons in controlled environments with simple and uniform backgrounds [Schuldt et al., 2004; Blank et al., 2005]. Recent works have tried to introduce more natural and unconstrained videos, such as sequences from feature films [Laptev et al., 2008], broadcast sports [Rodriguez et al., 2008] and YouTube [Liu et al., 2009b]. In addition to classification, action recognition systems are also trying to address localization, or estimating the spatio-temporal boundaries of a given action [Niebles et al., 2008; Oikonomopoulos et al., 2009; Mikolajczyk & Uemura, 2008; Willems et al., 2009; Ryoo & Aggarwal, 2009]. To this end, the action detection problem, i.e. the localization and recognition of the action, can be considered a form of object detection with higher dimensionality. Recent works have therefore extended object recognition and localization approaches such as interest-point detectors and bag-of-words models into the spatio-temporal domain [Dollar et al., 2005; Liu et al., 2009b; Laptev et al., 2008; Niebles et al., 2008; Wang et al., 2009; Kovashka & Grauman, 2010].

Interest-points, though discriminative, have the drawback of being sparse and as such, they are not robust to non-idealities such as low resolution, motion blur and camera movement more typical of uncontrolled videos. Indeed, [Wang et al., 2009] showed that dense sampling outperforms several types of interest-point detectors. Furthermore, the classical bag-of-words model assumes independence between the spatio-temporal “words” and does not make use of the rich spatio-temporal relationships inherent in actions. While some works have relaxed this independence assumption [Laptev et al., 2008; Niebles et al., 2008; Kovashka & Grauman, 2010], they fail to exploit the information in a more global context within the video sequence.

Inspired by the success of Hough transform-based methods [Ballard, 1981] in object detection [Leibe et al., 2008; Liebelt et al., 2008; Maji & Malik, 2009; Opelt et al., 2008;
Ommer & Malik, 2009; Gall et al., 2011b], we propose a new action detection system based on Hough transform voting. The use of spatio-temporal voting frameworks for action detection has so far been largely unaddressed, mostly due to the high dimensionality of the problem. A direct extension to the spatio-temporal domain would result in a Hough space of at least six dimensions, with votes being cast for location, scale, time and duration, as well as the action itself. If scale and location of the person changes over time during an action, $3 + 3 \cdot T$ parameters need to be estimated in this space, namely $3$ for the temporal boundaries and action class, and $3 \cdot T$ for the location and scale in each frame, where $T$ is the duration of the action.

We approach the high dimensionality problem by separating the voting into two stages and hence two lower-dimensional spaces. In an initial spatial localization stage, we apply a Hough forest trained for people detection [Gall et al., 2011b] and generate detection hypotheses for each frame of the video sequence independently (see Figure 3.1(a)). Due to the strong correlation of the detections’ location and scale across a sequence, the votes can be assembled across time by a particle filter into tracks (see Figure 3.1(b)). For instance, a parallelepiped structure would result for a translation. The detection tracks are then mapped to a cuboid representation, which we call “action tracks”, to obtain a location and scale invariant representation of the person in time (see Figure 3.1(c)). In a subsequent classification and temporal localization stage, votes are then cast for the action’s label and spatio-temporal center on the normalized action track (see Figures 3.1(d) and 3.2).

As far as we know, this is the first time that Hough-voting has been used for action recognition. We perform the voting with a collection of random trees, termed a Hough forest [Gall et al., 2011b], and learn a mapping between densely sampled spatio-temporal feature patches and an action center. Each tree in the Hough forest is trained to discriminate multiple action classes simultaneously. After training, the resulting set of leaf nodes in the trees can be considered a discriminative codebook with shared features across classes, since each leaf can vote probabilistically for all classes. This type of feature-sharing in such a voting framework is novel to our action recognition system and as of yet, has not been demonstrated even in related object detection systems.

For action recognition, our Hough-transform based framework uses randomized tree structures, which have several appealing properties:

1. Randomized trees allow for the use of dense features during training and testing, which yields advantages over the use of sparse features [Wang et al., 2009] and is very useful e.g. in surveillance with its low image resolutions.

2. Instead of learning a separate classifier for each action, we train a single classifier and benefit from the sharing of features between classes as demonstrated in [Torralba et al., 2007].
3.1. Introduction

Figure 3.1: (a) Detection hypotheses are generated for each frame independently. (b) Particle filtering is used to link hypotheses across frames. (c) Action tracks. (d) Hough voting by 3D patches for action label and center.

Figure 3.2: Hough voting space of skateboarding example in Figure 3.1 in (a) \((x, y)\) for the skateboarding class, (b) \((y, \text{time})\) for the skateboarding class and (c) \((\text{action class}, \text{time})\).

3. The trees are directly optimized for the Hough-transform, in the sense that the leaves cast probabilistic votes with small uncertainty for the label and the spatio-temporal location of the action.

In our experiments, we demonstrate and evaluate the use of Hough-voting for action recognition on six datasets covering a wide range of scenarios. We compare both the classification and localization performance of our system against other methods and demonstrate results either comparable or better than the state-of-the-art.
3.2 Related Works

Voting-based frameworks for action recognition have been relatively unaddressed, with the exception of [Wong et al., 2007; Oikonomopoulos et al., 2009], both of which used implicit shape models (ISM). [Wong et al., 2007] extended the probabilistic latent semantic analysis model of [Niebles et al., 2008] with an ISM called the pLSA-ISM, though this approach is unsupervised and aimed at motion detection. The focus of [Oikonomopoulos et al., 2009] was to address temporal segmentation of action sequences. A set of action-specific code-books were used as person detectors. The temporal activation pattern of the codewords were registered and temporal segmentation was then performed by a sliding window approach.

The presented Hough forest voting framework, unlike ISM-based voting approaches, builds up an an implicit codebook (i.e. the leaves of the learned forest) which is optimized for Hough-based detection. This is in contrast to the explicit codebook learned by ISM approaches, which use unsupervised clustering based only on appearance similarity.

The use of trees and forests for action recognition has been previously explored. For example, in [Jiang et al., 2012], a shape-motion prototype tree is built from shape-motion descriptors. In [Mikolajczyk & Uemura, 2008], a vocabulary forest is constructed with local static and flow features. In [Reddy et al., 2009] a sphere/rectangle tree (SR tree) is built with spatio-temporal interest point feature descriptors. All of these works, however, use trees as indexing structures for performing efficient nearest-neighbour searches in either a prototype space [Jiang et al., 2012] or in a feature space in the bag-of-words context [Mikolajczyk & Uemura, 2008; Reddy et al., 2009; Yu et al., 2011]. Actions are classified by weighting the $n$-nearest neighbours and localized either from a foreground segmentation [Lin et al., 2009] or from the features’ spatial information either stored during the training stage [Mikolajczyk & Uemura, 2008] or extracted at the test stage [Reddy et al., 2009]. In [Yu et al., 2011], random forests were used for retrieving actions from database based on single query examples. Similar [Reddy et al., 2009], the random forest was used for efficient indexing of spatio-temporal interest point feature descriptors. Localization was then performed by a branch-and-bound styled search.

Our approach differs fundamentally from each of these works in the sense that we are not building trees to speed up a bag-of-words approach nor nearest-neighbour search. Instead, we use randomized trees to learn the mapping between a 3D video patch and its vote in a 4D Hough space to obtain the class label and the spatio-temporal location of an action in the sense of a generalized Hough transform.
3.3 Voting Framework for Action Recognition

To introduce the concept of using a Hough transform-based voting framework for human action recognition, we first assume that the test sequence is already normalized as shown in Figure 3.1(c), i.e. the sequence is a 3D cuboid. This concept is then generalized in Section 3.3.3.

3.3.1 Training

For training, we assume that for each action class $c \in C$ a set of training sequences is available. Each training example is annotated such that it can be transformed into a normalized action track and contains roughly one action cycle, i.e. it is annotated by a 2D bounding box for each frame, the action label and the temporal boundaries of the action cycle. To learn the mapping between action tracks and a Hough space, we use the Hough forest structure [Gall et al., 2011b] due to its superior efficiency compared to codebook approaches. Since Hough forests were previously developed for 2D single-class object detection, we extend the idea to multi-class detection and the spatio-temporal domain.

Each tree $T$ in the Hough forest $\mathcal{T} = \{T_i\}$ is constructed from a set of patches $\mathcal{P}_i = (I_i, c_i, d_i)$ where,

- $\mathcal{P}_i$ is a 3D patch (e.g. of $16 \times 16 \times 5$ pixels) sampled from the action track as illustrated by the colored cuboids in Figure 3.1(d)
- $I_i$ are extracted features at a patch and can be multi-channeled to accommodate multiple features, i.e. $I_i = (I_i^1, I_i^2, ..., I_i^F) \in \mathbb{R}^F$, where each $I_i^f$ is feature channel $f$ at patch $i$ and $F$ is the total number of feature channels. In our system, we used six low-level feature channels: greyscale intensity, the absolute value of $x$, $y$ and time derivatives, and the absolute value of optical flow [Brox et al., 2004] in $x$ and $y$.
- $c_i$ is the action label ($c_i \in C$)
- $d_i$ is a 3D displacement vector from the patch center to the action track center. Since the patches are being drawn from the spatially normalized action tracks, the center displacements are spatially scale-invariant but not temporally. At run-time, however, the Hough forests can be applied at multiple scales to achieve temporal scale-invariance.

Each leaf node $L$ stores $p_c^L$, the proportion of patches per class label reaching the leaf after training, i.e. $\sum_c p_c^L = 1$, and $D_c^L = \{d_i\}_{c_i = c}$, the patches’ respective displacement vectors. Each non-leaf node $B$ of a tree is assigned a binary test in relation to the patch appearance $I$ during training. There are many possibilities in defining the binary test; we
use a simple comparison of two pixels at locations $p \in \mathbb{R}^3$ and $q \in \mathbb{R}^3$ in feature channel $f$ with some offset $\tau$. The binary test at node $B$, $t_B$, can be defined as

$$t_{B,f,p,q,\tau}(I) = \begin{cases} 0 & \text{if } I^f(p) < I^f(q) + \tau \\ 1 & \text{otherwise} \end{cases}$$ (3.1)

The random trees in Hough forests are constructed according to a standard random forest framework [Breiman, 2001]. Construction begins at the root by choosing a binary test, splitting the training patches according to the test results and then constructing children nodes. At each subsequent child node, the same procedure continues recursively, with each node being designated as a non-leaf node until the termination criteria is met, i.e. the child node is of a maximum depth, or there are less than a minimum number of patches remaining. Upon termination as a leaf, the remaining patches’ information, $(p^c_L, d^c_L)_{c \in C}$, is stored; otherwise, another binary test is chosen and the patches are split again. Since the patches from all classes $c \in C$ that pass the binary tests and arrive at a certain leaf share the same features, the probabilities $p^c_L$ represent the degree of sharing between the action classes (see Figure 3.4(b) for an example).

The ideal binary test should split the patches in such a way as to minimize the uncertainty of their class label and center offsets. To do this, we developed two measures to evaluate the uncertainty for a set of patches $A = \{P_i = (I_i, c_i, d_i)\}$. The first measure aims to minimize class uncertainty:

$$U_1(A) = -|A| \cdot \sum_c p_c \ln(p_c),$$ (3.2)

where $|A|$ is the number of patches in set $A$ and $p_c$ is the proportion of patches with label $c$ in set $A$. Note that the summation expression is the standard definition of entropy for the class labels. The second measure aims to minimize center offset uncertainty:

$$U_2(A) = \sum_i \|d_i - \overline{d_A}\|,$$ (3.3)

where $\overline{d_A} \in \mathbb{R}^3$ is the mean offset vector of set $A$. Note that the offset uncertainty is minimized for all classes at the same time.

At each node during training, a pool of binary tests $\{t^k\}$ is generated with random values of $f$, $p$, $q$ and $\tau$ falling within the constraints of the training data. Then, either class or offset uncertainty is chosen at random to be minimized at that given node. The set of patches arriving at the node will be evaluated with all binary tests in the pool and the binary test satisfying the following minimization objective will be chosen:

$$\arg\min_k \left( U_*(\{P_i | t^k = 0\}) + U_*(\{P_i | t^k = 1\}) \right),$$ (3.4)

where $*$ indicates the chosen uncertainty measure for the node. By randomly selecting the uncertainty measure, nodes decreasing both class and offset uncertainty are interleaved throughout the tree. As such, patches being stored at the leaves tend to have low variation in both class label and center displacement and vote with low uncertainty into the Hough-space.
3.3.2 Classifying and Localizing Actions

To classify and localize an action within an action track $S$, extracted patches are passed through each of the trees in the random forest; the leaves that the patches arrive in are then used to cast votes for the action label and the spatio-temporal center. To begin with, consider a patch $P(y) = (I(y), c(y), d(c(y), y))$ located at position $y \in \mathbb{R}^3$ in the action track, where $I(y)$ are the patch’s features, $c(y)$ the patch’s unknown class label and $d(c(y), y)$ the displacement of the patch from the unknown action’s center. Let $Q_c(x)$ be the event of the action track belonging to class label $c$ and centered at $x \in \mathbb{R}^3$. We are interested in finding the conditional probability $p(Q_c(x) | I(y))$, which can be decomposed as follows:

$$p(Q_c(x) | I(y)) = \sum_{l \in C} p(Q_c(x) | c(y) = l, I(y)) p(c(y) = l | I(y)) = p(Q_c(x) | c(y) = c, I(y)) = p(d(c, y) | c(y) = c, I(y)) p(c(y) = c | I(y)).$$  

Both factors in (3.5) can be estimated by passing the patch $P(y)$ through the trees. Suppose that the patch ends up in leaf $L$ of tree $T$. The first factor can then be approximated as the Parzen-window estimate of $D_L^c$, the offset vectors belonging to class $c$, while the second factor can be approximated as $p_L^c$, the probability of the patch belonging to class $c$. We can then rewrite Equation (3.5) for tree $T$ as

$$p(Q_c(x) | I(y), T) = \left( \frac{1}{|D_L^c|} \sum_{d \in D_L^c} G((y - x) - d) \right) \cdot p_L^c,$$

where $G$ is the 3D Gaussian Parzen window function. For the entire forest $T$, we average over all the trees

$$p(Q_c(x) | I(y), T) = \frac{1}{|T|} \sum_{t} p(Q_c(x) | I(y), T_t).$$  

Equations (3.6) and (3.7) define the probabilistic vote of a single patch $P$ for action class $c$. Votes from all patches are integrated into a 4D Hough accumulator:

$$V(x, c) = \sum_{y \in S(x)} p(Q_c(x) | I(y), T).$$

Note that the action classes are treated independently in the Hough accumulator while the 3D votes for each class are smoothed by a Gaussian window (see Eq.3.6).
Since the action track has already been localized in space, the Hough accumulator is marginalized across the spatial dimensions into a 2D image in class label and time. Note, however, that we vote in the spatial dimensions despite having spatially localized tracks in order to enforce spatial regularity of the patches during training, i.e. grouping together the patches which vote towards the actions 3D center. The local maxima in the remaining Hough accumulator are used to determine the track’s label and localization in time as illustrated in Figure 3.2.

To achieve time-scale invariance, the action tracks can in theory be either up- or down-sampled accordingly and the same Hough forest can then be applied to label actions at differing speeds. We note, however, that action speeds (disregarding a variation in frame-rate) typically do not vary more than by a factor of two. Furthermore, the system has some tolerance built in through variation in speed of the training data and in our current work, we found it unnecessary to apply the Hough forest in multiple time scales.

3.3.3 Building Action Tracks

In order to obtain action tracks from the test sequences, i.e. a cuboid representation normalized by scale and position of the human, we learn the mapping from the image domain to a 3D Hough space that encodes position and scale of a human. We train a Hough forest for the spatial domain similar to Section 3.3.2, where we use cropped and scale-normalized images of humans as positive examples and background images as negative examples. As features, we use color and histograms of gradients [Dalal & Triggs, 2005]. The voting is performed for each frame independently and detections are used to initialize an action track as illustrated in Figure 3.1.

Since humans show a large variation in pose and appearance, particularly for sport clips (Figure 3.4(c)), we do not get a correct detection for each frame. The votes, however, can be efficiently assembled into tracks by a particle filter [Doucet et al., 2001] due to the strong correlation of location and scale across a sequence. We model the state \( s \in \mathbb{R}^6 \) of a human by the position, scales, and velocity of its bounding box. As dynamical model \( p(s_t | s_{t-1}) \), we use a simple Gaussian process and camera motion compensation which is estimated from optical flow. The likelihood for an image \( I_t \) is given by

\[
p(I_t | s_t) = \frac{1}{Z} \exp \left( - \left( \alpha V_1(s_t) + \beta \sum_f V_2(s_t, f) \right) \right)
\]

with \( f \) features extracted from the bounding box. The parameters \( \alpha = 40 \) and \( \beta = 100 \) were experimentally determined. The term \( V_1 \) measures the response in the Hough space \( V \) (Figure 3.1(a)) for the candidate \( s_t \):

\[
V_1(s_t) = - \log \left( \sum_{x \in V} G(s_t - x) \right),
\]

(3.10)
i.e. we sum the votes in the neighborhood of $s_t$ weighted by a Gaussian kernel $G$. Note that the sum is in the range of $[0, 1]$. The term $V_2$ measures the similarity of the current particle with the initial detected bounding box $s_0$, i.e. the appearance is not updated, using the Bhattacharya distance:

$$V_2(s_t, f) = 1 - \sum_k \sqrt{h_k^f(s_0)h_k^f(s_t)}.$$  (3.11)

As image features, we use the HSV colour space and a local binary pattern operator [Ojala et al., 2002]. The extracted bounding boxes for a track are then normalized into spatial- and scale-invariant cuboids as shown in Figure 3.1. A more thorough explanation of the tracker used in building the action tracks as well as more extensive evaluations can be found in Appendix A, Tracking People in Broadcast Sports.

3.4 Experiments

3.4.1 Datasets

We evaluated our system on six datasets, covering a variety of action-recognition scenarios. The first two, Weizmann [Blank et al., 2005] and KTH [Schuldt et al., 2004], are popular benchmarks used in action recognition and consist of single persons performing actions in front of static backgrounds. Current state-of-the-art recognition systems have saturated in performance on these two datasets, but we include their evaluation for comparison purposes against other systems. We also evaluate our system on four more challenging datasets: the UCF sports dataset [Rodriguez et al., 2008], the UCR Videoweb Activities Dataset [Denina et al., 2010], the UT-Tower Dataset [Chen et al., 2010], and the TUM Kitchen Dataset [Tenorth et al., 2009].

3.4.2 Evaluation Measures

Classification

We evaluated our system’s ability to apply the correct action label to a given video sequence and call this classification. Classification was measured with three variations of training and testing data: (A) training and testing on tracks generated from ground-truth annotations (B) training on tracks from ground truth and testing on automatically extracted tracks and (C) training and testing on automatically extracted tracks. We refer to these as data variations A, B, and C respectively from this point on.
Localization

For the KTH and UCF sports dataset, we also evaluate the accuracy of detections in the automatically extracted action tracks and call this localization. The localization evaluation is the same as [Mikolajczyk & Uemura, 2008]; a detection is considered correct if (1) the action track that it belongs to was correctly classified and (2) the intersection-union ratio of the detection and ground truth bounding box is greater than 0.5. For the KTH dataset, selected frames were hand-annotated with bounding boxes and the bounding boxes for the frames in between were generated by linear interpolation. For the UCF dataset, bounding boxes were provided as part of the ground truth annotation released with the data. As a measure of localization, we present the average precision, or the area below the precision-recall curve.

3.5 Results

3.5.1 Benchmarks: Weizmann and KTH

The Weizmann dataset consists of 90 videos of nine actors performing ten different actions. Evaluations were done with a leave-one-out cross-validation. Eight actors were used for training and the ninth for testing; this was repeated over all nine actors and the results were averaged.

The KTH dataset consists of 599 videos of 25 actors performing six actions under four scenarios. We group the scenarios together as one large dataset rather than treating each scenario separately. Evaluations were done with a five-fold cross-validation. 20 actors were used for training and five for testing; this was repeated five times, such that all actors were used for testing once, and the results were averaged over all folds. As each sequence lasts several hundred frames, we limited each sequence to only one or two cycles of the action in our evaluation.

Classification results over the three variations of training / testing data are shown in Table 3.1 and compared with state-of-the-art results. [Liu et al., 2009b; Wang et al., 2009; Kovashka & Grauman, 2010; Klaeser et al., 2010a] are all variations of bag-of-words models using spatio-temporal interest points. [Klaeser et al., 2010a] uses

The confusion matrix for training on ground-truth action tracks and classification on automatically extracted action tracks (data variation B) are shown in Figure 3.3.

With data variation B, we report an average classification of 95.6% for Weizmann and 92.0% for KTH, both of which are comparable with state-of-the-art action recognition systems. The closest comparison is that of [Reddy et al., 2009], in which a random forest
3.5. Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Weizmann</th>
<th>KTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hough forest (data A)</td>
<td>97.8%</td>
<td>93.5%</td>
</tr>
<tr>
<td>Hough forest (data B)</td>
<td>95.6%</td>
<td>92.0%</td>
</tr>
<tr>
<td>Hough forest (data C)</td>
<td>92.2%</td>
<td>93.0%</td>
</tr>
<tr>
<td>pLSA-ISM [Wong et al., 2007]</td>
<td>-</td>
<td>83.9%</td>
</tr>
<tr>
<td>temporal segments [Oikonomopoulos et al., 2009]</td>
<td>-</td>
<td>81.2%</td>
</tr>
<tr>
<td>vocabulary forest [Mikolajczyk &amp; Uemura, 2008]</td>
<td>-</td>
<td>93.2%</td>
</tr>
<tr>
<td>SR tree [Reddy et al., 2009]</td>
<td>-</td>
<td>90.3%</td>
</tr>
<tr>
<td>random forest [Reddy et al., 2009]</td>
<td>-</td>
<td>72.9%</td>
</tr>
<tr>
<td>prototype tree [Lin et al., 2009]</td>
<td>100%</td>
<td>93.4%</td>
</tr>
<tr>
<td>[Liu et al., 2009b]</td>
<td>-</td>
<td>93.8%</td>
</tr>
<tr>
<td>[Wang et al., 2009]</td>
<td>-</td>
<td>92.1%</td>
</tr>
<tr>
<td>[Kovashka &amp; Grauman, 2010]</td>
<td>-</td>
<td>94.5%</td>
</tr>
</tbody>
</table>

*Table 3.1*: Comparison of KTH and Weizmann classification with other methods. Results of all other methods presented are comparable with data variation B. See 3.4.2 for definitions of the data variations.

of 50 trees were trained as a comparison against the sphere/rectangle trees; our performance, with only 5 trees in the random forest, is significantly higher and highlights the strength of the Hough-voting framework.

Localization results of each action in the dataset are presented in Table 3.2; both our method and the vocabulary forest method [Mikolajczyk & Uemura, 2008] achieve an average precision of 0.89 over all classes.

3.5.2 Broadcast Sports: UCF Sports

The UCF sports dataset is a collection of 150 broadcast sports sequences from network news videos and features ten different actions: diving, golfing, kicking, weightlifting, horseback-riding, running, skateboarding, swinging 1 (gymnastics, on the pommel horse and floor), swinging 2 (gymnastics, on the high and uneven bars) and walking. Evaluations were done with a five-fold cross-validation. Due to an unequal number of sequences in each action category, each fold consisted of approximately one-fifth of the total number of sequences per category. Four folds were used for training while the fifth was used for

<table>
<thead>
<tr>
<th>Method</th>
<th>Box</th>
<th>Clap</th>
<th>Jog</th>
<th>Run</th>
<th>Walk</th>
<th>Wave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hough forest</td>
<td>0.88</td>
<td>0.96</td>
<td>0.84</td>
<td>0.72</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td>vocabulary forest</td>
<td>0.98</td>
<td>0.97</td>
<td>0.79</td>
<td>0.78</td>
<td>0.86</td>
<td>0.96</td>
</tr>
</tbody>
</table>

*Table 3.2*: Comparison of KTH localization results.
testing; this was repeated five times, such that all clips were used once for testing and results were averaged over all folds.

Classification results over the three variations of training and testing data are shown in Table 3.3 and compared with the results reported from other methods. We outperform [Rodriguez et al., 2008] and [Yeffet & Wolf, 2009], and have comparable results with [Wang et al., 2009] though the we use much simpler features (their best results were achieved using 3D-HOG descriptors).

The confusion matrix for training on ground-truth action tracks and classification on automatically extracted action tracks are shown in Figure 3.4 along with some example classification results. There is some confusion between running and kicking, since kicking sequences often open with an individual running before kicking a ball. Similarly, walking

### Table 3.3: Comparison of UCF classification with other methods. Results of all other methods presented are comparable with data variation B.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hough forest (data A)</td>
<td>86.6%</td>
</tr>
<tr>
<td>Hough forest (data B)</td>
<td>81.6%</td>
</tr>
<tr>
<td>Hough forest (data C)</td>
<td>79.0%</td>
</tr>
<tr>
<td>[Rodriguez et al., 2008]</td>
<td>69.2%</td>
</tr>
<tr>
<td>[Yeffet &amp; Wolf, 2009]</td>
<td>79.2%</td>
</tr>
<tr>
<td>[Wang et al., 2009]</td>
<td>85.6%</td>
</tr>
<tr>
<td>[Kovashka &amp; Grauman, 2010]</td>
<td>87.3%</td>
</tr>
<tr>
<td>[Klaeser et al., 2010a]</td>
<td>86.7%</td>
</tr>
<tr>
<td>[Wang et al., 2011a]</td>
<td>88.2%</td>
</tr>
</tbody>
</table>

**Figure 3.3:** Confusion matrices for Weizmann and KTH dataset using ground truth action tracks for training and automatically extracted action tracks for testing.
3.5. Results

and golfing sequences are also confused since several walking sequences are drawn from individuals walking on a golf course, suggesting that the trees take some context into account when splitting the patches.

Localization results for the UCF dataset are presented in Table 3.4; no other works at this time have published a similar evaluation for comparison. Over all classes, we achieve an average precision of 0.54. The low average precision can be attributed to the fact that the ground truth annotations have changing aspect ratios, while we assumed a fixed aspect ratio when generating the action tracks since we are interested only in a cuboid representation. This is particularly relevant for the sports in which the people have irregular and rapidly changing articulations, such as diving, kicking and the swinging classes. Classification performance in these classes, however, are still very high because the cuboid representation is sufficient to capture the action.

To emphasize the effects of feature sharing, we plot the probability of a patch from a given class sharing features with a patch from another class (Figure 3.4(b)). The diving and weight-lifting classes are very distinct and share little to no features with other actions. On the other hand, the two gymnastics swing classes are very similar to (and only with) each other, and as such, share features with each other. There also exists less distinct groupings, such as walking, golfing, skateboarding and kicking, suggesting that both body position and context are accounted for in the feature sharing. For example, walking, golfing, and skateboarding all involve upright individuals with legs in relatively straight alignment with the body. On the other hand, several walking sequences are drawn from people walking on golf-courses with green fields, which also resemble the soccer fields in the kicking sequences.

### 3.5.3 Surveillance: UCR Videoweb Activities

The UCR Videoweb Activities Dataset consists of about 2.5 hours of video footage from four to eight cameras in various surveillance scenarios. From this footage, we selected 110 sequences of eight actions that not only illustrate changes in body configuration (sitting down, standing up), but also interaction with the environment (entering and exiting a car, opening and closing a trunk), interaction with other people (shaking hands) and interaction with objects (tossing a ball). Evaluations were done with a five-fold cross-validation.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Class</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dive</td>
<td>0.52</td>
<td>Kick</td>
<td>0.28</td>
</tr>
<tr>
<td>W.Lift</td>
<td>1</td>
<td>Run</td>
<td>0.37</td>
</tr>
<tr>
<td>Walk</td>
<td>0.67</td>
<td>Ride</td>
<td>0.66</td>
</tr>
<tr>
<td>Golf</td>
<td>0.77</td>
<td>Swing 1</td>
<td>0.44</td>
</tr>
<tr>
<td>Sk. Board</td>
<td>0.39</td>
<td>Swing 2</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 3.4: UCF localization results
in the same manner as the UCF sports dataset. As our system handles only monocular views, we treat the same action instance recorded by different cameras as different sequences. As this is a newly released dataset, there are no other works with comparable results that we know of.

We achieve an average performance of 91.5%, 85.2% and 92.6% for variations A, B, and C of training and testing data (see 3.4.2). The confusion matrix for variation B is shown in Figure 3.5, together with some example classification results. As expected, there are some confusions between action pairings such as sit down/stand up, enter/exit car and open/close trunk. Performance is remarkable considering the small size of the people in the surveillance sequences (typically 40 to 60 pixels high).

3.5.4 Aerial Footage: UT Tower Dataset

The UT-Tower Dataset [Chen et al., 2010] is a collection of videos taken from the top of a 90 meter tall tower. There are 12 actors performing 9 actions and due to the distant view, the average height of the actors are only around 20 pixels high. Due to the low resolution, we tracked the individuals based on foreground masks included in the dataset and train and test as per variation C. We achieved an overall classification performance of 95.4%. The confusion matrix and sample images are shown in Figure 3.6. There is some confusion between similar actions such as standing and pointing or wave1 and wave2 but all other actions are classified correctly.

3.5.5 Inhouse Monitoring: TUM Kitchen Dataset

Although the action recognition system as described in 3.3 is meant only for monocular videos, we extended it for a multi-view scenario. A separate Hough forest is trained for each of the cameras in the multi-view setup; the output per view is a confidence score of each action class over time, normalized such that the confidences over all classes at any time point sum up to 1 (see Figure 3.8). A classifier combination strategy is then used to combine the outputs from the multiple views [Kittler et al., 1998]. The motivation for fusing the single views is that actions which are ambiguous in one view, e.g. due to self-occlusion, may be more distinguishable from another view.

We apply the extended multi-view algorithm to the TUM Kitchen Dataset [Tenorth et al., 2009], recordings from 4 views of 4 subjects setting a table. The dataset is particularly challenging for action recognition as the actions are more subtle than those of KTH, Weizmann, UCF Sports, etc. In this dataset, the cameras are fixed and background subtraction was used to generate silhouettes of the person performing the action. Bounding boxes are then extrapolated around the silhouette and the trajectory of the bounding boxes is smoothed to build the track.
3.5. RESULTS

Figure 3.4: (a) Confusion matrix for UCF sports dataset using data variation B. (b) Probability of feature sharing in a patch between action classes. (c) Example classifications.

Figure 3.5: Confusion matrix for Videoweb Activities dataset for data variation B and example classifications.

Figure 3.6: Confusion matrix and sample images for classification on the UT-Tower Dataset.
Training was done on episodes 1-0 to 1-5, all of which are recorded from subject 1 and testing was done on episodes 0-2, 0-4, 0-6, 0-8, 0-10, 0-11, and 1-6, which are recorded from all 4 subjects. For the action recognition, we use the 9 labels that are annotated for the ‘left hand’ [Tenorth et al., 2009] and further split the idle/carry class according to whether the subject is walking or standing.

Results of the action recognition for the individual cameras as well as the fused results are shown in Table 3.5. For classifier fusion, we use the max-rule that gave the best performance compared to other standard ensemble methods [Kittler et al., 1998], though results were similar for all the methods. The confusion matrix for the fused classifier is shown in Figure 3.7. Figure 3.8 shows examples of action confidences for two single views and for the fused views.

![Confusion Matrix](image)

**Figure 3.7:** Confusion matrix for fused results of the TUM Kitchen Dataset according to the max-rule.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>Fused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>54.2%</td>
<td>49.3%</td>
<td>56.9%</td>
<td>56.4%</td>
<td>57.4%</td>
</tr>
<tr>
<td>Subject 2</td>
<td>53.2%</td>
<td>50.1%</td>
<td>45.6%</td>
<td>56.0%</td>
<td>58.5%</td>
</tr>
<tr>
<td>Subject 3</td>
<td>69.0%</td>
<td>71.8%</td>
<td>65.2%</td>
<td>66.6%</td>
<td>74.0%</td>
</tr>
<tr>
<td>Subject 4</td>
<td>61.9%</td>
<td>52.9%</td>
<td>61.0%</td>
<td>61.0%</td>
<td>70.6%</td>
</tr>
<tr>
<td>Average</td>
<td>59.6%</td>
<td>56.0%</td>
<td>57.2%</td>
<td>60.0%</td>
<td>65.1%</td>
</tr>
</tbody>
</table>

**Table 3.5:** Individual camera and fused action recognition performance for subjects 1-4; fused performance is higher than any individual camera view for each subject.
3.5. Results

![Figure 3.8](image)

**Figure 3.8:** Normalized action confidences for two camera views as well as fused results for frames 500-900 of episode 0-11.

![Figure 3.9](image)

**Figure 3.9:** For both the (a) KTH dataset and the (b) TUM Kitchen dataset, performances decreases when patch sampling is reduced. At 1 (dense), $10^{-2}$, $10^{-3}$, and $10^{-4}$, the average overlaps of two nearest patches are 91%, 56%, 39%, and 7%, respectively.
3.5.6 Dense patch sampling

The use of randomized trees allows for dense sampling of features, which has been shown to outperform sparse features at spatio-temporal interest points [Wang et al., 2009]. We investigate our system’s performance with respect to decreased patch sampling rates on the KTH and TUM Kitchen dataset. In Figure 3.9, the average classification performance is plotted with respect to sampling density. Since we are using dense sampling in three dimensions, there is considerable overlap between patches. Performance does not drop until around one percent of the original sampling density but from this point onwards, the performance decrease is smooth. This shows that the amount of data processing can be reduced by a factor up to 100 for time-critical applications.

On 100 frames of the KTH dataset, it takes around 10s to classify pre-existing action tracks and 170s to generate an action track on a standard PC.

3.6 Discussion and Outlook

In this chapter, we present a method for appearance-based action classification and localization using a Hough-transform based voting framework. We approach the problem from a object detection perspective, but split the otherwise high-dimensional and intractable problem into two lower-dimensional spaces. Our system can classify and localize actions in unconstrained video sequences, achieving state-of-the-art performance on two benchmarks datasets and also more challenging and realistic video sequences from sports broadcasts and various surveillance scenarios.

The system presented in the current chapter has also been applied to expression recognition, using optical flow and feature responses from a bank of Gabor filters. More details are provided found in Appendix B, Facial Expression Recognition. The classifier combination approach used to fuse camera views together for multi-view action recognition (see Section 3.5.5) has also been applied for group action recognition, details of which are provided in Appendix C, Group Action Recognition.

Currently, the system already shows promising performance using only low-level features; with more sophisticated features, we expect performance to improve even more. Future work includes coupling the building of action tracks with their classification and temporal localization simultaneously.
4

Leveraging Actions for Pose Estimation

4.1 Introduction

3D human pose estimation in multi-camera scenarios is an active field of research [Moeslund et al., 2006]. While recent approaches [Bo & Sminchisescu, 2010; Gall et al., 2010a; Lee & Elgammal, 2010; Li et al., 2010] report impressive results on benchmarks like HumanEva [Sigal et al., 2010], real-world applications such as in-house monitoring still pose many challenges. For example, background clutter, occlusions, and interactions with objects are all difficulties not encountered in studio recordings. In addition, realistic human models have over 50 parameters to estimate [Tenorth et al., 2009], yielding a high-dimensional optimization problem that needs to be solved efficiently.

To maintain robustness in more unconstrained scenarios, the use of priors on human actions and dynamics have become very popular. For instance, the poses of a certain group of actions can be embedded into a low-dimensional manifold [Li et al., 2010; Moon & Pavlovic, 2006; Urtasun et al., 2006]. While ‘full-body’ motions like walking, jogging, and golf swings can be nicely embedded, learning embeddings for more ambiguous actions like ‘carrying an object’, particularly from sparse and noisy data, is a much more difficult task. Furthermore, the complexity increases with the number of actions and many dimensionality reduction techniques struggle to establish useful embeddings for a high number of actions. Instead of embedding all actions into a single manifold, creating separate, action-specific manifolds is an easier task to solve. Moreover, this allows for the incremental addition of new actions, which is an important property to have in practice. Using multiple manifolds, however, leads to an unsolved problem: how can we estimate the pose from a set of manifolds? An approach would be to learn the transitions between each manifold, using techniques like motion graphs [Kovar et al., 2002] or switching models [Chen et al., 2009], but this does not scale with the number of actions.

Here, we propose a new algorithm for optimizing over a set of manifolds that can efficiently estimate human pose even in challenging scenarios like the TUM kitchen dataset [Tenorth et al., 2009]. We have adapted a particle-based annealing optimization scheme
4.2 Related Work

One of the most popular ways to reduce the complexity of the human pose estimation problem is to use a prior model learned from motion capture databases. The most basic approaches rely on database matching, where the previously estimated poses in the sequence are used as a query to search the most similar motion exemplar in a database. Approaches can be either on-line, to predict the pose for the next frame [Sidenbladh et al., 2002; Rosenhahn et al., 2007], or offline, to refine the tracked poses [Baak et al., 2009].

Since exemplar-based models do not generalise well, several methods have been proposed to model priors in low-dimensional spaces. Among the simplest are those based on PCA [Baumberg & Hogg, 1994; Sidenbladh et al., 2000; Urtasun et al., 2005]. More complex priors include those generated from dimensionality reduction techniques such as Isomap [Tenenbaum et al., 2000] (see Gall et al. [2010b]), LLE [Roweis & Saul, 2000] (see [Elgammal & Lee, 2004; Jäggli et al., 2009; Lee & Elgammal, 2010]) and Laplacian Eigenmaps [Belkin & Niyogi, 2002] (see [Sminchisescu & Jepson, 2004]) or probabilistic latent variable models such as the commonly used GPLVM [Lawrence, 2005] and GPDM [Wang et al., 2008] (see [Urtasun et al., 2006; Moon & Pavlovic, 2006; Hou et al., 2007; Geiger et al., 2009; Ukita et al., 2009]). More recently, Taylor et al. [2010] introduced the use of Conditional Restricted Boltzmann Machines, composed of large collections of discrete latent variables.

Instead of building priors on poses or motion models, other approaches learn a mapping between the image space and the pose space. These approaches recover the pose directly from silhouettes and image features [Rosales & Sclaroff, 2001; Agarwal & Triggs, 2006; Sminchisescu et al., 2007; Bo & Sminchisescu, 2010]. In [Taycher et al., 2006], for instance, pose estimation is formulated as inference in a conditional random field model where the observation potential function is learned from a large set of training data.
Little work has been done to leverage action recognition outputs for pose estimation, perhaps because much of the previous work in pose estimation has been focused on sequences of single action classes rather than multi-actioned sequences. In [Raskin et al., 2011], an annealed particle filter [Deutscher & Reid, 2005] was used for tracking in a low dimensional space trained on a few basic actions; action classification is then performed on the tracked poses. A similar approach was proposed in [Darby et al., 2010] where PCA is used for dimensionality reduction and a hidden Markov model for modeling dynamics, but in contrast to [Raskin et al., 2011] transitions between different actions are modelled explicitly.

Since the complexity increases with the number of actions and many dimensionality reduction techniques struggle to establish useful embeddings for a high number of actions, mixture models [Lin et al., 2006; Li et al., 2007, 2010] or switching models [Pavlovic et al., 2000; Jäggli et al., 2009; Chen et al., 2009] that rely on action-specific manifolds have been shown to be more flexible. We also follow the concept of action-specific manifolds. However, we do not need to observe transitions between actions for training since we do not model pose estimation as a filtering problem over time but as an optimization problem over the manifolds for each frame.

### 4.3 Framework

The multi-view system can be decomposed into action recognition on the 2D images and 3D pose estimation, with the action-specific manifolds acting as a link between the two. First, silhouettes are used to establish a track of the person over the sequence; the action recognition system then assigns labels for the track over time (Section 4.4). The confidence measure of the action labels are then used to distribute the particles in the particle-based optimization scheme over the action-specific manifolds (Section 4.5.1). Finally, the pose is estimated by an optimization over the entire set of manifolds (Section 4.5.3).

### 4.4 Action Recognition

For action recognition, a separate classifier is trained for each of the cameras in the multi-view setup; results from the individual classifiers are then combined with standard classifier ensemble methods (see 3.5.5 of Chapter 3). Motivation for fusing the single views is based on the assumption that actions which are ambiguous in one view, e.g. due to self-occlusion, is more distinguishable from another view.

Action recognition is performed according to the Hough-transform voting method presented in [Yao et al., 2010a] (see Chapter 3). It breaks down the action recognition problem into an initial localization stage, which generates tracks of the individual performing
4. Leveraging Actions for Pose Estimation

Figure 4.1: System Overview. (a) Silhouettes are extracted by background subtraction. (b) Tracks are built over the entire sequence and classified by a 2D action recognition system. (c) Confidences of each action are used to distribute the particles over the action-specific manifolds. (d) Final pose is obtained by optimizing over the manifolds.

Figure 4.2: For each action class \( a \), we learn an embedding in a low-dimensional manifold \( \mathbb{M}_a \). The manifolds are indicated by small circles while the high-dimensional state space \( \mathbb{E} \) is indicated by large circle. Having estimated the pose \( x_{t-1} \), a set of particles is selected from the previous particle sets (Select \( p_1 \)). To this end, the particles in \( \mathbb{E} \) are mapped by \( f_a \) to \( \mathbb{M}_a \) where each particle is associated to a manifold. This process is steered by a prior distribution on the actions obtained by the 2D action recognition system described in Chapter 3. Since the manifolds are action-specific, the pose for the next frame can be predicted by the function \( h_a \). The first optimization step, Optimization \( A \), optimizes jointly over the manifolds and the human poses embedded in the manifolds. Since our manifolds do not cover transitions between actions, we run a second optimization step, Optimization \( B \), over the particles mapped back to the state space \( \mathbb{E} \) by \( g_a \). Before the optimization, the particle set is augmented by making use of the embedding error of the previous pose \( x_{t-1} \) (Select \( p_2 \)).
the action, and a subsequent classification stage, which assigns action labels to the tracks. In scenarios where the cameras are fixed, it is not necessary to build the tracks with a tracking-by-detection technique as presented in [Yao et al., 2010a]. Instead, background subtraction is used to generate silhouettes of the person performing the action (see Figure 4.1). Bounding boxes are then extrapolated around the silhouette and the trajectory of the bounding boxes is smoothed to build the track.

The output of the classification stage is a confidence score of each action class over time, normalized such that the confidences over all classes at any time point sum up to 1. A classifier combination strategy such as the max-rule is then used to combine the outputs from the multiple cameras [Kittler et al., 1998].

4.5 Optimizing Over a Set of Manifolds

Having a skeleton and a surface model of the human, the human pose is represented by the joint angles $\Theta = \theta_1, \ldots, \theta_D \in \mathbb{E}_\Theta$ and the global orientation $r$ and position $t$, yielding a $D+6$ dimensional state space denoted by $\mathbb{E}$. In this paper, we formulate pose estimation as an optimization problem over $\mathbb{E}$ for a given positive energy function $V$, i.e. $\min_{x \in \mathbb{E}} V(x)$.

As the energy function, we use the negative log-likelihood based on edge and silhouette features as in [Shaheen et al., 2009], i.e.

$$V(x) = \lambda_{\text{edge}} \cdot V_{\text{edge}}(x) + \lambda_{\text{silh}} \cdot V_{\text{silh}}(x).$$ (4.1)

$\lambda_{\text{edge}}$ and $\lambda_{\text{silh}}$ controls the influence of the edge and silhouette terms respectively. $V_{\text{edge}}$ and $V_{\text{silh}}$ are determined by comparing the edges and silhouettes in the observed image versus that which is generated by projecting the human model according to the pose encoded in $x$. More precisely,

$$V_{\text{edge}} = \frac{|E_P(x) \not\subseteq E_I|}{|E_P(x)|},$$ (4.2)

i.e., the fraction of pixels observed in $E_P(x)$, the projected edge map from the model, which do not overlap with $E_I$, the edge map observed in image $I$ by applying a Sobel operator. Similarly,

$$V_{\text{silh}} = \frac{|S_P(x) \not\subseteq S_I|}{2 \cdot |S_P(x)|} + \frac{|S_I \not\subseteq S_P(x)|}{2 \cdot |S_I|},$$ (4.3)

i.e., the fraction of pixels in $S_P(x)$, the projected silhouette from the model, which do not overlap with $S_I$, the silhouette observed in image $I$ by applying background subtraction.
and vice versa. Note that edges and silhouettes are not optimal features for human pose estimation, since edges are sensitive to background clutter, clothing textures and wrinkles, while silhouettes are sensitive to occlusions and background changes. However, the associated energy function is fast to compute and therefore fixed for all our experiments. As a baseline, we implemented Interacting Simulated Annealing (ISA), a particle-based annealing optimization scheme over $E$. ISA has been used previously in the multi-layer pose estimation framework in [Gall et al., 2010a]. The optimization scheme, based on the theory of Feynman-Kac models [Del Moral, 2004], iterates over a selection and mutation step, and is also the underlying principle of the annealed particle filter [Deutscher & Reid, 2005].

We modify the baseline algorithm to optimize over a set of manifolds instead of a single state space. To this end, we consider a set of action classes $A = \{a_1, \ldots, a_{|A|}\}$, where we learn for each class an action-specific low-dimensional manifold $M_a \subset \mathbb{R}^{d_a}$ with $d_a \ll D$. We assume that the following mappings are available:

$$f_a : E_\Theta \mapsto M_a, \quad g_a : M_a \mapsto E_\Theta, \quad h_a : M_a \mapsto M_a,$$  \hspace{1cm} (4.4)

where $f_a$ denotes the mapping from the state space to the low-dimensional manifolds, $g_a$ the projection back to the state space, and $h_a$ the prediction within an action-specific manifold. Since the manifolds encode only the space of joint angles, a low-dimensional representation of the full pose is denoted by $y_a = (r, t, \Theta_a)$ with $\Theta_a = f_a(\Theta)$. A particle $s^i = (y^i_a, a^i)$ stores the corresponding manifold label $a^i$ in addition to the vector $y^i_a = (r^i, t^i, \Theta^i_a)$ and the set of particles is denoted by $S$. Our algorithm operates both in the state space as well as in the manifolds. An overview of the algorithm is given in Figure 4.2.

### 4.5.1 Action-Specific Manifolds

Each of the action-specific low-dimensional manifolds, $M_a$, are learned from the joint angles $\Theta$ in motion capture data using Isomap [Tenenbaum et al., 2000], a non-linear dimensionality reduction technique. As Isomap does not provide mappings between the high- and low-dimensional pose spaces, we learn two separate Gaussian Process (GP) regressions [Rasmussen & Williams, 2005], $f_a$ and $g_a$ (4.5), to map from the high-dimensional space to the low-dimensional space and back, respectively, where $m(\cdot)$ and $k(\cdot)$ denote the mean and covariance functions.

$$y = f_a(x) \sim \mathcal{GP}(m(x), k(x,x')); \quad x = g_a(y) \sim \mathcal{GP}(m(y), k(y,y')).$$  \hspace{1cm} (4.5)

In addition, a third GP regression, $h_a$, is learned to model temporal transitions between successive poses within each action-specific manifold:

$$y_t = h_a(y_{t-1}) \sim \mathcal{GP}(m(y_{t-1}), k(y_{t-1}, y'_{t-1})).$$  \hspace{1cm} (4.6)
4.5.2 Theoretical Discussion

As mentioned in Section 4.5, one seeks the solution of the minimization problem \( \min_{x \in E} V(x) \). When optimizing over a set of manifolds the problem becomes

\[
\min_{a \in A} \left( \min_{y \in M_a} V(g_a(y)) \right). \tag{4.7}
\]

Minimizing the problem this way, i.e. searching the global minimum in all manifolds \( M_a \) and then taking the best solution mapped back to the state space, does not scale well with the number of manifolds. Hence, we propose to optimize over all manifolds jointly. Before outlining the optimization procedure in Section 4.5.3, we briefly discuss the existence and the uniqueness of the solution. Since \( g_a \) and \( f_a \) are not direct inverses of each other, i.e. \( (g_a \circ f_a) \) does not equal the identity function, the optimization over the manifolds (4.7) does not provide the same solution as the original optimization problem over the state space. Indeed, this is the case only if the following is satisfied:

\[
\exists a \in A, \exists y \in M_a : \min_{x \in E} V(x) = V(g_a(y)). \tag{4.8}
\]

The uniqueness of the solution for the manifold and thus of the action \( a \) is interesting from the point of action recognition. It is given if and only if

\[
\forall a_1, a_2 \in A \text{ with } a_1 \neq a_2 : \min_{y \in M_{a_1}} V(g_{a_1}(y)) \neq \min_{y' \in M_{a_2}} V(g_{a_2}(y')). \tag{4.9}
\]

In most cases, optimization of the pose propagates the particles into the “right” manifold, i.e. the correct action, as plotted in Figure 4.3. However, there is usually an overlap of poses between the manifolds such that Eq. (4.9) is not satisfied. Note that in comparison to the action recognition, which takes a sequence of frames into account (Section 4.4), the pose is optimized only for the current frame.

To cope with the problem defined in (4.8), we introduce two optimization steps

\[
\hat{y}, \hat{a} = \arg\min_{a \in A, y \in M_a} V(g_a(y)), \quad \text{and} \quad \hat{x} = \arg\min_{x \in E} V(x), \quad \text{with} \quad x_0 = g_a(\hat{y}) \tag{4.10}
\]

as the initialization. In other words, we first search for the nearest approximation by optimizing over the manifolds and then use this result to initialize the optimization over the state space. With this procedure, we can design an optimization that converges to the global minimum in the state space, see Figure 4.2.

4.5.3 Algorithm

Optimization A: Since ISA [Gall et al., 2008b] is not directly applicable for optimizing over a set of manifolds, we have to modify the algorithm. For the weighting, the particles are mapped back to the full space in order to evaluate the energy function \( V \):
4. Leveraging Actions for Pose Estimation

Figure 4.3: HumanEva. Action recognition prior from camera C1 (a). The curves show the action confidence per frame. Note the smooth transitions between the actions around frame 800 for subject S4. After jogging, the subject walks a few steps before balancing. At the end of the sequence, the person walks away, as recognized by the action recognition system. The distribution of the particles among the action-specific manifolds after Optimization $A$ is shown by the area plot. The particles move to the correct manifold for nearly all frames. Pose estimate for jogging (b) and balancing (c).

$$w^i = \exp \left( -\beta_k \cdot V \left( r^i, t^i, g_{a_i}(\Theta^i) \right) \right),$$

(4.12)

where $k$ is the iteration parameter of the optimization. The weights of all particles are normalized such that $\sum_a w^i = 1$. Note that the normalization does not take the label of the manifold $a^i$ into account. As result, particles in a certain manifold might have higher weights than particles in another manifold since their poses fit the image data better. Since particles with higher weights are more likely to be selected, the distribution of the particles among the manifolds $\mathbb{M}_a$ changes after the selection step. This is desirable since the particles should migrate to the most likely manifold to get a better estimate within this manifold. While the selection is performed as in [Gall et al., 2008b], the mutation step needs to be adapted since the particles are spread in different spaces. To this end, we use $|A|$ mutation kernels $K_a$, one for each manifold, and an additional kernel $K_0$ for the global position and orientation. In our implementation, we use Gaussian kernels with covariance matrices $\Sigma_a$ proportional to the sample covariance within a manifold, i.e. $\mathbb{S}_a = \{ s^i \in \mathbb{S} : a^i = a \}$:

$$\Sigma_a = \frac{\alpha \sum_{s^i \in \mathbb{S}_a} \rho I + \sum_{s^i \in \mathbb{S}_a} (\Theta^i - \mu_a) (\Theta^i - \mu_a)^T}{|\mathbb{S}_a| - 1}, \quad \mu_a = \frac{1}{|\mathbb{S}_a|} \sum_{s^i \in \mathbb{S}_a} \Theta^i,$$

(4.13)

---

1Using the selection kernel $\epsilon_k(\eta) = \frac{1}{\inf \{ y : \eta(\{ x \in \mathbb{E} : \exp(-\beta_k V(x)) > y \}) = 0 \}}$. 


The scaling factor $\alpha_\Sigma = 0.4$ and the positive constant $\rho = 0.0001$, which ensures that the covariance does not become singular, are fixed for all kernels. The kernel $K_0$ for rotation and translation is computed over the full set of particles $S$:

$$\Sigma_0 = \frac{\alpha_\Sigma}{|S| - 1} \left( \rho I + \sum_{s' \in S} \left( (r^{s'}, t^{s'}) - \mu \right) \left( (r^{s'}, t^{s'}) - \mu \right)^T \right), \quad \mu = \frac{1}{|S|} \sum_{s' \in S} (r^{s'}, t^{s'}).$$  

(4.14)

Since we compute the extra kernel $K_0$ instead of taking $(r, t)$ as additional dimensions for the kernels $K_a$, the correlation between $(r, t)$ and $\Theta_a$ is not taken into account. However, the number of particles per manifold can be very small, such that $K_0$ computed over all particles provides a better estimate of the correlation between the global pose parameters $(r, t)$.

Select $p_2$: Before continuing with the optimization in the full state, the set of particles $S$ needs to be mapped from the manifolds $M_a$ to $\mathbb{E}$, where the particles build the initial distribution for the next optimization step. However, it can happen that the true pose is not well represented by any of the manifolds. This is typical of transitions from one action to another, which are not modelled in our setting. As we will show in our experiments, it is useful to use the previous estimate $\hat{x}_{t-1}$ to augment the initial particle set. To measure the discrepancy between the last estimated pose and the poses modeled by the manifolds, we compute $\Sigma_{\hat{a}}$ based on the reconstruction error for $\hat{x}_{t-1}$:

$$\hat{a} = \arg\min_{a \in A} \left\| \Theta_{t-1} - g_a(f_a(\Theta_{t-1})) \right\|, \quad \sigma_{\hat{a}, i} = \frac{|\Theta_{t-1} - g_a(f_a(\Theta_{t-1}))|_i}{3}.$$  

(4.15)

We create a new set of particles by sampling from $\mathcal{N}(\Theta_{t-1}, \Sigma_{\hat{a}})$, where $\Sigma_{\hat{a}}$ is the diagonal matrix with $\sigma_{\hat{a}, i}$ as entries. According to the $3\sigma$ rule, this means that nearly all samples are within the distance of the reconstruction error. The selection process between the two particle sets is controlled by the parameter $p_2 \in [0, 1]$. For all $s^i \in S$, we draw $u$ from the uniform distribution $U[0, 1]$. If $u < p_2$, $s^i = (r^i, t^i, \Theta^i)$ is added to the new set; otherwise the particle $(r^i, t^i, \hat{\Theta})$ is added to the set, where $\hat{\Theta}$ is sampled from $\mathcal{N}(\hat{\Theta}_{t-1}, \Sigma_{\hat{a}})$.

Optimization $B$: The second optimization step eventually runs ISA [Gall et al., 2008b] on the full state space. However, we do not start from the beginning but continue with the optimization, i.e. when $I_{t_A}$ is the number of iterations used for Opt. $A$, we continue with $\beta_{I_{t_A} + 1}$ instead of $\beta_1$.

Select $p_1$: After Opt. $A$, all the particles may aggregate into one single manifold, so we distribute the particles again amongst the manifolds $M_a$ when moving to the next frame $I_t$; otherwise, we get stuck in a single action class. Similar to the previous selection, we
4. Leveraging Actions for Pose Estimation

Figure 4.4: Evaluation of parameters. (a) Select \( p_1 \): The best result is obtained by \( p_1 = 0.5 \), which shows the benefit of taking both particle sets \( S^M \) and \( S^E \) into account. For \( p_1 = 1 \), the particles \( S^E \) from Opt. B are discarded. (b) Select \( p_2 \): The best results are achieved with \( p_2 \in [0.25, 0.5] \). It shows the benefit of taking the reconstruction error for \( \hat{x}_{t-1} \) into account. (c) Number of iterations for Opt. A (It_{A}) and Opt. B (15-It_{A}). The summed number of iterations was fixed to 15. Without a second optimization step (It_{A}=15), the error is significantly higher than for the optimal setting (It_{A}=5).

make use of two particle sets; the particles \( S^M \) in the manifolds \( M_{a} \) after Opt. A and the particles in the state space \( S^E \) after Opt. B. The selection is controlled by the parameter \( p_1 \in [0, 1] \). For all \( s^i \in S^M \), we draw \( u \) from the uniform distribution \( U[0, 1] \). If \( u < p_1 \), \( s^i \) is added to the new set; otherwise the particle \( (r^i, t^i, \Theta^i) \in S^E \) is mapped to one of the manifolds and added to the set. The manifold \( M_{a^i} \) is selected according to the probability \( p(A = a | T = t, I) \), yielding the mapped particle \( (r^i, t^i, f_{a^i}(\Theta^i), a^i) \). In our experiments, we use two choices for \( p(A | T = t, I) \):

\[
\begin{align*}
p(A = a | T = t, I) &= p(A = a) = \frac{1}{|A|} \quad \text{(Uniform Prior)} \\
p(A = a | T = t, I) &= p(A = a | I_{t-l} \ldots I_{t+l}) \quad \text{(Action Prior)}
\end{align*}
\]

The uniform prior is the current frame and results in a joint optimization over the manifolds \( M_{a \in A} \) and poses \( y \in M_{a} \). However, the prior does not scale well with the number of manifolds since the total number of particles is fixed and there must be a sufficient number of particles available for each manifold. The action prior distributes the particles to manifolds that are more likely a-priori, meaning that a manifold \( M_{a} \) cannot be explored when \( p(A = a | T = t, I) = 0 \) and \( \{s^i \in S^M : a^i = a\} = \emptyset \). This also motivates the use of the particle set \( S^M \) to increase the robustness to temporary errors in the action prior as demonstrated in Figure 4.4(a). Note that a zero-probability error for the true manifold over many frames cannot be compensated. In our experiments, \( p(A | I_{t-l} \ldots I_{t+l}) \) is obtained by an action recognition system which takes a set of frames in the neighborhood of \( t \) into account (Section 4.4).
4.6 Experiments

4.6.1 Datasets

**HumanEva-II** The HumanEva-II [Sigal et al., 2010] dataset is a standard benchmark on 3D human pose estimation. It contains two sequences, one for each subject, S2 and S4; each sequence has three actions (see Figure 4.3). The dataset provides a model for subject S4, which we also use for subject S2 despite differences in body shape. The human pose is represented by 28 parameters (15 joints, \( D = 22 \)). We perform two trials: training on S2 and testing on S4 and vice versa. For learning the action-specific manifolds, we use the tracking results of the multi-layer tracker [Gall et al., 2010a]\(^2\). We split the data into the three action classes and discard the transitions between the actions. Note that training data from markerless tracking approaches is in general noisier and less accurate than data from marker-based systems.

**TUM Kitchen** The TUM Kitchen dataset [Tenorth et al., 2009] is a more challenging dataset than HumanEva-II. The dataset contains 20 episodes of recordings from 4 views of 4 subjects setting a table. In each episode, a subject moves back and forth between the kitchen and a dining table, each time fetching objects such as cutlery, plates and cups and then transporting them to the table. The dataset is particularly challenging for both action recognition as well as pose estimation, as the actions are more subtle than standard action recognition benchmarks such as KTH [Schuldt et al., 2004] and Weizmann [Blank et al., 2005] and parts of the body are often occluded by objects such as drawers, cupboard doors and tables (see Figure 4.1). Sample images of the actions can be seen in Figure 4.9. The pose is represented by provided model with 84 parameters (27 joints\(^3\), \( D = 78 \)).

Testing was done on episodes 0-2, 0-4, 0-6, 0-8, 0-10, 0-11, and 1-6. We use two sets of training data, a full set (i.e. all episodes in the dataset except those used for testing) as well as a limited set on episodes 1-0 to 1-5, recorded only from subject 1, to test generalization capabilities of the framework. For the action recognition, we use the 9 labels that are annotated for the ‘left hand’ [Tenorth et al., 2009]. Since these labels are determined by the activity of the arms and we would like the manifolds to be representative of the entire body, we further split the idle/carry class according to whether the subject is walking or standing.

\(^2\)We have used tracking results to create the training data since the motion capture data for HumanEva II is withheld for evaluation purposes.

\(^3\)The original model has 28 joints but we do not consider the gaze since it has zero DOF.
4.6.2 Action Recognition

**HumanEva-II** We do not quantitatively evaluate action recognition on the HumanEva-II sequences as the actions are very simple and the system correctly identifies each of the actions. Examples of the action confidences from camera C1 are shown in Figure 4.3(a).

**TUM Kitchen** For each camera view, we trained a forest of 15 trees of depth 17 each with 50000 random tests generated at all the nodes. Results of the appearance-based action recognition for each individual camera view and for the two different training sets are shown in Table 4.6.2. We report here the classification rate from a frame-by-frame basis, averaged over the different action classes. For each sequence, we disregard a time window of 4 frames on either side of a transition from one action to another. Action recognition performance does not vary much from camera to camera, though there is a significant variation between subjects, i.e. for both training sets, S3 and S4 are easier to classify than S1 and S2.

For classifier fusion, we used the max-rule, which gave the best performance in comparison to other standard ensemble methods [Kittler et al., 1998]. The classifier fusion has a greater effect on the S1 training set (increased performances of up to 21%) than on the full training set (increased performances of up to 13%), so that even with lower average performance on the individual cameras, the fused performance is still equal (0.71).

We show a confusion matrix of the fused results for the S1 training set in Figure 4.5(a); results for the full training set are similar in trend. The most difficult actions to identify are “idle/carry (still)”, which is often confused with “release grasp” and “take object”,

<table>
<thead>
<tr>
<th></th>
<th>S1 training set</th>
<th>full training set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C1</td>
<td>C2</td>
</tr>
<tr>
<td>S1</td>
<td>0.60</td>
<td>0.54</td>
</tr>
<tr>
<td>S2</td>
<td>0.58</td>
<td>0.55</td>
</tr>
<tr>
<td>S3</td>
<td>0.72</td>
<td>0.75</td>
</tr>
<tr>
<td>S4</td>
<td>0.68</td>
<td>0.57</td>
</tr>
<tr>
<td>average</td>
<td>0.65</td>
<td>0.60</td>
</tr>
</tbody>
</table>

*Table 4.1: Individual camera and fused action recognition performance for subjects 1-4; fused performance is higher than any individual camera view for each subject. While the average performance from each camera view using the S1 training set is less than using the full training set, fusing each of the camera views makes up for the difference, such that the average fused performance is equal.*

*Note that these results are slightly higher than those presented in Section 3.5.5 of Chapter 3 due to the exclusion of transition frames.
Figure 4.5: Confusion matrices for fused action outputs of 2D appearance-based action recognition using (a) the S1 training set and (b) the full training set. Average performance over all classes is 0.71 for both training sets.
which is often confused with “open cupboard” and “close cupboard”. In particular, “take object” and “release grasp” are transition actions; such high-level movements may be difficult to define based on low-level features alone.

4.6.3 3D Pose Estimation

HumanEva-II  For evaluating the pose estimation, we measure the absolute 3D error of the estimated joint positions and report for the sequences S2 and S4 the mean error and standard deviation over frames. For comparison, we report the results for optimizing over the state space $\mathbb{E}$ (baseline), and the proposed algorithm with a uniform prior and an action prior, where the action prior is computed as described in Section 4.4. For evaluation, we use 5 iterations for Optimization A, and 10 iterations for Optimization B unless otherwise specified. For the baseline, we run ISA with 15 iterations. Sample pose estimates using the action prior are shown in Figure 4.3(b) and (c).

According to [Gall et al., 2010a; Sigal et al., 2010], pose estimation requires usually at least 200-250 particles to achieve good results on this dataset. We perform the optimization of the 28 parameters with 200 down to 25 particles as plotted in Figure 4.6. Unsurprisingly, the error for the baseline increases significantly when the number of particles drops below 100. When optimizing over the manifolds and the poses embedded in the manifolds, the error increases gently with a decreasing number of particles. Since the dataset contains only 3 action classes, the uniform prior performs very well and differences between the two priors become prominent only when using very few particles per action class. This indicates that the action prior scales better with a large number of classes since this basically limits the number of particles per action class. In general, the uniform prior describes the scenario where the action recognition is not better than a random guess. Timings and mean errors are given in Table 4.5. It is interesting to note that the errors for subject S2 is either comparable or even lower than for S4, suggesting that having a perfect body model is not essential to achieve reasonable pose estimates.

In Figure 4.4, we plot the impact of the parameters on the tracking accuracy. The results clearly support our design decisions for the algorithm (Section 4.5.3).

In Figure 4.7 and Table 4.2, we show the tracking performance with respect to number of camera views using 200 particles. Again, the proposed approach significantly outperforms the baseline. At first glance, the uniform prior and the action prior seem to perform similarly, due to the scaling of the plot from the large error of the baseline, though the action prior actually reduces the error on average by 4%. The benefit of the action prior

---

5“Take object” always occurs between “reach” and “idle/carry” while “release grasp” always occurs before “idle/carry”, after interacting with an object, the drawer or the cupboard.

6The worst-case scenario would be if the action recognition is biased and always misclassifies certain actions as others.
4.6. Experiments

Figure 4.6: 3D estimation error with respect to number of particles. The proposed approach performs significantly better than the direct optimization in the state space $\mathbb{E}$ (baseline), particularly for a small number of particles. The discrepancy between uniform prior and the prior obtained from 2D action recognition is getting larger for very few particles. In this case, the number of particles per manifold becomes very small for a uniform distribution. Note that competitive results are still achieved with only 25 particles. Timings are given in Table 4.5.

Figure 4.7: 3D Estimation error with respect to number of views for HumanEva-II. For the setting with two views, cameras C1 and C2 are taken. The reduced number of views results in more ambiguities. The proposed approach handles these ambiguities better than the direct optimization in the state space $\mathbb{E}$ (baseline). The mean errors are also given in Table 4.2.
Table 4.2: 3D estimation error (mean ± standard deviation over frames) of the optimization with respect to number of views (camera). ap: action prior; up: uniform prior.

<table>
<thead>
<tr>
<th>seq.</th>
<th>cameras</th>
<th>ap (mm)</th>
<th>up (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>C1-C2</td>
<td>54.6 ± 20.8</td>
<td>54.7 ± 21.5</td>
</tr>
<tr>
<td>S2</td>
<td>C1-C4</td>
<td>44.9 ± 9.5</td>
<td>49.4 ± 19.0</td>
</tr>
<tr>
<td>S4</td>
<td>C1-C2</td>
<td>56.9 ± 29.0</td>
<td>60.9 ± 32.5</td>
</tr>
<tr>
<td>S4</td>
<td>C1-C4</td>
<td>45.2 ± 13.4</td>
<td>45.2 ± 11.8</td>
</tr>
</tbody>
</table>

Table 4.3: Impact of smoothing.

<table>
<thead>
<tr>
<th>method</th>
<th>frames</th>
<th>time</th>
<th>cam</th>
<th>error(S2)</th>
<th>error(S4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Baak et al., 2009]</td>
<td>28</td>
<td>4</td>
<td>-</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>[Andriluka et al., 2010]</td>
<td>-350</td>
<td>28</td>
<td>1</td>
<td>101</td>
<td>-</td>
</tr>
<tr>
<td>[Sigal et al., 2010]</td>
<td>250</td>
<td>4</td>
<td>83</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>[Peursum et al., 2010]</td>
<td>-380</td>
<td>36</td>
<td>4</td>
<td>107</td>
<td>92</td>
</tr>
<tr>
<td>[Gall et al., 2010a]</td>
<td>124</td>
<td>4</td>
<td>38</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>[Bergtholdt et al., 2010]</td>
<td>20th</td>
<td>-</td>
<td>4</td>
<td>207</td>
<td>292</td>
</tr>
<tr>
<td>[Brubaker et al., 2010]</td>
<td>-350</td>
<td>-</td>
<td>2</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>[Corazza et al., 2010]</td>
<td>-150</td>
<td>-</td>
<td>4</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td>[Schmaltz et al., 2011]</td>
<td>15</td>
<td>4</td>
<td>-</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>[Gall et al., 2008a]</td>
<td>-400</td>
<td>30</td>
<td>4</td>
<td>-</td>
<td>36</td>
</tr>
<tr>
<td>[Gall et al., 2009]</td>
<td>9</td>
<td>4</td>
<td>-</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>[Darby et al., 2010]</td>
<td>15-20</td>
<td>2-3</td>
<td>97</td>
<td>93</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Comparison to state-of-the-art for HumanEva-II. Note that the methods are often not directly comparable since they rely on different assumptions. For instance, several methods have been applied only to a subset of frames (frames), e.g. only the walking activity (up to frame 350) or only every 20th frame. The number of cameras (cam) also varies. The error (error) is the average 3D error in mm and the approximate computational time (time) is measured in seconds per frame.

is more evident when less particles are used, as shown in Figure 4.6, since this results in even fewer particles being distributed to each action class.
4.6. Experiments

We also evaluated the impact of smoothing the estimated joint estimates over time. Since the pose estimates are obtained by computing the mean of a high-dimensional distribution approximated by as few as 200 particles, the estimates are very noisy. Therefore, we filtered the 3D joint positions with a lowpass filter. In our experiments, we processed the data by 3 passes of the moving average with span of 5 frames. As the results in Table 4.3 show, the smoothing reduces the average error by about 4%-6%.

In Table 4.4, we compare our approach to state-of-the-art methods that reported results for HumanEva-II. Although the methods are often not directly comparable since they rely on different assumptions, the results show that the proposed method achieves state-of-the-art performance with respect to accuracy and runtime. Even though the multi-layer framework [Gall et al., 2010a] achieves a higher accuracy on the full dataset, the approach is much slower (124 seconds per frame) than the proposed method (4 seconds per frame) since it uses more expensive image features and a second layer for segmentation-based pose refinement.

**TUM Kitchen** Based on the fused results of the action recognition, we also evaluate the tracking performance. For the dataset, we use the provided models with 84 parameters. The large errors for the baseline in Figure 4.8 show that 200 particles are not enough to optimize over a 84 dimensional search space. Note that we do not make use of any joint limits or geometric information about the kitchen and use only the images as input. The proposed approach estimates the sequences with an accuracy comparable to HumanEva-II, although the dimensions of the state space increased from 28 to 84, the number of action classes from 3 to 8 (the ‘open’ and ‘close’ actions are embedded in one manifold), and the silhouette quality is much worse due to truncations and occlusions. Compared to the uniform prior, the action prior reduces the error in average by 9%-11% depending on the different training setups.

The detailed results in Tables 4.6 and 4.7 show that the smoothing reduces the error by 7%-8% for the uniform prior as well as the action prior. Since the training data may influence not only the action prior but also the learned manifolds \( \mathcal{M}_a \), we evaluated the method for both training sets. Even when the system is trained only on one subject (S1 training set), the human poses are well estimated; showing that the method generalises well across subjects.

Using the full training set, we also evaluated the pose estimation error with 300 particles and provide the results in Table 4.8. Similar to HumanEva-II, the differences between action prior and uniform prior become marginal with an increasing number of particles (see Figure 4.6). Increasing the particles from 200 to 300 reduces the error by 11% for the uniform prior, whereas the error is only reduced by 2%-3% for the action prior. The error reduction is independent of the smoothing that reduces the error in average by 8% in both cases, as with 200 particles. This shows that the action prior is only beneficial when the number of particles per manifold is very small. Otherwise, the pose estimation
Figure 4.8: 3D Error for TUM kitchen dataset using two different training sets. While the proposed approach performs significantly better than the direct optimization in the state space $E$ (baseline), the impact of the different training sets is small. Mean and standard deviation are provided in Tables 4.6 and 4.7.
4.6. EXPERIMENTS

Ap: action prior; up: uniform prior; base: baseline.

<table>
<thead>
<tr>
<th></th>
<th>base+smooth</th>
<th>up+smooth</th>
<th>ap+smooth</th>
<th>base</th>
<th>up</th>
<th>ap</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Error (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TUM kitchen dataset</strong> (using S1 training set)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (sec.)</td>
<td>1.6 ± 0.7</td>
<td>1.0 ± 0.8</td>
<td>0.6 ± 0.4</td>
<td>0.4 ± 0.2</td>
<td>0.2 ± 0.1</td>
<td></td>
</tr>
<tr>
<td>S4 Error (mm)</td>
<td>114 ± 49.0</td>
<td>178 ± 70.9</td>
<td>174 ± 67.0</td>
<td>183 ± 76.0</td>
<td>155 ± 70.9</td>
<td></td>
</tr>
<tr>
<td>S2 Error (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Computation time per frame and 3D estimation error (mean ± standard deviation over frames)** of the optimization with reference to number of particles. The 2D action recognition takes additional 0.4 seconds for each frame consisting of 4 images, which is

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>np</th>
<th>ap</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Error (mm)</td>
<td>69.3 ± 31.1</td>
<td>100.5 ± 40.4</td>
<td>94.9 ± 39.5</td>
</tr>
<tr>
<td>Time (sec.)</td>
<td>96.0 ± 92.7</td>
<td>100.0 ± 98.3</td>
<td>98.0 ± 91.4</td>
</tr>
</tbody>
</table>

Table 4.5: Computation time for 20 particles. Ap: action prior; up: uniform prior; base: baseline.
<table>
<thead>
<tr>
<th></th>
<th>0-2</th>
<th>0-4</th>
<th>0-6</th>
<th>0-8</th>
<th>0-10</th>
<th>0-11</th>
<th>1-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>48.1 ± 22.4</td>
<td>58.4 ± 27.1</td>
<td>64.6 ± 30.4</td>
<td>45.8 ± 27.8</td>
<td>69.3 ± 39.9</td>
<td>68.7 ± 31.6</td>
<td>75.9 ± 36.5</td>
</tr>
<tr>
<td>up</td>
<td>50.3 ± 25.4</td>
<td>57.2 ± 25.5</td>
<td>61.9 ± 27.4</td>
<td>49.0 ± 25.7</td>
<td>67.2 ± 36.9</td>
<td>167.1 ± 114.0</td>
<td>78.6 ± 40.4</td>
</tr>
<tr>
<td>base</td>
<td>116.5 ± 45.1</td>
<td>181.9 ± 70.6</td>
<td>174.8 ± 61.2</td>
<td>183.0 ± 61.4</td>
<td>229.4 ± 85.0</td>
<td>190.6 ± 65.0</td>
<td>155.4 ± 70.4</td>
</tr>
<tr>
<td>ap+smooth</td>
<td>43.5 ± 21.3</td>
<td>53.2 ± 25.7</td>
<td>59.2 ± 28.8</td>
<td>41.0 ± 26.7</td>
<td>63.8 ± 38.5</td>
<td>64.8 ± 30.8</td>
<td>71.6 ± 35.2</td>
</tr>
<tr>
<td>up+smooth</td>
<td>45.6 ± 24.3</td>
<td>51.9 ± 22.9</td>
<td>56.7 ± 25.9</td>
<td>43.7 ± 23.6</td>
<td>61.5 ± 35.0</td>
<td>161.3 ± 109.7</td>
<td>74.5 ± 39.3</td>
</tr>
<tr>
<td>base+smooth</td>
<td>114.3 ± 45.0</td>
<td>179.3 ± 70.7</td>
<td>172.1 ± 61.1</td>
<td>180.3 ± 61.4</td>
<td>227.4 ± 85.1</td>
<td>188.4 ± 64.7</td>
<td>153.3 ± 70.7</td>
</tr>
</tbody>
</table>

Table 4.7: 3D Error for TUM kitchen dataset in mm (using full training set). ap: action prior; up: uniform prior; base: baseline.

<table>
<thead>
<tr>
<th></th>
<th>0-2</th>
<th>0-4</th>
<th>0-6</th>
<th>0-8</th>
<th>0-10</th>
<th>0-11</th>
<th>1-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>46.9 ± 23.0</td>
<td>57.4 ± 28.6</td>
<td>59.5 ± 27.3</td>
<td>49.2 ± 33.8</td>
<td>63.2 ± 36.0</td>
<td>66.6 ± 32.2</td>
<td>74.3 ± 37.0</td>
</tr>
<tr>
<td>up</td>
<td>47.5 ± 24.0</td>
<td>56.2 ± 23.4</td>
<td>61.9 ± 31.6</td>
<td>47.8 ± 35.0</td>
<td>65.7 ± 41.7</td>
<td>67.4 ± 33.0</td>
<td>72.1 ± 34.6</td>
</tr>
<tr>
<td>ap+smooth</td>
<td>42.4 ± 22.1</td>
<td>52.4 ± 26.1</td>
<td>54.4 ± 26.1</td>
<td>44.6 ± 33.1</td>
<td>58.1 ± 35.2</td>
<td>63.1 ± 31.0</td>
<td>70.4 ± 35.9</td>
</tr>
<tr>
<td>up+smooth</td>
<td>43.0 ± 22.7</td>
<td>51.1 ± 21.3</td>
<td>56.5 ± 27.4</td>
<td>43.1 ± 34.3</td>
<td>60.4 ± 40.0</td>
<td>63.5 ± 31.9</td>
<td>68.1 ± 33.0</td>
</tr>
</tbody>
</table>

Table 4.8: 3D Error for TUM kitchen dataset in mm (using full training set and 300 particles). ap: action prior; up: uniform prior; base: baseline.
Table 4.9: Fraction of particles which fall into the “correct“ action manifold, i.e. that which corresponds to the action label annotation; values are much lower than that of the appearance-based classifier (see Table 4.6.2 but still higher than chance. There is little difference between the type of prior and number of particles used. ap: action prior; up: uniform prior

<table>
<thead>
<tr>
<th></th>
<th>(200 particles)</th>
<th>(300 particles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>up</td>
<td>ap</td>
</tr>
<tr>
<td>S1</td>
<td>0.23</td>
<td>0.25</td>
</tr>
<tr>
<td>S2</td>
<td>0.34</td>
<td>0.33</td>
</tr>
<tr>
<td>S3</td>
<td>0.28</td>
<td>0.27</td>
</tr>
<tr>
<td>S4</td>
<td>0.37</td>
<td>0.35</td>
</tr>
<tr>
<td>average</td>
<td>0.31</td>
<td>0.30</td>
</tr>
</tbody>
</table>

efficiently allocates most of the particles to the relevant manifolds and achieves accurate pose estimates even with a uniform prior.

At the end of Optimization A (see Section 4.5.3), the set of particles $S$ are associated with one of the manifolds $M_a$. The distribution of the particles amongst the set of manifolds serves as a complimentary cue to indicate the action suggested by the pose. We can obtain an action label by taking the manifold with the most number of particles after resampling. When the actions are simple, such as in HumanEva-II, the particles most often migrate to the “correct” manifold, i.e. the manifold with the action matching to the action label annotation (see Figure 4.3). We look at the distribution of the particles for the TUM kitchen case for both the action prior and the uniform prior for the full training set and summarize the subject specific results in Table 4.9.

In general, the classification of actions is much worse than the appearance-based classifier and having more particles (i.e. 300 versus 200) does not help much. However, the classification is still much higher than chance. Note that the appearance-based classifier takes sequences of frames into account, while the pose is optimized and associated with a manifold for only a single frame. Since a unique solution of the optimal manifold does not exist (4.9), the particles might end up in any of the manifolds that contain the right pose.

4.7 Conclusion

We have presented an algorithm that efficiently solves the problem of optimizing over a set of manifolds. In the context of 3D pose estimation, we demonstrated that the algorith...
Figure 4.9: Cropped images and pose estimates for the actions of TUM Kitchen Dataset, shown for cameras 1 and 3; Close cupboard and close drawer are not shown.
rithm handles high-dimensional spaces with very few particles. Since transitions between actions are not explicitly modeled, as in previous work, it is an important step towards pose estimation with many action classes. Furthermore, we have shown that a prior distribution based on action recognition improves the performance. This is interesting since it is expected that the algorithm scales very well with the number of classes when the action recognition system does as well. In this way, 3D human pose estimation can be linked to the progress in the field of action recognition. As there are very few datasets for pose estimation and action recognition available and none contains many action classes, new datasets are required to investigate scalability more in detail. The work also shows the importance of a taxonomy of actions. For instance, we discovered that it is better to embed ‘opening’ and ‘closing’ in a single manifold even though the actions have different labels.
Pose Estimation of Complex Activities

5.1 Introduction

Tracking human 3D articulated motions from video sequences is well known to be a challenging machine vision problem. Estimating the human body’s 3D location and orientation of the joints is notoriously difficult because it is a high-dimensional problem and is riddled with ambiguities coming from noise, monocular imagery and occlusions. To reduce the complexity of the task, it has become very popular to use prior models of human pose and dynamics [Sidenbladh et al., 2000; Urtasun et al., 2006; Wang et al., 2008; Xu & Li, 2007; Jäggli et al., 2009; Li et al., 2010; Taylor et al., 2010].

Linear models (e.g. PCA) are among the simplest priors [Sidenbladh et al., 2000;Ormoneit et al., 2001; Urtasun et al., 2005], though linearity also restricts a model’s expressiveness and results in inaccuracies when learning complex motions. Priors generated from non-linear dimensionality reduction techniques such as Isomap [Tenenbaum et al., 2000] and LLE [Roweis & Saul, 2000] have also been used for tracking [Gall et al., 2010b; Jäggli et al., 2009]. These techniques try to preserve the local structure of the manifold but tend to fail when manifold assumptions are violated, e.g., in the presence of noise, or multiple activities. Moreover, LLE and Isomap provide neither a probability distribution over the space of possible poses nor a mapping from the latent space to the high dimensional space. While such a distribution and or mapping can be learned post hoc, learning them separately from the latent space typically results in suboptimal solutions.

Probabilistic latent variable models (e.g. probabilistic PCA), have the advantage of taking uncertainties into account when learning latent representations. Taylor et al. [Taylor et al., 2010] introduced the use of Conditional Restricted Boltzmann Machines (CRBM) and implicit mixtures of CRBM (imCRBM), which are composed of large collections of discrete latent variables. Unfortunately, learning this type of model is a highly complex task. A more commonly used latent variable model is the Gaussian Process Latent Variable Model (GPLVM) [Lawrence, 2005] which has been applied to animation [Wang et al., 2008] and tracking [Urtasun et al., 2005, 2006; Geiger et al., 2009; Hou et al.,]
5. Pose Estimation of Complex Activities

While the GPLVM is very successful at modeling small training sets with single activities, it often struggles to learn latent spaces from larger datasets, especially those with multiple activities. The main reason is that the GPLVM is a non-parametric model; learning requires the optimization of a non-convex function, for which complexity grows with the number of training samples. As such, having a good initialization is key for success [Lawrence, 2005], though good initializations are not always available [Geiger et al., 2009], especially with complex data. Additionally, GPLVM learning scales cubicly with the number of training examples, and application to large datasets is computationally intractable, making it necessary to use sparsification techniques to approximate learning [Quinonero-Candela & Rasmussen, 2006; Lawrence, 2007]. As a consequence, the GPLVM has been mainly applied to single activities, e.g., walking or running.

More recent works have focused on handling multiple activities, most often with mixture models [Lin et al., 2006; Li et al., 2007, 2010] or switching models [Pavlovic et al., 2000; Jäggi et al., 2009; Chen et al., 2009]. However, coordinating the different components of the mixture models requires special care to ensure that they are aligned in the latent

Figure 5.1: Representative poses, data (Euclidean) distance matrices and learned latent spaces from walking, jumping, exercise stretching and basketball signal sequences. GPLVM was initialized using probabilistic PCA; while stochastic GPLVM was initialized randomly.
space [Roweis et al., 2002], thereby complicating the learning process. In addition, both mixture and switching models require a discrete notion of activity which is not always available, e.g. dancing motions are not a discrete set. Others have tried to couple discriminate action classifiers with action-specific models [Baak et al., 2009; Gall et al., 2010b], though accuracy of such systems does not scale well with the number of actions.

A good prior model for tracking should be accurate, expressive enough to capture a wide range of human poses, and easy and tractable for both learning and inference. Unfortunately, none of the aforementioned approaches exhibit all of these properties. In this paper, we are interested in learning a probabilistic model that fulfill all of these criteria. Towards this end, we propose a stochastic gradient descent algorithm for the GPLVM which can learn latent spaces from random initializations. We draw inspiration for our work from two main sources. The first, [Urtasun & Darrell, 2008], approximates Gaussian process regression for large training sets by doing online predictions based on local neighbourhoods. The second, [Lawrence & Urtasun, 2009], maximizes the likelihood function for GPLVM by considering one dimension of the gradient at a time in the context of collaborative filtering. Based on these two works, we propose a similar strategy to approximate the gradient computation within each step of the stochastic gradient descent algorithm. Local estimation of the gradients allows our approach to efficiently learn models from large and complex training sets while mitigating the problem of local minima. Furthermore, we propose an online algorithm that can effectively learn latent spaces incrementally without extensive relearning. We demonstrate the effectiveness of our approach on the task of monocular and multi-view tracking and show that our approach outperforms the state-of-the-art on the standard benchmark HumanEva [Sigal et al., 2010].

5.2 Stochastic learning

We first review the GPLVM, the basis of our work, and then introduce our optimization method for learning with stochastic local updates. Finally, we derive an extension of the algorithm which can be applied to the online setting.

5.2.1 GPLVM Review

The GPLVM assumes that the observed data has been generated by some unobserved latent random variables. More formally, let $Y = [y_1, \ldots, y_N]^T$ be the set of observations $y_i \in \mathbb{R}^D$, and $X = [x_1, \ldots, x_N]^T$ be the set of latent variables $x_i \in \mathbb{R}^Q$, with $Q \ll D$. The GPLVM relates the latent variables and the observations via the probabilistic mapping $y^{(d)} = f(x) + \eta$, with $\eta$ being i.i.d. Gaussian noise, and $y^{(d)}$ the $d$-th coordinate of the observations. In particular, the GPLVM places a Gaussian process prior over the mapping
6. Pose Estimation of Complex Activities

Let $f$ such that marginalization of the mapping can be done in closed form. The resulting conditional distribution becomes

$$p(Y | X, \beta) = \frac{1}{\sqrt{(2\pi)^N |K|}} \exp\left(-\frac{1}{2} tr \left(K^{-1} YY^T\right)\right),$$  \hspace{1cm} (5.1)

where $K$ is the kernel matrix with elements $K_{ij} = k(x_i, x_j)$ and the kernel $k$ has parameters $\beta$. Here, we follow existing approaches [Urtasun et al., 2005, 2006] and use a a kernel compounded from an RBF, a bias, and Gaussian noise, i.e., $k(x, x') = \beta_1 \exp\left(-\frac{\|x-x'\|^2}{\beta_2}\right) + \beta_3 + \delta_{x,x'} \beta_4$.

The GPLVM is usually learned by maximum likelihood estimation of the latent coordinates $X$ and the kernel hyperparameters $\beta = \{\beta_1, \cdots, \beta_4\}$. This is equivalent to minimizing the negative log likelihood $L$:

$$L = -\ln p(Y | X, \beta) = -\frac{D N}{2} \ln 2\pi - \frac{D}{2} \ln |K| - \frac{1}{2} tr \left(K^{-1} YY^T\right).$$  \hspace{1cm} (5.2)

Typically a gradient descent algorithm is used for the minimization. The gradient of $L$ with respect to $X$ can be obtained via the chain rule, where

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial K} \cdot \frac{\partial K}{\partial X} = -\left(K^{-1} YY^T K^{-1} - DK^{-1}\right) \cdot \frac{\partial K}{\partial X}. \hspace{1cm} (5.3)$$

Similarly, the gradient of $L$ with respect to $\beta$ can be found by substituting $\frac{\partial K}{\partial X}$ with $\frac{\partial K}{\partial \beta}$ in Eq. (5.3) (see [Lawrence, 2005] for the exact derivation). As $N$ gets large, however, computing the gradients becomes computationally expensive, because inverting $K$ is of $O(N^3)$, with $N$ the number of training examples. More importantly, as the negative log likelihood $L$ is highly non-convex, especially with respect to $X$, standard gradient descent approaches tend to get stuck in local minima, and rely on having good initializations for success.

We now demonstrate how a stochastic gradient descent approach can be used to reduce computational complexity as well as decrease the chances of getting trapped in local minima. In particular, as shown in our experiments (Section 5.3), we are able to obtain smooth and accurate manifolds (see Figure 5.1) from random initialization.

### 5.2.2 Stochastic Gradient Descent

In standard gradient descent, all points are taken into account at the same time when computing the gradient; stochastic gradient descent approaches, on the other hand, approximate the gradient at each point individually. Typically, a loop goes over the points
in a series or by randomly sampling from the training set. Note that after iterating over all the points, the gradient is exact. As the GPLVM is a non-parametric approach, the gradient computation at each point does not decompose, making it necessary to invert $K$, an $O(N^3)$ operation at every iteration. We propose, however, to approximate the gradient computation within each step of the stochastic gradient descent algorithm. Therefore, the gradient of $L$ can be estimated locally for some neighbourhood of points $X_R$, centered at a reference point $x_r$, rather than over all of $X$. Eq. (5.3) can then be evaluated only for the points within the neighbourhood, i.e.,

$$\frac{\partial L}{\partial X_R} \approx - (K_R^{-1} Y_R Y_R^T K_R^{-1} - D K_R^{-1}) \cdot \frac{\partial K_R}{\partial X_R}, \quad (5.4)$$

where $K_R$ is the kernel matrix for $X_R$ and $Y_R$ is the corresponding neighbourhood data points.

We employ a random strategy for choosing the reference point $x_r$. The neighbourhood $R$ can be determined by any type of distance measure, such as Euclidean distance in the latent space and/or data space, or temporal neighbours when working with time series. More critical than the specific type of distance measure, however, is allowing sufficient coverage of the latent space so that each neighbourhood is not restricted too locally. To keep the complexity low, it is beneficial to sample randomly from a larger set of neighbours (see Section 5.3.1).

The use of stochastic gradient descent has several desirable traits that correct for the aforementioned drawbacks of GPLVMs. First, computational complexity is greatly reduced, making it feasible to learn latent spaces with much larger amounts of data. Secondly, estimating the gradients stochastically and locally improves robustness of the learning process.

---

**Algorithm 1** Stochastic GPLVM

Randomly initialize $X$
Set $\beta$ with an initial guess

**for** $t = 1$ to $T$ **do**

randomly select $x_r$
find $R$ neighbours around $x_r$: $X_R = X \in R$
Compute $\frac{\partial L}{\partial X_R}$ and $\frac{\partial L}{\partial \beta_R}$ (see Eq. (5.3))
Update $X$ and $\beta$:

$$\Delta X_t = \mu_X \cdot \Delta X_{t-1} + \eta_X \cdot \frac{\partial L}{\partial X_R}$$
$$X_t \leftarrow X_{t-1} + \Delta X_t$$

$$\Delta \beta_t = \mu_\beta \cdot \Delta \beta_{t-1} + \eta_\beta \cdot \frac{\partial L}{\partial \beta_R}$$
$$\beta_t \leftarrow \beta_{t-1} + \Delta \beta_t$$

**end for**
process against local minima, making it possible to have a random initialization. An algorithmic summary of stochastic gradient descent learning for GPLVMs is given in Algorithm 1.

### 5.2.3 Incremental Learning

In this section, we derive an incremental learning algorithm based on the stochastic gradient descent approach of the previous section. In this setting, we have an initial model which we would like to update as new data comes in on the fly. More formally, let $Y_{orig}$ be the initial training data, and $X_{orig}$ and $\beta_{orig}$ be a model learned from $Y_{orig}$ using stochastic GPLVM. For every step in the online learning, let $Y_{incr}$ be new data, which can be as little as a single point or an entire set of training points.

Let $Y = [Y_{orig}, Y_{incr}] \in \mathbb{R}^{(N+M) \times D}$ be the set of training points containing both the already trained data $Y_{orig}$, and the new incoming data $Y_{incr}$, and let $X = [X_{orig}, X_{incr}] \in \mathbb{R}^{(N+M) \times Q}$ be the corresponding latent coordinates, where $M$ is the number of newly added training examples. Let $\hat{X}_{orig}$ be the estimate of the latent coordinates that has already been learned.

A possible strategy is to update only the incoming points; however, we would like to exploit the new data for improving the estimate of the entire manifold, therefore we propose to learn the full $X$. To prevent the already-learned manifold from diverging and also to speed up learning, we add a regularizer to the log-likelihood to encourage original points to not deviate too far from their initial estimate. To this end, we use the Frobenius norm of the deviation from the estimate $\hat{X}_{orig}$. Learning is then done by minimizing the regularized negative log-likelihood

$$L_{incr} = L + \frac{\lambda}{N} \|X_{1:N,:} - \hat{X}_{orig}\|_F^2. \quad (5.5)$$

Here, $X_{1:N,:}$ indicates the first $N$ rows of $X$, while $\lambda$ is a weighting on the regularization term. The gradient of $L$ with respect to $X_R$ can then be computed as

$$\frac{\partial L_{incr}}{\partial X_R} = \frac{\partial L}{\partial X_R} + \lambda \cdot \frac{2}{N} \cdot (X_{1:N,:} - \hat{X}_{orig}) \frac{\partial X_{1:N,:}}{\partial X_R}. \quad (5.6)$$

We employ a stochastic gradient descent approach for our incremental learning, where the points are sampled randomly from $X_{incr}$. Note that while $x_r$ is only sampled from $X_{incr}$ in the subsequent learning step, this does not exclude points in $X_{orig}$ from being a part of the neighbourhood $R$, and thus from being updated. We have chosen a nearest neighbour approach by comparing $Y_{incr}$ to $Y_{orig}$ for estimating an initial $X_{incr}$, though other possibilities include performing a grid search in the latent space and selecting locations with

$$\frac{\partial L_{incr}}{\partial \beta_R} = \frac{\partial L}{\partial \beta_R} \quad \text{since the regularization term does not depend on $\beta_R$.}$$
5.2. Stochastic learning

Algorithm 2 Incremental stochastic GPLVM

\begin{algorithm}
\begin{algorithmic}
\FOR {$t = 1$ to $T_1$}
\STATE Learn $X_{\text{orig}}$ and $\beta_{\text{orig}}$ as per Algorithm 1.
\ENDFOR
\STATE Initialize $X_{\text{incr}}$ using nearest neighbours.
\STATE Set $\beta = \beta_{\text{orig}}$
\STATE Group data: $Y = [Y_{\text{orig}}, Y_{\text{incr}}]$ \hfill $X = [X_{\text{orig}}, X_{\text{incr}}]$ \hfill \\
\FOR {$t = T_1$ to $T_2$}
\STATE randomly select $x_r \in X_{\text{incr}}$
\STATE find $R$ neighbours around $x_r$: $X_R = X \in \mathcal{R}$
\STATE Compute $\frac{\partial L_{\text{incr}}}{\partial X_R}$ and $\frac{\partial L_{\text{incr}}}{\partial \beta_R}$ (see Eq. (5.6))
\STATE Update $X$ and $\beta$:
\begin{align*}
\Delta X_t &= \mu_X \cdot \Delta X_{t-1} + \eta_X \cdot \frac{\partial L_{\text{incr}}}{\partial X_R} \\
X_t &\leftarrow X_{t-1} + \Delta X_t \\
\Delta \beta_t &= \mu_{\beta} \cdot \Delta \beta_{t-1} + \eta_{\beta} \cdot \frac{\partial L_{\text{incr}}}{\partial \beta_R} \\
\beta_t &\leftarrow \beta_{t-1} + \Delta \beta_t
\end{align*}
\ENDFOR
\end{algorithmic}
\end{algorithm}

the highest global log-likelihood (Eq. (5.2)) or training a regressor from $Y_{\text{orig}}$ to $X_{\text{orig}}$ to be applied to $Y_{\text{incr}}$. An algorithmic summary of the incremental method is provided in Algorithm 2.

5.2.4 Tracking Framework

During training, a latent variable model $M$ is learned from $Y_M$, where $Y_M$ are relative joint locations with respect to a root node. We designate the learned latent points as $X_M$. During inference, tracking is performed in the latent space using a particle filter. The corresponding pose is computed by projecting back to the data space via the Gaussian process mapping learned in the GPLVM.

We model the state $s$ at time $t$ as $s_t = (x_t, g_t, r_t)$ where $x_t$ denotes position in the latent space, while $g_t$ and $r_t$ are the global position and rotation of the root node.

Particles are initialized in the latent space by a nearest neighbour search between the observed 2D image pose in the first frame of the sequence and the projected 2D poses of $Y_M$. Particles are then propagated from frame to frame using a first-order Markov model

$$
x_{t} = x_{t-1} + \dot{x}_{t}, \quad g_{t} = g_{t-1} + \dot{g}_{t}, \quad r_{t} = r_{t-1} + \dot{r}_{t}.
$$

(5.7)
We approximate the derivative $\dot{x}^i$ with the difference between temporally sequential points of the nearest neighbours in $X_M$, while $\dot{g}^i$ and $\dot{r}^i$ are drawn from individual Gaussians with means and standard deviations estimated from the training data. The tracked latent position $\hat{x}_t$ at time $t$ is then approximated as the mode over all particles in the latent space while $\hat{y}_t$ is estimated via the mean Gaussian process estimate

$$\hat{y}_t = \mu_M + Y^T_M K^{-1} k(\hat{x}_t, X_M),$$

(5.8)

with $\mu_M$ the mean of $Y_M$ and $k(\hat{x}_t, X_M)$ the vector with elements $k(\hat{x}_t, x_m)$ for all $x_m$ in $X_M$. Note that the computation of $K^{-1}$ needs to be performed only once and can be stored.

### 5.3 Experimental Evaluation

We demonstrate the effectiveness of our model when applied to tracking in both monocular and multi-view scenarios. In all cases, the latent models were learned with $\mu_X = 0.8$, $\mu_\beta = 0.5$, $\eta_X = 10^{-4}$, $\eta_\beta = 10^{-8}$; we annealed these parameters over the iterations. To further smooth the learned models, we incorporate a Gaussian Process prior over the dynamics of the training data in the latent space [Wang et al., 2008] for the GPLVM and the stochastic GPLVM.

#### 5.3.1 Neighbourhood Type and Size

We tested three different distance measures (‘xL2’ - Euclidean distance in the latent space, ‘yL2’ - Euclidean distance in the data space and ‘temp’ - temporal neighbours) for determining the neighbourhood $\mathcal{R}$ and compared them to random sampling of points for estimating the gradients $\frac{\partial C}{\partial x_R}$ and $\frac{\partial C}{\partial \beta_R}$. We also compare the effect of neighbourhood size, as determined by the number of points $R$ used to compute the gradients. Finally we look at the effect of randomly subsampling $R$ from a larger selection $\kappa \cdot N$ neighbours. We apply the learning algorithm to 4 motion capture sequences of walking (740 frames total) from a single subject. For the same initialization, we repeat the learning process ten times for the differing neighbourhood types and sizes and compare the minimum, mean and maximum negative-log likelihoods over the repetitions in Figure 5.2.

When we select the nearest $R$ neighbours, learning on the GPLVM is poor; ‘xL2’ outperforms ‘yL2’ and temp, but in all three cases, the gradient estimates are too local to capture the more global shape of the latent space. If we subsample $R$ neighbours from a larger neighbourhood of $\kappa \cdot N$ neighbours, however, we find that the resulting negative-log likelihood is much lower and there is little difference between the different types of distance measures. Randomly sampling $R$ neighbours, on the other hand, while sufficient for estimating the latent space, is not as successful as maintaining some form of neighbourhood.
Figure 5.2: Negative log-likelihood (lower is better) of learning walking motions with different types and sizes of neighbourhoods. Bars show the mean value over 10 runs, while the error bars indicate the maximum and minimum value over runs. The different colored bars indicated differing sizes of \( R \), where \( R \) is given as a fraction of the total number of training samples (\( N = 740 \)). To maintain a fair comparison, we keep the number of iterations for learning, \( T \), inversely proportional to \( R \).

Figure 5.3: Example of a synthesized 2D image from an exercise stretching sequence, with different levels of noise corruption; noise amplitude is measured as a percentage of the data variance.
5.3.2 Monocular Tracking

We compare in the monocular setting the use of PCA, regular GPLVM and our stochastic GPLVM to learn latent spaces from motion capture sequences (from the CMU Motion Capture Database [CMU Mocap Database]). We chose simple single-activity sequences, such as walking (3 subjects, 18 sequences) and jumping (2 subjects, 8 sequences), as well as complex multi-activity sequences, such as stretching exercises (2 subjects, 6 sequences) and basketball refereeing signals (7 subjects, 13 sequences). The stretching exercise and basketball signal sequences were cut to each contain four types of activities. We synthesized 2D data by projecting the mocap from 3D to 2D and then corrupting the location of each joint with different levels of additive Gaussian noise (see Figure 5.3). We then recover the 3D locations of each joint from the noisy images by tracking with the particle filter described in the previous section.

Examples of learned latent spaces for each type of sequence (i.e., walking, jumping, exercise, basketball) are shown in Figure 5.1. We used a neighbourhood of 60 points for the single activity sequences, which have on average 250 training examples, and 100 points for the multiple activity sequences, which have on average 800 training examples. For a sequence of 800 training examples, the stochastic GPLVM takes only 27s to learn (neighbourhood of 100 points, 2500 iterations); in comparison, the regular GPLVM takes 2560s for 312 iterations, while with FITC approximations [Lawrence, 2007] takes on average 1700s (100 active points, 2500 iterations)

In general, as illustrated by Figure 5.1, the manifolds learned with stochastic GPLVM have smoother trajectories than those learned from PCA and GPLVM, with better separation between the activities in the multi-activity sequences.

We evaluate the effectiveness of the learned latent pose models for tracking by comparing the average tracking error per joint per frame between PCA, GPLVM and stochastic GPLVM in two sets of experiments. In the first, training and test sequences are performed by the same subject; in the second, to test generalization properties of the different latent spaces, we train and test on different subjects. We report results average over 10 sequences, each repeated over 10 different runs of the tracker. We use importance sampling and weight each particle at time $t$ proportionally to a likelihood defined by the reprojection error

$$w_t^i \propto \exp \left( -\alpha \sum_j \| p_{j,t}^i - q_{j,t} \|^2 \right),$$

(5.9)

Note that none of the models have completed training. For timing purposes, we take here a fixed number of iterations for the stochastic method and the FITC approximation and the “equivalent” for the regular GPLVM, i.e., 2500 iterations /8, where 8 comes from the fact that 8X more points are used in computing $\mathbf{K}$. 
where \( p_{j,t} \) is the projected 2D position of joint \( j \) in \( y_i^t \) from \( x_i^t \) (see Eq. (5.8)) and \( q_{j,t} \) is the observed 2D position of joint \( j \), assuming that the camera projection and correspondences between joints are already known. \( \alpha \) is a parameter determining selectivity of the weight function (we use \( \alpha = 5 \cdot 10^{-5} \)).

Figs. 5.4 and 5.5 depict 3D tracking error as a function of the amount of Gaussian noise for different number of particles employed in the particle filter for the within- and cross-subject experiments respectively. As expected, tracking error is lower within-subject than cross-subject for all types of latent models. For the simple activities such as walking and jumping, GPLVM generally outperforms PCA, but for the complex activities, it performs only comparably or worse than PCA (with the exception of cross-subject basketball signals). Our stochastic GPLVM, on the other hand, consistently outperforms PCA and matches or outperforms the regular GPLVM in all experimental conditions, with significantly better performance in the complex, multi-activity sequences.

### 5.3.3 Online Tracking

We took two exercise sequences with three different activities from the same subject and apply the online learning algorithm (see Sec. 5.2.3), setting \( \lambda = 2 \). We consider each activity as a new batch of data, and learn the latent space on the first sequence and then track on the second and vice versa. The online algorithm less accurate for tracking than the stochastic GPLVM learned with all data, which is expected since the latent space is biased towards the initial set of activities. We note, however, that the incremental stochastic GPLVM still outperforms the regular GPLVM, as illustrated in Figure 5.6(b). Examples of the learned manifolds are shown in Figure 5.6(a).

### 5.3.4 Multi-view Tracking on HumanEva

We also evaluate our learning algorithm on the HumanEva benchmark [Sigal et al., 2010] on the activities walking and boxing. For all experiments, we use a particle filter as described in Sec. 5.2.4 with 25 particles and an additional annealing component [Deutscher & Reid, 2005] of 15 layers. To maintain consistency with previous works, we use the images from the 3 color cameras and the simple silhouette and edge likelihoods provided in the HumanEva baseline algorithm [Sigal et al., 2010].

**HumanEva-I Walking:** As per [Taylor et al., 2010; Xu & Li, 2007; Li et al., 2010], we track the walking validation sequences of subjects S1, S2, and S3. The latent variable models are learned on the training sequences, being either subject-specific or with all subjects combined. Subject-specific models have \( \sim 1200-2000 \) training examples each, for which we used a neighbourhood of 60 points, while the combined model has \( \sim 4000 \) training examples with a neighbourhood of 150 points. 3D tracking errors averaged over
Figure 5.4: **Within-subject 3D tracking errors** for each type of activity sequence with respect to amount of additive noise for different number of particles, where error bars represent the standard deviation from repetitions runs.

Figure 5.5: **Cross-subject 3D tracking errors** for each type of activity sequence with respect to amount of additive noise for different number of particles, where error bars represent the standard deviation from repetitions runs.
Figure 5.6: (a) Learned manifolds from regular GPLVM, stochastic GPLVM and incremental stochastic GPLVM from an exercise stretching sequence, where blue, red, green indicate jumping jacks, jogging and squats respectively and (b) the associated 3D tracking errors (mm), where error bars indicate standard deviation over repeated runs.

Figure 5.7: Example poses from tracked results on HumanEva.

15 joints as specified in [Sigal et al., 2010] and over all frames in the full sequence are depicted in Table 5.1. Sample frames of the estimated poses are shown in Figure 5.7. In four of the six training/test combinations, the stochastic GPLVM model outperforms the state-of-the-art CRBM and imCRBM model from [Taylor et al., 2010], while in the other
Table 5.1: Comparison of 3D tracking errors (mm) on the entire walking validation sequence with subject-specific models, where ± indicates standard deviation over runs, except for [Li et al., 2010], who reports tracking results for 200 frames of the sequences, with standard deviation over frames.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Tracking Error</th>
<th></th>
<th></th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GPLVM</td>
<td>CRBM</td>
<td>imCRBM</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>-</td>
<td>57.6 ± 11.6</td>
<td>48.8 ± 3.7</td>
<td>58.6 ± 3.9</td>
<td>44.0 ± 1.8</td>
</tr>
<tr>
<td>S1,2,3</td>
<td>S1</td>
<td>140.3</td>
<td>64.3 ± 19.2</td>
<td>55.4 ± 0.8</td>
<td>54.3 ± 0.5</td>
<td>41.6 ± 0.8</td>
</tr>
<tr>
<td>S2</td>
<td>S2</td>
<td>-</td>
<td>68.7 ± 24.7</td>
<td>98.2 ± 15.8</td>
<td>47.4 ± 2.9</td>
<td>47.4 ± 2.9</td>
</tr>
<tr>
<td>S1,2,3</td>
<td>S2</td>
<td>149.4</td>
<td>155.9 ± 48.8</td>
<td>99.1 ± 23.0</td>
<td>69.3 ± 3.3</td>
<td>64.0 ± 2.9</td>
</tr>
<tr>
<td>S3</td>
<td>S3</td>
<td>-</td>
<td>69.6 ± 22.2</td>
<td>71.6 ± 10.0</td>
<td>49.8 ± 2.2</td>
<td>31.4 ± 0.9</td>
</tr>
<tr>
<td>S1,2,3</td>
<td>S3</td>
<td>156.3</td>
<td>120.8 ± 16.7</td>
<td>70.9 ± 2.1</td>
<td>43.4 ± 4.1</td>
<td>46.5 ± 1.4</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of 3D tracking errors (mm) on boxing validation sequence for S1, where ± indicates standard deviation over runs. Our results are comparable to the state-of-the-art [Taylor et al., 2010].

5.4 Conclusion and Future Work

In this paper, we try to learn a probabilistic prior model which is accurate yet expressive, and is tractable for both learning and inference. Our proposed stochastic GPLVM fulfills all these criteria - it effectively learns latent spaces of complex multi-activity datasets in a computationally efficient manner. When applied to tracking, our model outperforms state-of-the-art on the HumanEva benchmark, despite using very few particles and only a simple first-order Markov model for handling dynamics. In addition, we have also derived a novel approach for learning latent spaces incrementally. One of the great criticisms of current latent variable models is that they cannot handle new training examples without
relearning; given the sometimes cumbersome learning process, this is not always feasible. Our incremental method can be easily applied to an online setting without extensive relearning, which may have impact in applications such as robotics where domain adaptation might be key for accurate prediction. In the future, we plan to further investigate the incorporation of dynamics into the stochastic model, particularly for multiple activities.
6

Pose-based Action Recognition

6.1 Introduction

Some of the earliest works in action recognition focused on tracking body parts and classifying the joint movements [Gavrila & Davis, 1995; Campbell & Bobick, 1995; Yacoob & Black, 1999]. These *pose-based approaches* stem directly from the definition of an action as a sequence of articulated poses and are the most straight-forward. However, they require accurate tracking of body parts, which is a notoriously challenging task in its own right. As recent trends in action recognition have shifted towards analysis in natural and unconstrained videos, efforts have shifted from high-level modeling of the human body to directly classifying actions with abstract and low-level appearance features [Laptev & Lindeberg, 2003; Efros et al., 2003; Dollar et al., 2005; Jhuang et al., 2007; Schindler & Van Gool, 2008; Willems et al., 2009] in *appearance-based approaches*.

Appearance-based approaches require little to no high-level processing and can bypass many difficulties of pose estimation. They can also, to varying degrees, take some contextual information into account, such as background, since the features are usually not restricted to the human body. And despite having to deal with extremely high intra-class variations, such as human appearance, background clutter and differing viewpoints, appearance-based systems are applicable in many scenarios in which pose estimation may be extremely difficult (e.g. monocular views) or even impossible (e.g. very low resolutions [Efros et al., 2003]).

Pose-based action recognition approaches have received little attention in recent years due to the inherent difficulty of extracting human pose, particularly under realistic imaging conditions. But despite requiring more initial processing, pose-based action recognition has several advantages. First, pose representations suffer little of the intra-class variances that plague appearance-based systems. In particular, 3D skeleton poses are viewpoint and appearance invariant, such that actions vary less from actor to actor. Secondly, using pose-based representations greatly simplifies the learning for the action recognition itself, since the relevant high-level information has already been extracted. Given the great progress
6. Pose-based Action Recognition

Figure 6.1: We address the question of whether it is useful to perform pose estimation for the task of action recognition by comparing the use of appearance-based features, pose-based features and combined appearance- and pose-based features.

in pose estimation over the past few years [Bandouch & Beetz, 2009; Gall et al., 2010a; Taylor et al., 2010; Li et al., 2010; Gall et al., 2010b; Girshick et al., 2011], we feel that pose-based action recognition systems are now a feasible consideration and warrant a second look. More importantly, we address the question of whether it is useful to perform pose estimation for the action recognition task or if it is better for the classifier to identify the necessary information only from low-level appearance features drawn from video data.

In this work, we compare pose-based and appearance-based features for the task of action recognition as depicted in Figure 6.1. Pose-based features are derived from articulated 3D joint information, while we label as appearance-based any type of feature which can be extracted from video data without explicit articulated modeling of the human body. We modify the action recognition system of Chapter 3 to take in pose-based features and compare the two different sets of features. Finally, we investigate the combination of the two types of features into a single action recognition system.

6.2 Related Work

Early works in recognising human motions relied on recovering the articulated poses from each frame and then linking either the poses or pose-derived features into sequences. Pose information was typically obtained from moving light displays [Gavrila & Davis, 1995], motion capture systems [Campbell & Bobick, 1995] or segmentation [Yacoob & Black, 1999; Rao et al., 2002]. The sequences themselves were then classified through exemplar matching [Gavrila & Davis, 1995; Yacoob & Black, 1999; Rao et al., 2002] or with state-space models such as HMMs [Campbell & Bobick, 1995]. More recent approaches which
follow this line of work include [Yilmaz & Shah, 2005; Ali et al., 2007; Husz et al., 2011],
though they all assume that poses are readily available, either from hand labeling [Yilmaz
& Shah, 2005; Ali et al., 2007] or from an independent tracker [Husz et al., 2011], or they
use 2D pose estimates which are not view-point invariant [Ramanan & Forsyth, 2003;
Tran et al., 2011].

An alternative line of work models the entire body as a single entity, using silhouettes or
visual hulls [Bobick & Davis, 2001; Lv & Nevatia, 2007; Weinland et al., 2007; Weinland
& Boyer, 2008; Blank et al., 2005]. These works are sometimes referred to as pose-based
approaches, in reference to the extracted silhouettes of the human body. However, we con-
sider silhouettes to be a specialised appearance feature, since it offers little interpretation
of the individual body parts, and categorise these works as appearance-based approaches.

6.3 Methods

For classifying the actions, we adapt the Hough-transform voting method of Chapter 3
from using appearance-based features to pose-based features. For completeness, we re-
view the theory first presented in Chapter 3, Section 3.3, but reformulate the problem in
terms of generic features rather than as spatio-temporal patches.

A random forest is trained to learn a mapping between features extracted from the data
(either from appearance or pose) and a corresponding vote in an action Hough space,
i.e. a Hough forest [Gall et al., 2011b]. Each tree $T$ in the Hough forest is constructed from
a set of annotated features $\mathcal{P} = \{(F_i, a_i, d_i)\}$. $F_i$, feature $i$, can be either appearance-
based or pose based; $a_i$ is the action label ($a_i \in A$) and $d_i$ is the temporal displacement
of the feature center with respect to the action center in the sequence.

Trees are built recursively, starting with the entire collection of features at the root. At
each non-leaf node, a large pool of binary tests $t$ associated with the feature values are
randomly generated to split the annotated features $\mathcal{P}$ into two subsets, $\mathcal{P}_L(t)$ and $\mathcal{P}_R(t)$.
The optimal binary test $t^*$ maximizes the gain

$$\Delta H(t) = H(\mathcal{P}) - \sum_{S \in \{L, R\}} \frac{|\mathcal{P}_S(t)|}{|\mathcal{P}|} \cdot H(\mathcal{P}_S(t)). \quad (6.1)$$

Depending on the measure $H$ used, nodes can be either classification or regression nodes.
For classification, entropy

$$H(\mathcal{P}) = -\sum_{a \in A} p(a|\mathcal{P}) \log p(a|\mathcal{P}) \quad (6.2)$$
is used, where \( p(a|\mathcal{P}) \) is given by the percentage of samples with class label \( a \) in the set \( \mathcal{A} \). For regression, the sum-of-squared-differences is used as objective function:

\[
H(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_i \|d_i - \overline{d}\|_2^2,
\] (6.3)

where \( \overline{d} \) is the mean of the temporal displacement vectors. The \( t^* \) found is stored at the node and the sets \( \mathcal{P}_L(t^*) \) and \( \mathcal{P}_R(t^*) \) are passed to the left and right child node. The tree grows until some stopping criterion is met, i.e. the child node is of a maximum depth, or there are less than a minimum number of patches remaining. When training is complete, the leaves store the proportion of features per class which reached the leaf \( L \) (\( p^L_a \)) and the features’ respective displacement vectors (\( D^L_a \)).

At classification time, features are densely extracted from the test track and passed through all trees in the forest. The features are split according to the binary tests stored in the non-leaf nodes and, depending on the reached leaf \( L \), cast votes proportional to \( p^L_a \) for the action label \( a \) and the associated temporal center.

### 6.3.1 Appearance Features

When using appearance features with Hough forests, \( \mathcal{F}_i \) is a spatio-temporal patch (15 x 15 x 5 pixels) extracted from feature channels such as spatial gradients or optical flow, i.e. \( \mathcal{F}_i = (I^1_i, ..., I^f_i, ..., I^F_i) \), where each \( I^f_i \) is channel \( f \) at patch \( i \) and \( F \) is the total number of channels. We keep the same low-level appearance features as in Chapter 3: colour(Figure 6.2(a)), dense optical flow [Brox et al., 2004](Figure 6.2(b)) and spatio-temporal gradients(Figure 6.2(c,d)). While more sophisticated spatio-temporal features exist in the literature, we omit them from our experimentation as the above-mentioned low-level features achieve comparable results.

The binary tests at each node are comparisons of two pixels at locations \( \mathbf{p} \in \mathbb{R}^3 \) and \( \mathbf{q} \in \mathbb{R}^3 \) in feature channel \( f \) with some offset \( \tau \):

\[
t(f; \mathbf{p}, \mathbf{q}; \tau) = \begin{cases} 
0 & \text{if } I^f(\mathbf{p}) - I^f(\mathbf{q}) < \tau \\
1 & \text{otherwise}
\end{cases}
\] (6.4)

where \( \mathbf{p}, \mathbf{q}, f \) and \( \tau \) are learned during training.

### 6.3.2 Pose Features

For encoding pose information, we have adopted the relational features introduced by [Müller et al., 2005]. These features describe geometric relations between specific joints in a single pose or a short sequence of poses (e.g. the distance between the shoulder and
6.3. Methods

Figure 6.2: Appearance-based features. (a) Colour in the Lab colour space. (b) Dense optical flow in $x$ and $y$. (c) Spatial gradients in $x$ and $y$. (d) Temporal gradient.

Figure 6.3: Pose-based features. (a) Euclidean distance between two joints (red). (b) Plane feature: distance between a joint (red) and a plane (defined by three joints - black). (c) Normal plane feature: same as plane feature, but the plane is defined by its normal (direction of two joints - black squares) and a joint (black circle). (d) Velocity feature: velocity component of a joint (red) in the direction of two joints (black). (e) Normal velocity feature: velocity component of a joint in normal to the plane defined by three other joints (black).

Given multiple instances of an action, relational features are more robust to spatial variations than the poses themselves [Müller et al., 2005]. Previous works have also shown that semantically similar motions belonging to the same action are not necessarily numerically similar [Kovar & Gleicher, 2004; Müller et al., 2005]; by encoding the pose in a relative manner, it is easier to capture semantic similarity [Müller et al., 2005]. While [Müller et al., 2005] hand-tuned the features for indexing and retrieval of motion capture data, we allow them to be chosen randomly in the Hough Forest framework by casting $F_i$ as the geometric comparison of 3D joint positions and velocities.

Let $p_{j_i,t} \in \mathbb{R}^3$ and $v_{j_i,t} \in \mathbb{R}^3$ be the 3D location and velocity of joint $j_i$ at time $t$. The joint distance feature $F^{jd}$ (see Figure 6.3(a)) is defined as the Euclidean distance between joints $j_1$ and $j_2$ at time $t_1$ and $t_2$ respectively:

$$F^{jd}(j_1, j_2; t_1, t_2) = \|p_{j_1,t_1} - p_{j_2,t_2}\|,$$  \hspace{1cm} (6.5)
If \( t_1 = t_2 \), then \( F^{jd} \) is the distance between two joints in a single pose; if \( t_1 \neq t_2 \), then \( F^{jd} \) would encode distances between joints separated by time.

The plane feature \( F^{pl} \) (see Figure 6.3(b)) is defined as

\[
F^{pl}(j_1, j_2, j_3, j_4; t_1, t_2) = \text{dist}\left(p_{j_1, t_1}, \langle p_{j_2, t_2}, p_{j_3, t_2}, p_{j_4, t_2} \rangle \right),
\]

(6.6)

where \( \langle p_{j_2, p_{j_3}, p_{j_4}} \rangle \) indicates the plane spanned by \( p_{j_2}, p_{j_3}, \) and \( \text{dist}(p_j, \langle \cdot \rangle) \) is the Euclidean distance from point \( p_j \) to the plane \( \langle \cdot \rangle \). Similarly, the normal plane feature \( F^{np} \) (see Figure 6.3(c)) is defined as

\[
F^{np}(j_1, j_2, j_3, j_4; t_1, t_2) = \text{dist}\left(p_{j_1, t_1}, \langle p_{j_2, t_2}, p_{j_3, t_2}, p_{j_4, t_2} \rangle^n \right),
\]

(6.7)

where \( \langle p_{j_2, p_{j_3}, p_{j_4}} \rangle^n \) indicates the plane with normal vector \( p_{j_2} - p_{j_3} \) passing through \( p_{j_4} \).

The velocity feature \( F^{ve} \) (see Figure 6.3(d)) is defined as the component of \( v_{j_1, t_1} \) along the direction of \( p_{j_2} - p_{j_3} \) at time \( t_2 \):

\[
F^{ve}(j_1, j_2, j_3; t_1, t_2) = \frac{v_{j_1, t_1} \cdot (p_{j_2, t_2} - p_{j_3, t_2})}{\| (p_{j_2, t_2} - p_{j_3, t_2}) \|}.
\]

(6.8)

Similarly, the normal velocity feature \( F^{nv} \) (see Figure 6.3(e)) is defined as the component of \( v_{j_1, t_1} \) in the direction of the normal vector of the plane spanned by \( p_{j_2}, p_{j_3} \) and \( p_{j_4} \) at time \( t_2 \):

\[
F^{nv}(j_1, j_2, j_3, j_4; t_1, t_2) = v_{j_1, t_1} \cdot \hat{n}_{\langle p_{j_2, t_2}, p_{j_3, t_2}, p_{j_4, t_2} \rangle},
\]

(6.9)

where \( \hat{n}_{\langle \cdot \rangle} \) is the unit normal vector of the plane \( \langle \cdot \rangle \).

Binary tests for the pose features can be defined as follows:

\[
t(f; j_1, ..., j_n; t_1, t_2; \tau) = \begin{cases} 
0 & \text{if } F^f(j_1, ..., j_n; t_1, t_2) < \tau \\
1 & \text{otherwise}
\end{cases}
\]

(6.10)

where \( f, j_1, ..., j_n, t_1, t_2, \tau \) are pose-based feature types, joints, times and thresholds respectively. The parameters of the binary tests are selected during training as for the appearance features.

\[^1\text{We have kept all planes to be defined by joints at } t_2, \text{ though planes can in theory be defined in space-time by joints at different time points.}\]
6.3.3 Combined Features

For combining appearance and pose-based features into a single forest, we allow the generated random tests to randomly select whether to use appearance or pose features. In this way, the classifier automatically selects the most relevant features. Note that as appearance features, $F_i$ are patches sampled in space and time, as pose features, $F_i$ are poses sampled in time only.

6.3.4 Dataset and Experimentation

We focused our comparison on a home-monitoring scenario and used the TUM kitchen dataset [Tenorth et al., 2009], with multi-view video data and corresponding motion capture of actors setting a table. The actions are relatively realistic, subtler, and thus more challenging than standard benchmarks [Schuldt et al., 2004; Blank et al., 2005]. The fixed camera views eliminate much of the problems associated with background variance for appearance-based methods. We use the provided 3D joint positions in the dataset as pose data; these poses were determined by a markerless motion capture system [Bandouch & Beetz, 2009] i.e., not measured directly from markers, and then manually corrected in cases of large errors.

For each type of feature, we trained a forest of 15 trees of depth 15 each, generating 20000 random tests at all nodes. Of the 20 episodes in the dataset, we used 0-2,0-8,0-4,0-6,0-10,0-11,1-6 for testing and the remaining 13 episodes for training, from which we extracted 40 or less instances per action class (akin to the “full training set” of Chapter 4). We normalized the output of the Hough forest into a confidence score of each action over time, such that all actions at any time sum up to 1. To maintain consistency, we employed action recognition labels for the left hand as ground truth and also split the idle/carry class according to whether the subject is walking or standing.

For appearance-based features, we generated silhouettes using background subtraction and thus extracted bounding boxes which we linked into tracks. For each training instance, we randomly selected 1200 patches of size $15 \times 15 \times 5$. We trained independent forests for each camera view and then used a classifier-combination strategy (max-rule) to combine the outputs from the multiple views as per [Gall et al., 2010b].

For each type of pose-based feature, we trained a Hough forest on the full 3D skeletons provided (without limb endpoints) [Tenorth et al., 2009] as well as a reduced set of 13 joints (see Figure 6.5(a,b)). To simulate less-than-optimal pose estimations, we tested the classifier on joint data corrupted by Gaussian noise. For all pose-based feature experiments, we used 200 pose features per training instance\(^2\), with a time duration of 5 frames.

\(^2\)The possible number of unique pose features is much smaller than that of appearance-based features patches.
6.4 Results

6.4.1 Appearance-based Features

Using the appearance features described in Section 6.3.1, we achieved a combined performance of 0.698. A sample of the normalized classifier output is shown in Figure 6.5(c) and a confusion matrix for the individual classes is shown in Figure 6.4(a). Of the different features tested, colour was selected most often in the binary tests assigned to the nodes (43%, Fig 6.8(a)) in comparison to gradients (36%) and optical flow (21%).

6.4.2 Pose-based Features

All pose-based features outperformed the appearance-based features by 7-10%. Of the different types of pose-based features tested on the full skeleton, velocity features and plane features have comparable results, slightly higher than that of the joint distance (see...
Table 6.1. Combining all three feature types yielded the best result of 0.815, with the confusion matrix shown in Figure 6.4(b). For the reduced skeleton, results are all slightly lower than or equal to that of the full skeleton. The slight performance loss is probably not only due to the reduced number of joints but also due to the changed distribution of the joints on the skeleton (e.g., spine and hip in Figure 6.5(a,b)). When combining several features, the performance does not improve by much and is sometimes even lower than that of the best feature (see Table 6.1); this behaviour has been observed for random forests when the number of redundant feature channels increases [Gashler et al., 2008]. When all features are used together, the selection of the features at the nodes is nearly equal (Figure 6.8(b)).

When using only joint-distance features, we examined which of the joints were selected at the nodes according to the different actions (Figure 6.6). While different joints are favoured for the different actions, they are not immediately intuitive (e.g., joints from the legs or feet are not always selected for walking), suggesting that joints not associated with the limbs performing the action can also encode some information for discriminating the various actions.

Finally, we tested the robustness of the pose-based features by corrupting the test joint data with Gaussian noise to simulate errors in the extracted poses; classification accuracy versus noise is plotted in Figure 6.7. Performance of velocity features drops off quickly; joint distance and plane features, however, are more robust and maintain almost the same performance until around 75mm of added noise on each joint. At 100mm of added noise, performance is about equal to that of the appearance-based features.

### 6.4.3 Combined Appearance- and Pose-Based Features

We found no improvements after combining the appearance-based and pose-based features and achieve a mean classification of 0.801. The confusion matrix for combined outputs are shown in Figure 6.4(c). Looking at the normalized classifier output (Fig-

<table>
<thead>
<tr>
<th>Pose Features</th>
<th>Full skeleton (27 joints)</th>
<th>Reduced Skeleton (13 joints)</th>
</tr>
</thead>
<tbody>
<tr>
<td>joint distance (Figure 6.3 (a))</td>
<td>0.777</td>
<td>0.733</td>
</tr>
<tr>
<td>plane features (Figure 6.3 (b) &amp; (c))</td>
<td>0.802</td>
<td>0.787</td>
</tr>
<tr>
<td>velocity features (Figure 6.3 (d) &amp; (e))</td>
<td>0.803</td>
<td><strong>0.803</strong></td>
</tr>
<tr>
<td>joint distance &amp; plane features</td>
<td>0.784</td>
<td>0.769</td>
</tr>
<tr>
<td>joint distance &amp; velocity features</td>
<td>0.800</td>
<td>0.774</td>
</tr>
<tr>
<td>plane &amp; velocity features</td>
<td>0.804</td>
<td>0.773</td>
</tr>
<tr>
<td>all features</td>
<td><strong>0.815</strong></td>
<td>0.776</td>
</tr>
</tbody>
</table>

*Table 6.1: Classification performance with different pose-based features.*
Figure 6.5: (a) 27-joint full skeleton. (b) 13-joint reduced skeleton (in red). (c) Normalized action confidences for appearance-based, pose-based, and combined features for frames 200-700 of episode 0-8. In general, action confidences are higher for pose features than appearance features; given that they are also more accurate (see Table 6.1), this suggests that pose-based features are more discriminative than appearance-based features.

Figure 6.6: Joints selected by binary tests assigned to nodes of the Hough forest, where size of the plotted joint on the stick figure corresponds to frequency of selection. Note that each test uses two joints as the relational pose feature.
Figure 6.7: Classification accuracy as a function of the Gaussian noise added to the joint 3D locations. Velocity features are badly affected, while the other pose-based features slowly degrade in performance. The skeletons visualize the amount of added noise.

Figure 6.8: Feature selection. (a) In the appearance-based classifier, colour features (L, a, b) were selected at 43% of the nodes, gradient features (I_x, I_y, I_t) at 36% and optical flow (OF_x, OF_y) at 21%. (b) In the pose-based classifier, joint distance was selected at 19% of the nodes, plane features at 40% and velocity features at 41%. (c) In the combined classifier, appearance features were selected at 53% of the nodes and pose features at 47%.
6.5 Conclusion

In this chapter, we raised the question of whether it is useful to perform pose estimation for the action recognition task or if it is better for the classifier to identify the necessary information from low-level appearance features drawn from video data. Our results showed that, using the same classifier on the same dataset, pose-based features outperform appearance features. While pose-based action recognition is often criticized for requiring extensive preprocessing for accurate segmentation and tracking of the limbs, we have shown that this is not necessarily the case. Even with high levels of noise (up to 100mm of additive Gaussian noise), the pose-based features either matched or outperformed appearance-based features, indicating that perfect pose estimates are not necessary.

On the other hand, appearance features are more versatile to use than pose features and can be applied in many cases in which poses cannot be extracted. In addition, appearance-based features are capable of encoding contextual information (e.g. the appearance of the cupboards and drawers) which are missing from the poses alone. We believe that a combination of appearance and pose features would be most ideal when actions cannot be classified by the pose alone though this was not the case in our experiments. However, the question remains whether contextual information should be better learned from low-level or from high-level information extracted from the data. From looking at the most confused action class (“release grasp”), we observe that actions are often defined on high level information which is very difficult to learn from low-level features directly.
7

Integrated Action Recognition and Pose Estimation

7.1 Introduction

While the objectives of action recognition and pose estimation differ, they share a significant information overlap. For instance, poses from a given action tend to be a constrained subset of all possible configurations within the space of physiologically possible poses. Therefore, many state-of-the-art pose estimation systems use action-specific priors to simplify the pose estimation problem [Geiger et al., 2009; Li et al., 2010; Taylor et al., 2010; Lee & Elgammal, 2010; Chen et al., 2009]. At the same time, pose information can be a very strong indicator of actions and action labels can be determined from as little as a single frame [Schindler & Van Gool, 2008; Thurau & Hlavac, 2008; Yang et al., 2010; Maji et al., 2011]. However, as neither pose estimation nor action recognition are trivial tasks, few systems have tried to couple the two tasks together into a single system. On the one hand, priors from many state-of-the-art pose estimation systems are of a single activity, thereby assuming that the activity is already known, and cannot handle sequences of multiple activities [Taylor et al., 2010]. On the other hand, action recognition approaches either model poses implicitly through pose-related descriptors [Thurau & Hlavac, 2008; Klaeser et al., 2010b; Natarajan et al., 2010; Yang et al., 2010] or completely bypass the difficulties of pose estimation and directly classify actions with abstract and low-level appearance features [Laptev & Lindeberg, 2003; Efros et al., 2003; Dollar et al., 2005; Jhuang et al., 2007; Schindler & Van Gool, 2008; Willems et al., 2009].

Given that human pose estimation and action recognition are such closely intertwined tasks, information from one task can be leveraged to assist the other and vice versa. As such, we advocate the use of information from action recognition to help with pose estimation and vice versa for the following reasons. First, using the results of an action classifier is a simple way to bring together many single-activity priors for pose estimation in multi-activity sequences. Secondly, pose-based action recognition has several advantages. For example, pose representations suffer little of the intra-class variances common
Figure 7.1: The framework begins with 2D appearance-based action recognition based on low-level appearance features (a) such as colour, optical flow and spatio-temporal gradients. Outputs of the 2D action recognition (b) are used as a prior distribution (c) for the particle-based optimization for 3D pose estimation (d) (Arrow 1). Finally, 3D pose-based action recognition (g) is then performed based on pose-based features (f) extracted from the estimated poses (e) (Arrow 2).

In appearance-based systems; in particular, 3D skeleton poses are viewpoint and appearance invariant, such that actions vary less from actor to actor. Furthermore, using pose-based representations greatly simplifies learning for the action recognition itself, since the relevant high-level information has already been extracted.

In Chapter 4, Leveraging Actions for Pose Estimation, we introduced a method which builds upon the results of action recognition to help with human pose estimation. In Chapter 6, Pose-based Action Recognition, we introduced a pose-based action recognition method. Here, we bring the two chapters together into a single integrated framework, as
illustrated in Figure 7.1. Our framework begins with 2D appearance-based action recognition based on low-level appearance features such as colour, optical flow and spatio-temporal gradients. The outputs of the 2D action recognition are used as a prior distribution for the particle-based optimization for 3D pose estimation (Arrow 1 in Figure 7.1). Finally, we perform 3D pose-based action recognition based on pose-based features extracted from the estimated poses (Arrow 2 in Figure 7.1). As a whole, the integrated system can be viewed as an EM-like procedure, transitioning from (appearance-based) action labels to 3D poses and then back to (pose-based) action labels. Such a process can be iterated and we take this one step further to re-estimate the pose.

7.2 Pose-based action recognition

Using the TUM Kitchen Dataset [Tenorth et al., 2009], we follow the same procedure as the pose-based action recognition experiments in Chapter 6 except we now substitute our own estimated poses from Chapter 4 at test time. We test using the “ground truth” poses, i.e. the poses provided in the dataset TUM as well as the estimated poses using our action prior, uniform prior and baseline. Like in Chapter 4, we compare two different training sets (S1 training set vs. full training set) and also look at the effects of smoothing the poses over time.

We show the action recognition performance in Table 7.1. Unlike the fused 2D action recognition, there is about 10% performance drop from the full training set to using only the S1 training set. A similar drop in performance does occur for 2D action recognition in the single view case (Table 4.6.2), though the classifier fusion scheme compensates for the loss in performance in the 2D case. In contrast to the pose estimation, the action recognition clearly benefits from more training data.

As reported in Chapter 6, Table 6.1, when testing with the TUM poses, there is little difference between the joint distance, plane features and velocity features; combining the three different types of features does not improve the action recognition and average classification remains at 0.81. When using the poses estimated from the action prior with 200 particles, there is a 7%-10% performance drop from the TUM poses; the best performance is achieved using the combined features (0.74) though the velocity features on the smoothed poses are similar (0.73). When using the poses estimated from the uniform prior, there is a further 5% drop; again, the best performance is achieved using either the combined features or the velocity features from the smoothed poses (both 0.68). The poses estimated from the baseline algorithm are too poor to be used for action recognition, indicating that a pose estimation error over 100mm is insufficient for reliable pose-based action recognition.

While temporal smoothing has no effect on the joint distance features and the plane features, it is essential for the velocity features, which are by definition more sensitive to
<table>
<thead>
<tr>
<th></th>
<th>S1 training set</th>
<th></th>
<th></th>
<th></th>
<th>full training set</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>particles</td>
<td>joint dist.</td>
<td>plane</td>
<td>velocity</td>
<td>combined</td>
<td>joint dist.</td>
<td>plane</td>
<td>velocity</td>
</tr>
<tr>
<td>TUM</td>
<td>-</td>
<td>0.59</td>
<td>0.68</td>
<td>0.68</td>
<td>0.70</td>
<td>0.76</td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td>ap</td>
<td>200</td>
<td>0.55/0.54</td>
<td>0.56/0.57</td>
<td>0.31/0.54</td>
<td>0.54/0.57</td>
<td>0.67/0.67</td>
<td>0.70/0.68</td>
<td>0.46/0.73</td>
</tr>
<tr>
<td>up</td>
<td>200</td>
<td>0.52/0.52</td>
<td>0.51/0.50</td>
<td>0.30/0.49</td>
<td>0.54/0.55</td>
<td>0.67/0.65</td>
<td>0.64/0.62</td>
<td>0.41/0.68</td>
</tr>
<tr>
<td>base</td>
<td>200</td>
<td>0.19/0.19</td>
<td>0.20/0.20</td>
<td>0.13/0.23</td>
<td>0.25/0.24</td>
<td>0.28/0.28</td>
<td>0.27/0.27</td>
<td>0.21/0.36</td>
</tr>
<tr>
<td>ap</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td>0.68/0.68</td>
<td>0.72/0.72</td>
<td>0.52/0.77</td>
<td>0.75/0.75</td>
</tr>
<tr>
<td>up</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td>0.69/0.69</td>
<td>0.71/0.69</td>
<td>0.52/0.76</td>
<td>0.73/0.73</td>
</tr>
</tbody>
</table>

Table 7.1: Action recognition performance for different relational pose features extracted from the ground truth, action prior, uniform prior and baseline pose estimates. For the action prior, uniform prior and baseline, we report two values to indicate the effects of smoothing on the action recognition performance (unsmoothed/smoothed). TUM: ground truth; ap: action prior; up: uniform prior; base: baseline.
7.2. Pose-based action recognition

noise. This effect was also observed in the synthetic experiments of Chapter 6, Section 6.4.2 when the TUM poses were corrupted by additive Gaussian noise. If we use the poses estimated with 300 particles, which are on average 2%-3% \((\text{action prior})\) or 11% \((\text{uniform prior})\) lower in 3D error in comparison to poses estimated with 200 particles, then the action recognition performance is about 4% higher. Since the pose estimates with \((\text{action prior})\) and \((\text{uniform prior})\) become similar with more particles, the action recognition performance becomes similar as well.

In comparison to the appearance-based action recognition performance (see Chapter 4, Table 4.6.2) using the poses from the \((\text{action prior})\) (0.76) or \((\text{uniform prior})\) (0.75) with the full training set and 300 particles is better than the fused results (0.71). Using the S1 training set and 200 particles, however, performance is about 10% worse. Comparing the action confidence outputs shown in Figure 7.3, confidences for the 3D pose-based action recognition is higher than the 2D appearance-based recognition. Using the full training versus the S1 training set also results in slightly higher confidences, though this effect is more pronounced in the 2D appearance-based outputs than the 3D pose-based outputs.

Figure 7.2: Confusion matrix for the 3D pose-based action recognition using the full training set with velocity features extracted from smoothed pose estimates, estimated with 300 particles.
When looking at the confusion matrix (Figure 7.2), one sees that the most difficult classes are again the ambiguous actions such as “release grasp” and “take object”, though performance is better than the 2D appearance-based system (Figure 4.5).

We finally remark that the pose estimation with uniform prior already provides reliable estimates for pose-based action recognition (0.75) that performs better than appearance-based action recognition (0.71), although there is room for further improvement to match the performance with the “ground truth” (TUM) poses (0.81). This is particularly relevant for scenarios when view- and environment-dependent training data is difficult to acquire and only MoCap training data is available.
7.3 Closing the Loop

In our system, we transition from an action label to pose estimates and then back to actions again. One can continue and re-estimate the pose based on the 3D pose-based action labels; based on the re-estimated poses, one can again solve for the action labels. In a subsequent iteration, the pose estimation error is reduced by about 3% (see Table 7.2) and the action recognition by 1% error (using velocity features from smoothed poses, we improve from 0.77 to 0.78). These results highlight that for both action recognition and pose estimation, the more accurate the information being leveraged, the better the results.

7.4 Conclusion

Based on the work of Chapters 3, 4 and 6 we have presented an integrated system that couples the closely intertwined tasks of action recognition and pose estimation.

Within our proposed action recognition system, 3D pose-based features have been shown to be more successful at classifying the actions than 2D appearance-based features. The same has been shown to be true even when the pose-based features were extracted from the estimated poses of our pose estimation system, indicating that the quality of estimated poses with an average error between 42mm-68mm is sufficient enough for reliable action recognition. Since 3D pose-based features are, in contrast to 2D appearance features, view-independent, it is easier to acquire training data from other datasets. In this way, the pose estimation system with a uniform prior and the pose-based action recognition method can be easily set-up at any location without requiring additional view-specific training data.
Table 7.2: 3D Error for TUM kitchen dataset in mm (using full training set and 300 particles). \( \text{ap(2D)} \): action prior from 2D action recognition (Table 4.6.2); \( \text{ap(3D)} \): action prior from 3D action recognition (Table 7.1).

<table>
<thead>
<tr>
<th>(mm)</th>
<th>0-2</th>
<th>0-4</th>
<th>0-6</th>
<th>0-8</th>
<th>0-10</th>
<th>0-11</th>
<th>1-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ap(2D)} )</td>
<td>46.9 ± 23.0</td>
<td>57.4 ± 28.6</td>
<td>59.5 ± 27.3</td>
<td>49.2 ± 33.8</td>
<td>63.2 ± 36.0</td>
<td>66.6 ± 32.2</td>
<td>74.3 ± 37.0</td>
</tr>
<tr>
<td>( \text{ap(3D)} )</td>
<td>44.1 ± 15.7</td>
<td>58.7 ± 26.7</td>
<td>58.3 ± 23.9</td>
<td>46.6 ± 31.5</td>
<td>61.0 ± 30.0</td>
<td>68.1 ± 34.8</td>
<td>68.5 ± 31.1</td>
</tr>
<tr>
<td>( \text{ap(2D)+smooth} )</td>
<td>42.4 ± 22.1</td>
<td>52.4 ± 26.1</td>
<td>54.4 ± 26.1</td>
<td>44.6 ± 33.1</td>
<td>58.1 ± 35.2</td>
<td>63.1 ± 31.0</td>
<td>70.4 ± 35.9</td>
</tr>
<tr>
<td>( \text{ap(3D)+smooth} )</td>
<td>39.3 ± 13.9</td>
<td>53.7 ± 24.9</td>
<td>53.0 ± 21.9</td>
<td>42.0 ± 30.6</td>
<td>55.8 ± 28.6</td>
<td>64.4 ± 34.0</td>
<td>64.6 ± 29.9</td>
</tr>
</tbody>
</table>
Online Action Segmentation and Classification

8.1 Introduction

In recent years, much research effort has been placed on video-based human action recognition. To advance the field, several datasets (see [Kuehne et al., 2011] for a recent overview) have been proposed for the study and evaluation of various approaches. The community has made great progress from recognition on the standard benchmarks KTH [Schuldt et al., 2004] and Weizmann [Blank et al., 2005]. New datasets are more natural and unconstrained, with sequences from television [Patron-Perez et al., 2010], movies [Laptev et al., 2008; Kuehne et al., 2011], broadcast sports [Rodriguez et al., 2008] and YouTube [Liu et al., 2009b]. Action recognition using these datasets follows a well-defined task: given an input video sequence, already temporally (and often spatially) segmented, select a label from $n$ available activities as output. Such scenarios are generally unrealistic. First, the action classes are atypical of a real world set-up. Secondly, the available segments may not be optimal for recognition [Satkin & Hebert, 2010]. Third and most importantly, having already temporally segmented input sequences is an implausible assumption, especially for many real-world applications.

To overcome these issues, we propose in this chapter an action recognition algorithm which can perform simultaneous temporal segmentation and classification. The algorithm takes a continuous stream of human body poses as input and returns the temporal boundaries of segmented actions as well as action labels of each segment as output. Output labels can be an action already present in the training data or an unknown label indicating a previously unseen action. The online handling of an incoming stream of body poses makes the algorithm particularly well suited for applications such as surveillance, home monitoring and human-computer interfacing. When an input sequence of poses is classified as unknown, it could be used to either generate an alarm (e.g. when detecting unusual activity) or be considered as a new candidate activity to be added to the training data.
In the past, pose-based action recognition has been considered impractical due to the difficulties of extracting human pose, especially in realistic imaging conditions. The few exceptions that model pose explicitly [Gall et al., 2011a; Yao et al., 2011a; Tran et al., 2011] require poses determined offline by sophisticated pose estimation systems [Bandonch & Beetz, 2009; Andriluka et al., 2009]. With recent developments in consumer depth cameras and associated algorithms which take RGB-depth input, however, it has become possible to estimate human pose relatively accurately in real-time [Ganapathi et al., 2010; Shotton et al., 2011; Girshick et al., 2011].

Our work follows a key-pose based action representation as per [Gall et al., 2011a], in which actions, defined by a sequence of key poses, are compared using a modified Levenshtein distance [Levenshtein, 1966]. Using this similarity measure, we propose an online method for segmenting actions from an incoming stream of poses. Contributions in the current chapter include:

- A novel algorithm which performs simultaneous segmentation and classification of actions in an online manner.
- A thresholding scheme for detecting abnormal or unseen actions.
- A new RGB-depth human-computer interaction dataset to evaluate our framework in a real-time and real-world scenario.

While the concept of joint segmentation and classification is not new [Hoai et al., 2011], the problem has only been addressed from an offline perspective. Given a sequence in its entirety, segmentation is a much easier task as it is possible to optimize both forwards and backwards in time. To the best of our knowledge, no other works have addressed concurrent online segmentation and action recognition.

8.2 Related Works

Several previous works in action recognition have addressed the estimation of an action’s spatio-temporal boundaries, *i.e.* action detection [Shechtman & Irani, 2007; Laptev & Pérez, 2007; Niebles et al., 2008; Ke et al., 2010; Yuan et al., 2009; Cao et al., 2010]. Actions are modeled as spatio-temporal objects and their detection follows counterparts in standard object detection methods. The treatment of actions as objects, however, allows for simultaneous occurrence in space and/or time, when this may not be the case in many applications.

On the other hand the segmentation of body poses into actions from motion capture has been addressed in a variety of manners based on zero-velocity crossings of angular velocity [Jenkins & Mataric, 2002], changes in PCA dimensionality [Barbic et al., 2004],
8.3. Approach

We propose a framework which takes in a continuous stream of pose data and performs both temporal segmentation and action classification online. It uses a key pose based representation which is computationally efficient and also robust to inter-subject differences, as demonstrated in [Gall et al., 2011a]. First, a set of key poses are extracted from a training set. Then, a vocabulary of training actions is constructed from the key poses using a genetic algorithm. Finally, at run-time, the input stream of human body poses is continuously converted to the nearest key pose and a dynamic lookup is performed on the vocabulary of actions. The dynamic lookup determines the final temporal segmentation and action label as well as detect unknown actions.

8.3.1 Pose and Action Representation

Let $x_j$ be the 3D coordinate of the $j$-th joint, offset by the pelvis (root) position and rotation and $d_{ij}$ be the difference vector $x_i - x_j$ normalized to unit length. A pose $p$ can then be represented in absolute terms as

$$p_{\text{abs}} = \{x_j\} \quad j \in J,$$

where $J$ is the set of all joints.
where $J$ is the set of all body joints, or represented in relative terms as

$$p_{rel} = \{ \hat{d}_{ij} \} \quad i \in J, \ j \in J, \ i \neq j.$$ \hfill (8.2)

For a set of poses $P_{\text{train}}$ taken from training sequences, one can select a set of key poses $K = \{ k \}$. Previous works have selected key poses based on $k$-means clustering ($K_{km}$) [Gall et al., 2011a], motion energy minima/maxima ($K_{mot}$) [Lv & Nevatia, 2007], or action discriminativeness ($K_{act}$) [Weinland et al., 2007].

An action $A$ of length $T$, i.e.

$$A = (p_1, ..., p_t, ..., p_T)$$ \hfill (8.3)

can be modeled in key pose representation by first converting each pose $p_t$ to the most similar key pose. We have used Euclidean distance for matching poses but any other metric can be used as well. Consecutively equal key poses are then merged into a single pose, e.g. $(k_1, k_1, k_1, k_2, k_2, k_3)$ would be merged into $(k_1, k_2, k_3)$. While this compression creates a loss in temporal information, it has the advantage of being able to account for differences in temporal alignment between actions without the need for time warping.

### 8.3.2 Comparing Two Actions

We adopt a modified version of the Levenshtein distance as proposed in [Gall et al., 2011a] to compare two actions or sequences of key poses. The distance is determined by counting the minimum number of edits (insertions, deletions or substitutions) required to modify one key pose sequence into a second, weighted by differences between the key poses involved in the edit. With such a distance measure, however, shorter actions tend to have smaller distances, which we offset by normalizing the distance of two actions $A_i$ and $A_j$, $D(A_i, A_j)$, by the sum of their duration in terms of key poses (see Eq.8.4):

$$D_n(A_i, A_j) = \frac{D(A_i, A_j)}{|A_i| + |A_j|}.$$ \hfill (8.4)

An important practical property of this distance function is its unimodality. Based on our observations, Eq. 8.4 has a well defined peak when matching a template string to an input of increasing size (see Fig. 8.1).
8.3.3 Vocabulary of Actions

Once all training actions have been converted to sequences of key poses, a common representation or an action exemplar for each action is selected to be used at run-time. Having a common representation makes the recognition approach scale efficiently with the number of training examples and activities. Ideally, such an exemplar should have a minimal distance to instances of the same action and a maximal distance to instances of other actions. As the space of possible exemplars is huge, we use a genetic algorithm for narrowing the search efficiently.

The algorithm is initialized with a family of instances for each action, which at the first iteration are simply the corresponding training sequences. We then define the fitness function $FF$ of a sequence $A_i$ as

$$FF(A_i) = \frac{1}{|T|-|I|} \sum_{j \in T, j \notin I} D_n(A_i, A_j) - \frac{1}{|I|} \sum_{j \in I} D_n(A_i, A_j),$$

where $T$ is the set of all the training sequences and $I$ are the training instances of the considered action. At each iteration step, or generation, a set of sequences are randomly sampled with a probability proportional to their fitness. The sequences are then evolved through a randomly chosen operation:

- **Crossover** One key pose sequence is merged with another; both sequences are chosen according to their fitness. The last $m$ key poses of the second sequence are appended to the first $n$ key poses of the first sequence, $m$ and $n$ being random integers between 1 and the respective sequence’s length.

- **Substitution** One key pose within the sequence is substituted with another random key pose with a probability inversely proportional to the (Euclidean) distance between the poses.

- **Deletion** One randomly chosen key pose is removed from the selected sequence.

- **Insertion** An additional key pose is inserted into the sequence at a random location. Again, the key pose to be inserted is selected according to a probability inversely proportional to its distance to neighbouring key poses.

After $N$ generations, the best individual encountered so far, according to $FF$, is chosen as the vocabulary entry for the considered action. This procedure is repeated for all the actions that build the training dataset. Figure 8.2 shows the obtained exemplars for two actions of the CMU basketball refereeing dataset. They are displayed as a sequence of key poses, consistently with their actual representation.
Figure 8.1: Distance function of an input stream to 4 different action exemplars, in different colors. The behaviour of the distance function is unimodal with respect to the length of the input.

Figure 8.2: Exemplar selection: Exemplars obtained by the GA optimization for two actions of the CMU dataset. They are visualized as sequences of key poses.

8.3.4 Online Processing

Every time a new body pose is observed as input, it is first converted to the corresponding key pose, which is then appended and compressed to the string of key poses currently under evaluation. Such string is then matched, using the modified Levenshtein distance, to all known action exemplars. The system continually outputs the label of the action exemplar with the closest distance so far. The candidate string continues to be compared to all exemplars until they are implausible, i.e. the matching distance keeps decreasing. If the matching distance to an exemplar starts to increase consistently, such exemplar is no longer matched to the input to save computation time. When such a condition is met by all exemplars, the analysis of the current string terminates and the lowest distance obtained by the input string gives the output action. The input stream is then segmented at the time
step in which such distance was obtained. If the resulting distance is exceeds a certain threshold (previously set using training or validation data), then the action is marked as unknown. The procedure is summarized in Algorithm 3.

Algorithm 3 Online Processing of a Body Pose Stream

```plaintext
while Flag do
    $C_{t_0 \rightarrow t} \leftarrow \text{Compress}(K_0 \cdots K_t)$
    for $a = 1$ to NumActions do
        $D_{a,t} \leftarrow \text{Distance}(A_a, C_{t_0 \rightarrow t})$
    end for
    if isIncreasing($D_{i,t_0 \rightarrow t}$) forAll actions $i$ then
        Flag $\leftarrow$ False
    end if
    CurrentLabel $\leftarrow \arg\min_i(D_{i,t_0 \rightarrow t})$
    $t \leftarrow t + 1$
end while
Score $\leftarrow \min(D_{i,t_0 \rightarrow t})$
if Score $<\text{Threshold}$ then
    Label $\leftarrow \arg\min_i(D_{i,t_0 \rightarrow t})$
else
    Label $\leftarrow$ Unknown
end if
StartTime $\leftarrow t_0$
EndTime $\leftarrow t$
```

8.4 Experiments

8.4.1 Datasets

We perform experiments on markered motion capture data from the [CMU Mocap Database] and on markerless motion capture data from a novel Human-Computer Interaction (HCI) dataset captured with a consumer depth camera. From the CMU database, we select 13 sequences from 7 subjects performing basketball refereeing signals. In all experiments, we use subjects 26-29 for training and 30-32 for testing. A visualization of the refereeing signals is shown in Fig. 8.3. Since the subjects remain stationary and move only their arms, we have considered only their shoulder, elbow and wrist joints in the experiments.
8. Online Action Segmentation and Classification

Figure 8.3: CMU Dataset: A set of 7 refereeing signals are shown as trajectories of select upper body limbs.

Figure 8.4: HCI Dataset: A set of 8 HCI actions are shown as trajectories of select upper body limbs.

Figure 8.5: Evaluation of the classification performance of the system on pre-segmented sequences of the CMU dataset. Parameters varied are the pose parametrization, the key pose selection technique, the number of key poses and the used action exemplars. The results of DTW with 1-NN matching are shown as a baseline.
The HCI dataset is made by 6 subjects performing 8 different actions. Since it is specifically designed for online evaluation, we have also recorded a test sequence for each of the subjects where different actions are performed in a continuous fashion. The 8 recorded actions are depicted in Fig. 8.4. The motion trajectories shown in the figure are not as smooth as in the markered motion capture, and this is due to the noisy measurements provided by the markerless rgb-depth based system.

### 8.4.2 Classification

We first evaluate our system only for classification by using already segmented sequences of the CMU dataset. As a baseline comparison, we use Dynamic Time Warping (DTW) to warp all training sequences with respect to the test sequence and then use 1-NN matching based on Euclidean distance. The test sequence is then assigned the label of the nearest training sequence. We follow a similar 1-NN approach for matching the key pose-represented test actions with respect to action exemplars, though matching is now done by the modified Levenshtein distance explained in Section 8.3.2. With this classification setup, we compare the use of absolute and relative pose representations, the different key pose selection methods (see Section 8.3.1), the number of selected key poses as well as different methods of selecting action exemplars. Results are plotted in Fig. 8.5, from which a number of observations can be made regarding key pose representation and selection as well as action exemplar selection.

**Key pose representation and selection** The relative pose representation, \( p_{rel} \) (bottom row of Fig 8.5) consistently outperforms absolute pose representation, \( p_{abs} \) (top row of Fig 8.5) for both the DTW baseline as well as all instances of key pose-represented actions.

Fig. 8.5 shows that the selection through the minima/maxima of motion energy, \( K_{mot} \), has on average a better performance than other techniques. Finally, having more key poses does not guarantee better performance. For both pose representations and almost all key pose selection methods, the best performance peaks somewhere between 20 to 40 key poses. We believe that this is because with too few key poses, there is not sufficient representation to distinguish all the actions while with too many key poses, performance also drops since there is too much variation between the actions of the same label. This drop in performance with too many key poses is particularly evident when actions are represented only by a single exemplar but less so with nearest neighbor matching.

**Action exemplar selection** We compared the action exemplars selected by the genetic algorithm to three other baselines: a randomly selected exemplar from the training set, the best exemplar in the training set (the initialization point of the genetic algorithm, see
Section 8.3.3) as well as retaining the entire training set of sequences as exemplars. We see a graceful decline in performance as we progress from keeping all training exemplars to one well-selected training exemplar (on average GA selected performs better than best) to one randomly selected exemplar. With a relative pose representation, matching to one training exemplar already has comparable performance to the DTW baseline. If all training examples are available, performance even exceeds the DTW baseline.

**Design choices** At test-time, matching the input to all training examples is not a viable option if large datasets are considered. We choose, based on the classification results (see Fig. 8.5), a set of parameters which will then be used for the online evaluation of continuous streams of human poses: a relative pose representation, the motion energy-based key pose selection and the genetic algorithm for determining action exemplars. Selection of the optimal classification parameters for joint segmentation and classification is motivated by the fact that segmentation is performed based on the distance between the chosen exemplars and the input stream. As such, having more discriminative exemplars, *i.e.* better classification performance, should lead to better segmentation results.

### 8.4.3 Segmentation and Classification

We perform joint segmentation and classification on the CMU sequences as well as on the HCI dataset. We evaluate the results on a per-frame and per-segment basis, based on the thorough analysis presented in [Ward et al., 2011] on performance metrics for activity recognition. Due to the segmentation aspect of our work, we are interested in going beyond standard accuracy metrics of the frames and/or segments by also characterizing the segmentation results as well. Detected activities can be correctly matched (*i.e.* correct timing and label), falsely inserted (I) when there is no correspondence to ground truth and deleted (D) when no events are detected. In addition, overfills (O) and underfills (U) can

<table>
<thead>
<tr>
<th>Action</th>
<th>Frame Level</th>
<th>Segment Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel</td>
<td>0.61</td>
<td>0.37</td>
</tr>
<tr>
<td>Double Dribble</td>
<td>0.64</td>
<td>0.43</td>
</tr>
<tr>
<td>Over and Back</td>
<td>0.91</td>
<td>0.67</td>
</tr>
<tr>
<td>No Score</td>
<td>0.29</td>
<td>0.17</td>
</tr>
<tr>
<td>Jump Ball</td>
<td>0.92</td>
<td>0.67</td>
</tr>
<tr>
<td>Pushing</td>
<td>0.34</td>
<td>0.30</td>
</tr>
<tr>
<td>Technical</td>
<td>0.46</td>
<td>0.33</td>
</tr>
<tr>
<td>Average</td>
<td>0.60</td>
<td>0.42</td>
</tr>
</tbody>
</table>

*Table 8.1: CMU Refereeing Signals Dataset: Joint Segmentation and Classification Performance*
Figure 8.6: Qualitative segmentation results for the CMU dataset for 3 test sequences.

Figure 8.7: Qualitative segmentation results for the HCI dataset for 3 test sequences. Scroll Up and Scroll Down actions tend to be confused, while results for other actions are usually robust.

occur when the segmentation exceeds or falls short of the the ground truth boundaries respectively, fragmentation (F) when an activity is segmented into multiple occurrences (of the correct class) and merging (M) when two events are segmented as a single event. Note that these errors cannot occur in isolation; for example, the insertion of an event for one class results in deletion or fragmentation of another class.

Frame-level and segment-level accuracy for each of the classes are shown in Table 8.1 and 8.2 for the CMU and HCI datasets respectively. When looking at the segmented results in Figs. 8.4.3 and 8.7, almost all actions have been detected. However, we are particularly prone to overfill and underfill errors (see Table 8.3 for corresponding quantitative results), i.e. incorrect border segmentations. This could be attributed to the compression of key poses, making it difficult to find the exact border between actions.
Unknown action detection  To evaluate the capability of our algorithm to detect unseen actions, we performed a simple experiment by removing the “Zoom-Out” action from the set of exemplars. In the testing phase, the missing action was classified as “Scroll-Down”. Note, however, that the confidence score (inversely proportional to the distance) is much lower during this misclassified segment (see Fig. 8.8). As such, a simple thresholding on the distances would allow the algorithm to determine potentially unknown actions.

8.5 Conclusion

We have presented a novel method which takes in a continuous stream of body poses and performs online action segmentation and classification. Unlike previous works in action recognition, the proposed method does not rely on already-segmented test sequences. Obtained results show that the approach performs well on both a per-frame and per-segment level. A confidence score is provided for each recognized segment, which can then be used to isolate previously unseen actions.
Figure 8.8: Unknown action detection: **Top row:** Segmentation results with full set of action exemplars on a test sequence from the HCI dataset. **Middle row:** Segmentation results on the same sequence after removing the action Zoom Out from the set of exemplars. Frames corresponding to that action are then assigned another label. **Bottom row:** Confidence scores of the action labels corresponding to the top row (blue) and the middle row (red). When the action is not part of the dataset and therefore unknown, the confidence level of its classification label is much lower. In this case a simple thresholding is enough to detect the unseen action.
Conclusion

This thesis presented methods of analyzing human motion from video. The primary focus was on action recognition and we presented two different approaches for this task, one appearance-based and one pose-based. In addition, we examined how action recognition could be leveraged to help with pose estimation and an alternative method for encoding poses in a low-dimensional manifold. We integrated action recognition and pose estimation into a single system, taking output from the appearance-based action recognition as a prior for 3D pose estimation and refinement of the action labels using pose-based action recognition. Finally, we examined simultaneous temporal segmentation and classification of actions from a continuous stream of poses. A more detailed look at the specific contributions are summarized below.

9.1 Contributions

In Chapter 3, Appearance-based Action Recognition, we presented a Hough-transform based voting framework for localization and classification of human actions in video. The problem was broken down into two stages by first localizing the person of interest in the video using a tracking-by-detection approach (Appendix A) and subsequently determined the action label by voting in an spatio-temporal-action Hough space. Voting was done by training a random forest to learn a mapping between densely-sampled feature patches and their corresponding votes in the Hough space. The leaves formed a discriminative multi-class codebook that shares features between the action classes and vote for action centers in a probabilistic manner. The approach was also extended to classify emotions (Appendix B) and group actions (Appendix C).

In Chapter 4, Leveraging Actions for Pose Estimation, we presented an algorithm for 3D pose estimation which takes the results of the action recognition system presented in Chapter 3 as a prior distribution for optimization. We use a collection of action-specific low-dimensional manifolds to simplify the pose estimation problem for a multi-activity sequence. The pose estimation system jointly optimizes over all the manifolds and then
continues the optimization in a high-dimensional space of all possible human poses to handle unobserved pose variations and transitions.

In Chapter 5, *Pose Estimation of Complex Activities*, we explored an alternative method for embedding poses in a low-dimensional manifold to serve as a pose prior. Existing approaches for establishing pose priors tend to be either too simplistic (linear), too complex to learn, or can only learn latent spaces from ”simple data”, i.e. single activities such as walking or running. We presented an efficient stochastic gradient descent algorithm for learning probabilistic non-linear latent spaces composed of multiple complex activities. We further extend this method and derive an incremental learning algorithm for an online setting which can update the latent space without extensive relearning.

In Chapter 6, *Pose-based Action Recognition*, we adapted the method of Chapter 3 to handle relational pose features for pose-based action recognition. We compared the use of appearance- and pose-based features for action recognition, arriving at the conclusion pose-based features outperform low-level appearance, suggesting that pose estimation is beneficial for the action recognition task.

In Chapter 7, *Integrated Action Recognition and Pose Estimation*, we presented an integrated framework that coupled the individual systems of Chapters 3, 4 and 6. We demonstrated the feasibility of pose-based action recognition using poses from our own pose-estimation system of Chapter 4 and show improvement in both pose estimation and action recognition as we iterate over the two procedures.

In Chapter 8, *Online Action Segmentation and Classification*, we explored the temporal segmentation and classification of actions from a continuous stream of body poses. We convert the incoming stream into a compressed key pose representation which allows for efficient online analysis and demonstrate the system on both markered motion capture data and less robust markerless poses collected from the Microsoft Kinect.

9.2 Perspectives

**Localize first, classify second.** The appearance-based action recognition system presented in Chapter 3, uses an initial spatial localization step and a subsequent temporal localization and classification step. This effectively breaks down action detection into a two stage problem of “where are the people“ and then ”what are these people doing at this time“? By first finding the people in the scene, we have simplified the action detection problem; such an approach is supported by [Klaeser et al., 2010a] and [Klaeser et al., 2010b], which found that bag-of-feature approaches can also benefit from person-detection [Klaeser et al., 2010a] and promoted the idea of ”person-centric“ action recognition [Klaeser et al., 2010b], i.e. detecting and tracking people before classifying their actions.
Despite the advances of pedestrian trackers and other specialized tracking-by-detection approaches (e.g. the "sports" tracker presented in Appendix A), it is not always feasible to separately generate tracks first in all action recognition applications. In cases where only parts of the body are visible (which is often the case in sequences from television and film), part-based detection and tracking would be necessary. In addition, the tracking problem becomes much more difficult once there are multiple people present in a scene. In such a scenario, it may be more beneficial to bias the tracker to target only people performing certain actions and ideally, one would like to simultaneously track and classify at the same time.

**Appearance- versus pose-based action recognition.** In Chapter 6, we directly compared the use of appearance and pose-based features for the action recognition task. With “ground-truth” poses from the TUM Kitchen dataset\(^1\), we found that pose-based features outperformed the appearance features. However, appearance features are more versatile to use than pose features and can be applied in many cases in which poses cannot be extracted. In addition, appearance-based features are capable of encoding some contextual information (e.g. the appearance of the cupboards and drawers in the TUM Kitchen dataset) which are missing from the poses alone. Therefore, a combination of appearance and pose features would be most ideal when actions cannot be classified by the pose alone, though this was not the case in our experiments.

What we found in Chapter 7, when using poses from our own pose estimation system was that action recognition performance is heavily dependent on accuracy of the pose. Currently, the system can tolerate errors of around 42-68mm (see footnote\(^2\)). Pose estimates with higher errors result in action recognition performances which are lower than the multi-view appearance-based action recognition, making it meaningless to first estimate the pose. To place such a value into perspective, pose estimates from the Microsoft Kinect have a mean average precision across joints of 0.74 [Shotton et al., 2011] to 0.8 [Girshick et al., 2011]\(^3\). At the time of writing, we believe that the Microsoft Kinect has the most potential in terms of motivating pose-based action recognition systems. However, the direct application of pose estimates from the Kinect into the framework of Chapter 7 is not yet feasible.

We have tried to incorporate some robustness to pose estimation error through the use of relational pose features. However, perhaps an even more robust method would be to replace estimated poses at individual time steps with a compressed set of key poses, as proposed in Chapter 8. While such a representation is robust enough to handle poses estimated from the Kinect, granularity between distinguishable actions is much lower and the subtleties between actions are quickly lost.

---

1. Poses were obtained with a markerless motion capture system and then manually corrected.
2. Such a value is of course highly dataset dependent.
3. In [Shotton et al., 2011] and [Girshick et al., 2011], pose estimation is framed as a detection problem. Each joint must be predicted within 100mm of the ground truth to be considered a true positive.
Integrating actions and poses. The success of the proposed method in Chapter 4 for pose estimation, which achieves state-of-the-art performance, builds on the ability to jointly optimize over several low-dimensional spaces that represent poses of different actions. Beyond that, unobserved pose variations or unobserved transitions between actions are resolved by continuing the optimization in the high-dimensional space of all human poses. Our experiments have shown that this combination is superior compared to optimization in either space individually. On the one hand, the full human pose has too many degrees of freedom to be optimized efficiently. On the other hand, learned low-dimensional embeddings can be poor at generalization, such that poses which are not present in the training data can not be well estimated\(^4\). The proposed method benefits from the efficiency of low-dimensional embeddings but also overcomes the problem of generalization.

The selection of priors for pose estimation, however, be it the action prior or the uniform prior, is related to the amount of computational resources available at hand, \textit{i.e.} the number of particles to be used and hence the amount of time required for the pose estimation algorithm. The less the resources, the more benefit there is to be gained from using action information; with more resources, differences between the action prior and the uniform prior are no longer distinguishable. Given unlimited resources, however, even the baseline algorithm which does not make use of any action information is expected to perform reasonably well. Our experiments have shown that an action prior improves the pose estimation when the number of particles for optimization are limited. The benefit of the action prior compared to a uniform prior, however, becomes smaller with an increasing number of particles.

We have shown the advantages of using action recognition for pose estimation through the selection of priors as well as the advantages of using pose estimation for action recognition\(^6\) and have integrated the two tasks into a single system\(^7\). Performance of the pose-based action recognition, while tolerant of errors, is directly related to the pose accuracy. As such, we envision two possible settings for using the integrated system. If one has more computational resources for pose estimation, then it is preferable to use the uniform prior and bypass the initial 2D action recognition stage, since the benefits of the action prior for pose estimation is no longer distinguishable from the uniform prior. On the other hand, with more limited resources and a focus on pose estimation, it is more preferable to keep the 2D action recognition to improve the accuracy of the pose estimates.

Contextual information To advance vision-based human motion analysis beyond isolated actions and poses, one should integrate contextual information, either from the environment or objects. Environmental context, \textit{e.g.} the type of scene or even specific locations within a scene can provide strong indicators to the types of actions and there-

---
\(^4\)One can try to minimize this impact with more expressive embeddings, for example, the proposed stochastic GPLVM approach in Chapter 5.
fore poses which can be expected. Furthermore, interactions with objects can often be
the defining characteristic of an action and having a better understanding of human-object
interactions would lead to improved recognition on high-level actions such as “take ob-
ject” or “release grasp” from the TUM Kitchen dataset. This line of work has already
started (see Chapter 2, Section 2.6). Future work will be focused on methods of encod-
ing the contextual information so that it can be efficiently integrated into coupled action
recognition and pose estimation systems.
A

Tracking People in Broadcast Sports

A.1 Introduction

Object tracking in video is a long-standing computer vision problem; in particular, tracking people has captured the interest of many researchers due to its potential for applications such as intelligent surveillance, automotive safety and sports analysis. State-of-the-art people trackers have predominantly focused on pedestrians for traffic or surveillance scenarios. For sports analysis, however, standard pedestrian trackers face significant challenges since in many broadcast sports, the camera moves and zooms to follow the movements of the athlete. Furthermore, in some sports, the athlete may perform abrupt movements and have extensive body articulations that result in rapid appearance changes and heavy motion blur (see Fig. A.1). As such, sports tracking to date [Kang et al., 2003; Choi et al., 2004; Okuma et al., 2004; Kristan et al., 2009; Breitenstein et al., 2009; Hess & Fern, 2009] has been limited to team sports such as football and hockey, in which there is wide view of the playing field and athletes remain relatively upright. In addition, these works are primarily focused on the data-association problem of multi-target tracking and do not deviate substantially from the pedestrian tracking scenario.

In the current work, we present a method for tracking people in monocular broadcast sports videos by coupling a standard particle filter [Doucet et al., 2001] with a vote-based confidence map of an “athlete”-detector [Gall et al., 2011b]. We target sporting disciplines in which the athletes perform fast and highly articulated movements, e.g. diving and gymnastics. Tracking in these types of sports is particularly difficult since the athletes do not remain in an upright configuration. Our confidence map, built from the Hough accumulator of a generalized Hough transform designed for people detection, is well suited for handling pose and appearance changes and athlete occlusions, as it is generated from a vote-based method. While we focus on tracking in broadcast sports clips, as they provide a challenging testbed, our method is applicable to generic people tracking in unconstrained videos. We demonstrate the tracker’s effectiveness on the UCF Sports Dataset [Rodriguez et al., 2008], a collection of footage from the 2008 Olympics and a PETS 2009 sequence [Ferryman & Shahroekni, 2009].
Figure A.1: Select frames from the UCF Sports Dataset [Rodriguez et al., 2008], showing challenges of tracking in sports videos such as (a) motion blur, (b) extensive body articulation and (c) occlusions.

Figure A.2: Components of the sports tracker. From the original frame(a), the vote-based confidence map(b) is computed using a Hough Forest [Gall et al., 2011b]. The dynamical model estimates camera motion from the frame border(c) and motion of the tracked athlete from the frame interior using optical flow(d). Each particle in the particle distribution(e) is weighted according to the confidence map and appearance features such as colour and texture(f).
A.2 Related Works

Early approaches in sports tracking began with background extraction and then morphological operations to isolate foreground areas which may represent the athlete [Kang et al., 2003; Choi et al., 2004; Sullivan & Carlsson, 2006]. Tracking was then performed by enforcing spatial continuity through either Kalman or particle filtering. These approaches, both single- and multi-camera, relied heavily on colour as a cue for separating the athletes from the background as well as for tracking, though shape and motion information of the athletes have also been used [Lu & Little, 2006; Kristan et al., 2009]. Most of the proposed algorithms, however, have been designed for specific sports, such as soccer [Kang et al., 2003; Choi et al., 2004], speed-skating [Liu et al., 2009a] or hockey [Okuma et al., 2004] and rely on sport-specific scene-knowledge, such as distances between field lines [Khattoonabadi & Rahmati, 2009].

Accurate modeling of target observations, be it athletes, pedestrians or generic objects has been the focus of several current tracking works. One line of approach learns and adapts appearance models online [Collins et al., 2005; Grabner et al., 2008; Babenko et al., 2009]; these methods cope well with appearance changes and are not limited to tracking specific object classes, but are susceptible to drift as tracking errors accumulate. Another line of approach uses pre-trained models of the targets. Tracking-by-detection methods follow this type of paradigm, in which object detectors are first trained offline and detections across the sequence are then associated together to form the track, e.g. by particle filtering. Tracking-by-detection has been used for pedestrians [Breitenstein et al., 2009; Andriluka et al., 2008; Leibe et al., 2008] and in specific sports such as hockey [Okuma et al., 2004; Breitenstein et al., 2009] and soccer [Breitenstein et al., 2009; Hess & Fern, 2009]. All these approaches, however, assume that the humans remain upright - an assumption that does not hold for broadcast sports videos in general.

The key component of our tracker is the use of a vote-based confidence map to estimate the location of the targets. It is similar in spirit to the Fragment Tracker in [Adam et al., 2006], which tracks object fragments or patches that vote for an object center. Our work differs from [Adam et al., 2006] in that we track possible object centers from the accumulated votes in the confidence map rather than the individual patches that vote for a center.

A.3 Sports Tracker

The sports tracker is a tracking-by-detection approach with three components: (1) a continuous vote-based confidence map to estimate the target location (see A.3.2), (2) appearance matching of the target based on feature templates (see A.3.3) and (3) motion estimation of the camera and the target from optical flow (see A.3.4).
A.3.1 Tracking Overview

Tracking in the sports videos is done using a particle filter [Doucet et al., 2001]. We model the state \( s = \{x, y, c, u, v, d\} \in \mathbb{R}^6 \) of a human by the image position and scale \((x, y, c)\) and velocity and change in scale \((u, v, d)\). For particle \( i \), the weight at frame \( t \) is assigned as follows:

\[
    w_i^t = \frac{1}{Z} \exp \left( -K \cdot (\alpha \cdot V_1(s_i^t) + (1 - \alpha) \cdot \sum_f \lambda_f V_2(s_i^t, f)) \right). \tag{A.1}
\]

The term \( V_1 \) measures the response in the vote space (Figure A.2(b), see A.3.2) for particle \( s_i^t \). The term \( V_2 \) measures the similarity of particle \( s_i^t \) with respect to some template appearance feature \( f \) extracted from the associated bounding box of the particle (Figure A.2(f), see A.3.3). \( K \) is a scaling constant and \( \alpha \in [0, 1] \) is a weighting parameter for \( V_1 \) and \( V_2 \). \( \lambda_f \) are weighting parameters between the different features and sum up to 1. \( Z \) is the normalization term of the weights.

The tracker is initialized using the ground truth from the first frame of the sequence. Particles are propagated by a dynamical model accounting for camera motion (Figure A.2(c)) and estimated athlete motion (Figure A.2(d), see A.3.4).

A.3.2 Vote-Based Confidence Map

The confidence map is generated from the output of a Hough forest [Gall et al., 2011b] trained for detecting athletes. The Hough forest is a random forest trained to map image feature patches to probabilistic votes in a 3D Hough accumulator \( H \) for locations and scales of the athlete. We use cropped and scale-normalized images of the athletes as positive examples, background images as negative examples, and colour and histograms of gradients [Dalal & Triggs, 2005] as features. For a detailed description of the training procedure, we refer to [Gall et al., 2011b]. For detection, feature patches are densely sampled from the image and passed through the trees of the Hough forest to cast votes in \( H \). While a detector as in [Gall et al., 2011b] thresholds the local maxima in \( H \) to obtain a discrete set of object hypotheses, we consider \( H \) as a continuous confidence mapping of athlete locations and scales. Directly considered a continuous confidence mapping of athlete locations and scales if taken as a whole, or transformed into discrete detection hypotheses by taking the maxima locations and values. From \( H \), the vote response \( V_1(s_i^t) \) of particle \( s_i^t \) is determined by

\[
    V_1(s_i^t) = -\log \left( \sum_{x \in N(s_i^t) \cap H} G(s_i^t - x) \right), \tag{A.2}
\]

i.e. we sum the votes in the neighborhood \( N \) of \( s_i \) weighted by a Gaussian kernel \( G \). Note that the sum is in the range of \([0, 1]\).
A.3.3 Appearance Model

The appearance of particle $s_t^i$, denoted as $V_2(s_t^i, f)$, is a measure of similarity between that particle’s feature response $h^f(s_t^i)$ and some template $h_T^f$ for feature $f$. To measure similarity, we use the Bhattacharyya coefficient $BC$:

$$V_2(s_t^i, f) = 1 - BC(h_T^f, h^f(s_t^i))$$  (A.3)

As image features, we use HSV colour histograms and local binary patterns [Ojala et al., 2002] to model colour and texture respectively. For the template, we use a weighted mixture of the particle’s feature response in the initial frame at $t_0$ and the previous frame $t-1$. Weighting of the individual appearance features in the final particle weight (Equation A.1) is determined by $\lambda_f$, in our case $\lambda_{\text{colour}}$ and $\lambda_{\text{texture}}$.

A.3.4 Dynamical Model

For the dynamical model, we use an estimated velocity based on optical flow. The reason for this is two-fold. First, constant-velocity models which perform well for tracking walking or running people perform poorly for actions in which the athletes move erratically, i.e. in gymnastics. Secondly, in many broadcast sports, the cinematography already provides some framing and tracking of the athlete, i.e. when the camera pans to follow the athlete across a scene. As such, the position of the athlete changes in an inconsistent manner within the frame and it is necessary to estimate the particle motion while accounting for camera motion. Particles are propagated from frame to frame by

$$(x, y, c)^i_t = (x, y, c)^i_{t-1} + (u, v, d)^i_{t-1} + N(0, \sigma_{\text{tran}}),$$  (A.4)

where $\sigma_{\text{tran}}$ is the variance of added Gaussian noise for the transition. Velocity is estimated as a weighted mixture between camera-compensated optical flow and velocity in the previous frame, while change in scale remains constant.

$$(u, v)^i_t = \eta \cdot \left((u, v)^{of}_{t-1} - \gamma \cdot (u, v)^{\text{cam}}_{t-1}\right) + (1 - \eta) \cdot (u, v)^i_{t-1}$$  (A.5)

Optical flow is computed according to [Brox et al., 2004]; camera motion is estimated as the average optical flow in the border of the frame (Figure A.2(b)). $\eta$ is a weighting parameter between estimated motion versus a constant velocity assumption, while $\gamma$ serves as a scaling parameter for the estimated camera motion.
A.4 Experiments

A.4.1 Datasets

We evaluate our tracker on sports and non-sports videos. For sports, we use the UCF Sports Dataset [Rodriguez et al., 2008] and our own collection of Olympics footage. The UCF dataset, consisting of 150 sequences (50-100 frames each) from network news videos, was originally intended for action recognition. To supplement the UCF dataset, we annotated 31 sequences (150-2000 frames each) from the 2008 Olympics, featuring sports such as diving, equestrian and various disciplines of gymnastics. The sequences are longer and more challenging than UCF, with significant motion blur and de-interlacing artifacts. For non-sports videos, we track three people from the PETS 2009 [Ferryman & Shahrokni, 2009] sequence S2.L1, View001. For the sports datasets, we train on all images of annotated athletes within the dataset other than from the test sequence, in a leave-one-out fashion. For the PETS sequence, we trained on the TUD pedestrian database [Andriluka et al., 2008].

A.4.2 Evaluation

For evaluation, we use the VOC [Everingham et al., 2007] criterion (the intersection over union, IOU, of the tracked bounding box and the ground truth bounding box must be greater than 0.5). We hand annotated select frames of the Olympics data and the PETS sequence and used linear interpolation to generate bounding boxes for the frames in between. For the UCF database, bounding boxes were provided as a part of the ground truth annotation released with the data.

We run three experiments on the Olympics data to test the impact of each component of the tracker. First, the confidence map is compared with discrete detections; for fair comparison, we generate the discrete detections from the confidence maps by thresholding1 the local maxima of $H$ (see A.3.2). Second, the effect of the appearance modeling is tested by removing the colour and texture features from the tracker. In the third experiment, we vary the $\eta$ and $\gamma$ parameters and look at the effects of removing camera compensation as well as comparing our current dynamic model to a constant velocity model. We also compare our tracker’s performance on the PETS2009 sequence with the Fragment Tracker in [Adam et al., 2006], using source code provided on the author’s website2. Run time on all datasets was around 1 second per frame for 50 particles on a standard CPU.

---

1 The threshold was set to achieve a high recall.

2 http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm
Table A.1: Average tracking performance on the Olympics sequences, where a higher % indicates better performance. There is a decrease in tracking performance with each removed component of the tracker; the most critical component seems to be the vote map, as using discrete components results in significantly lower performance.

### A.5 Results

#### Olympics Data
We take the following parameter settings \(\{\alpha = 0.5, \lambda_{\text{colour}} = 0.09, \lambda_{\text{texture}} = 0.91, \eta = 0.3, \gamma = 1.5\}\) and use these as our default scenario. Parameters are set at these values for all experiments unless otherwise stated. Results for default scenario, split by discipline are shown in Figure A.3 (a). Tracking results from the first three experiments are shown in Table A.1. From the first experiment, we see that using the vote-based confidence map in the tracker gives a significant improvement over the use of discrete detections. In fact, for the sports, having discrete detections is comparable to not using any detections \((\alpha = 0)\). This can be attributed to the many false-positive detections with high confidences, which have the effect of attracting and clustering the particles to erroneous locations. Our second experiment shows that removing either or both appearance features results only in a slightly decreased performance, again emphasizing the importance of the confidence map in the tracker. In the last experiment, we show that the use of our motion estimate in the dynamical model outperforms a constant velocity model, particularly with having the camera compensation. Varying \(\eta\) and \(\gamma\) had little effect, with performance ranging from 71.6%-74.9%. Select frames from the tracked results are shown in Figure A.4.

#### UCF Sports Dataset
Tracking performance for the UCF Dataset are shown in Figure A.3(b); select frames from the tracks are shown in Figure A.5. On average, 81.8% ± 16.0% of the frames have tracks with an IOU greater than 0.5. The tracker performs well in sports where people remain upright, i.e. golfing, running, and skateboarding, but faces some difficulty with sports with more extensive articulation such as diving, kicking and gymnastics. Part of the error results from ground truth being tight bounding boxes around the athletes while tracked bounding boxes are of a fixed ratio.
Figure A.3: Average tracking performance by sport for (a) Olympics Dataset and (b) UCF Sports Dataset, where a higher % indicates better performance.

Figure A.4: Tracking on the Olympics sequences: select frames from diving (top), equestrian (second row), floor routine (third row) and vault (bottom). The tracker successfully follows the athletes but has difficulty with very fast motions, e.g. on the floor routine, in the third frame, the tracker fails to track the tumbling sequence through the air.
Figure A.5: Tracking on the UCF Sports Dataset, showing select frames from running (top row), skateboarding (middle row) and gymnastics (bottom row).

Figure A.6: Select frames from the PETS2009 sequence. The tracker successfully follows the target in track 1 and 2 (top and middle row). Track 2 is particularly challenging as it is over 500 frames long and several people including the target are all wearing black clothing. In frame 294 of track 2, the tracker handles occlusion of the target by another person wearing similar coloured clothing. In frame 31 of track 3 (bottom row), there is an identity switch (true target is indicated by the white arrow); in frame 115, the tracker switches back onto the correct target. Figure is best viewed in colour.
PETS2009 We compare the performance of our Sports Tracker with the Fragment Tracker [Adam et al., 2006] in Table A.2. The Sports Tracker successfully follows two of the three tracks, but breaks down on track 3, most likely due to the lack of multiple target handling. There are two identity switches, first from the target to another person at frame 31 when several people group together and then back to the target after frame 115. Select frames are shown in Figure A.6. The Fragment Tracker successfully tracks one of the three tracks, but suffers from drift on the other two tracks and around 100 frames into the tracks, loses the target completely.

<table>
<thead>
<tr>
<th>Track</th>
<th>Frame</th>
<th>Sports Tracker</th>
<th>Fragments Tracker [Adam et al., 2006]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21 - 259</td>
<td>85.4</td>
<td>14.8</td>
</tr>
<tr>
<td>2</td>
<td>222 - 794</td>
<td>95.5</td>
<td>12.6</td>
</tr>
<tr>
<td>3</td>
<td>0 - 145</td>
<td>13.8</td>
<td>78.6</td>
</tr>
</tbody>
</table>

Table A.2: Comparison of the Sports Tracker with the Fragments Tracker in [Adam et al., 2006] on the PETS2009 S2.L1 View001 sequence. Results shown are the % of frames with IOU > 0.5, where a higher % indicates better performance.

A.6 Conclusion

We have presented a method for tracking athletes in broadcast sports videos. Our sports tracker combines a particle filter with the vote-based confidence map of an object detector. The use of feature templates and target motion estimates add to the performance of the tracker, but the strength of the tracker lies in the confidence map. By providing a continuous estimate of possible target locations in each frame, the confidence map greatly outperforms tracking with discrete detections. Possible extensions to the tracker include making voting for the confidence map adaptive and online, so that tracked bounding boxes are of varying ratios to yield tight bounding boxes around the athlete’s body, and making a multi-target version of the tracker to better handle team sports.
B

Facial Expression Recognition

B.1 Introduction

Computers, already part of our lives, will never seamlessly blend in until they are able to communicate with people the same way as we do. This means that machines should be able to sense and reproduce affective behavior, i.e., understand the user’s feelings and react accordingly.

Facial expressions represent one of the most important ways for humans to transmit and recognize feelings and intentions. Since the seminal work of [Darwin, 1872], the field has fascinated psychologists, neuroscientists, and lately also computer scientists. [Ekman & Friesen, 1971]’s study suggested that all emotions belong to a rather small set of categories. These “basic” emotions (anger, disgust, fear, happiness, sadness, and surprise) are expressed by the same facial movements across different cultures, and therefore represent an appealing choice when designing automatic methods for facial expression classification.

The ability for a computer system to sense the user’s emotions opens a wide range of applications in different research areas, including security, law enforcement, medicine, education, and telecommunications [Sebe et al., 2004]. However, it is important not to confuse human emotion recognition from facial expression recognition: the latter is merely a classification of facial deformations into a set of abstract classes, solely based on visual information. Indeed, human emotions can only be inferred from context, self-report, physiological indicators, and expressive behavior which may or may not include facial expressions [Cohn, 2006].

There are two main methodological approaches to the automatic analysis of facial expressions [Fasel & Luettin, 2003]: Judgment-based approaches attempt to directly map visual inputs such as images or video sequences into one of a set of categories, while sign-based approaches describe facial expressions by means of coded facial actions, e.g., methods based on Ekman’s Facial Action Coding System [Ekman et al., 1978], which represent
face deformations by activations of a set of Action Units corresponding to facial muscles movements.

This paper presents a judgment-based method for the classification of facial expressions into one of the basic emotion labels. Having seen the success of Hough transform-based methods [Ballard, 1981] for object detection [Maji & Malik, 2009; Ommer & Malik, 2009; Fanelli et al., 2009; Gall et al., 2011b] and action recognition [Yao et al., 2010a], we investigate a Hough transform voting approach applied to the task of facial expression recognition. After having localized and normalized the faces with respect to the eyes’ centers, the image sequences are arranged into cuboids, or, extending the notation of [Yao et al., 2010a], expression tracks. These are a representation of the face which is invariant to location, scale, and (in-plane) rotation. On the tracks, classification is performed by casting votes for the expression label and temporal center of the expression.

To our knowledge, this is the first time that a Hough-voting approach is applied to the task of facial expression recognition. As in the work of [Yao et al., 2010a], the voting is performed by a forest of random trees, or Hough forest [Gall et al., 2011b], and a mapping is learned between densely sampled spatio-temporal features and the center of the expression in the video sequence. The trees are trained in a multi-class fashion and can therefore discriminate between different classes simultaneously. The leaf nodes can vote for each class and represent a discriminative codebook sharing features across classes.

Compared to the task of action recognition form video [Yao et al., 2010a], facial expressions present more subtle differences and are therefore more difficult to classify. Additions to the work of [Yao et al., 2010a] include the normalization of the tracks with respect to rotation and the use of more discriminative shape features. In the experiment section, we thoroughly evaluate our system on standard databases of facial expressions. Our results are comparable to state-of-the-art methods, which supports our idea that Hough-voting approaches are promising tools for advancing in the field of automatic facial expression recognition.

B.2 Related work

[Suwa et al., 1978] were the first to attempt automatically recognizing facial expressions in 1978. Since then, the new field of research has seen a steady growth, gaining momentum in the 1990’s thanks to the advances in algorithms for face detection and the availability of cheaper computing power, as the surveys of [Fasel & Luettin, 2003] and [Zeng et al., 2009] show. following we review some of the recent works forming a context for our proposed approach.

The initial face localization and normalization step, common to virtually all approaches to facial expression recognition from video, serves to achieve a representation of the face
invariant to scale, translation, in-plane rotation, and illumination conditions. The literature is rich with approaches which normalize the images based on the location of the face [Buenaposada et al., 2008], of the eyes [Bartlett et al., 2005], or thanks to facial features tracking methods [Aleksic & Katsaggelos, 2006; Dornaika & Davoine, 2008]. After the normalization stage, the remainder of an automatic facial expression recognizer consists of feature extraction and classifier design. Features need to minimize variation within the expression classes while maximizing the variation between different classes. Features can be computed from geometric measurements, e.g., from the locations of specific points tracked on the face throughout the sequence [Shang & Chan, 2009; Aleksic & Katsaggelos, 2006]. Alternatively, image-based features can be extracted from texture patches covering either the whole face (holistic) or specific regions (local). Commonly employed feature extraction methods from facial textures and their temporal variations include optical flow [Essa & Pentland, 1997; Yeasin et al., 2006], Gabor filter responses [Bartlett et al., 2005; Wu et al., 2010], and Linear Binary Patterns [Shan et al., 2009; Zhao & Pietikäinen, 2009]. For the actual classification, AdaBoost and its combination with Support Vector Machines have recently gained a lot of attention [Bartlett et al., 2005; Littlewort et al., 2006]. Other popular approaches include nearest-neighbor searches [Buenaposada et al., 2008] and Hidden Markov Models [Shang & Chan, 2009; Zhao & Pietikäinen, 2009; Aleksic & Katsaggelos, 2006].

Trees and forests have been previously used for action recognition, but only as indexing structures for performing efficient nearest-neighbor searches [Reddy et al., 2009; Jiang et al., 2012]. Following [Yao et al., 2010a], we build a holistic, image-based method for recognizing facial expressions which uses a random forest to learn the mapping between 3D video patches and votes in a Hough space for the label and the temporal location of the expression.

B.3 Voting Framework for Facial Expression Recognition

Having seen the successful application of random forests and Hough voting to action recognition [Yao et al., 2010a], we investigate its performance on the task of recognizing facial expressions. In order to introduce the basics of the method, we assume our data to be already arranged into expression tracks, i.e., the face images are cropped and aligned as shown in Fig. B.1(a). Section B.4 provides insights on how this normalization is performed.
B.3.1 Training

We start from the assumption of having a set of training expression tracks available for each class \( c \in C \). Training sequences are annotated for the expression label and the temporal location of the apex in the track. In order to learn the mapping between patches from the expression tracks and a voting space, we use the Hough forest method [Gall et al., 2011b]. Previously developed for 2D single-class object detection, Hough forests have recently been extended to handle multi-class detection in the spatio-temporal domain and applied to the task of action recognition [Yao et al., 2010a].

Randomized Hough forests are composed of a set of random trees. A tree \( T \) is constructed from a set of patches \( \mathcal{P} = \{P_i = (I_i, c_i, d_i)\} \) randomly sampled from the training sequences. \( P_i \) is a 3D patch (e.g. of \( 20 \times 20 \times 3 \) pixels) sampled from the expression track as illustrated by the colored cuboids in Fig. B.1. \( I_i \) are the multi-channel features extracted at a patch, i.e., \( I_i = (I_{1i}, I_{2i}, ..., I_{Fi}) \in \mathbb{R}^F \), where each \( I_{fi} \) is feature channel \( f \) at patch \( i \) and \( F \) is the total number of feature channels. \( c_i \) is the expression label \( (c_i \in C) \) and \( d_i \) is a 3D displacement vector from the patch center to the center of the expression in the sequence. Fig. B.1 shows an expression track \((a)\) and sample 3D patches extracted from it \((b)\), voting for both the expression class and the center of the expression in the sequence.

During training, the trees are built recursively starting from the root, as in the standard random forest framework [Breiman, 2001]. Each non-leaf node is assigned a binary test based on the patch appearance \( I \); depending on the test’s result, the training patches are split into the children nodes. The process is iterated until a leaf is created, either from reaching a maximum tree depth or from reaching a minimum number of remaining patches. As tests, we use simple comparisons of two pixels at locations \( p \in \mathbb{R}^3 \) and \( q \in \mathbb{R}^3 \) in feature channel \( f \) with some offset \( \tau \). For node \( B \), the corresponding test \( t_B \) is defined as:

\[
\text{(a)} \quad \text{(b)}
\]

**Figure B.1:** Hough voting in the case of expression recognition. (a) Sample facial expression track. (b) Sample 3D patches drawn from the track, voting for the expression label and its spatio-temporal center.
Similar to [Yao et al., 2010a], each binary test is assigned in order to either optimize class-label or center offset uncertainty. To this end, a set of binary tests \( \{ t^k \} \) is generated at each node, with random values for \( f, p, q \) and \( \tau \), and evaluated on all the patches arriving at that node. The optimal test (the minimizing class label or center offset uncertainty in the split of the patches) is then chosen and assigned to the node.

When the training process is over, the leaves will store \( p_{cL} \) (the proportion of patches per class label which reached the leaf) and \( D_{cL} \) (the patches’ respective displacement vectors). Patches extracted from different classes arriving to the same leaf share the same features. The proportion of patches per class label at a leaf note can be used as class probabilities \( p_{cL} \) which can indicate the degree of sharing among classes.

### B.3.2 Facial Expression Classification

At classification time, patches are densely extracted from the test track and sent through all trees in the forest. The patches are split according to the binary tests in the non-leaf nodes and, depending on the reached leaf, cast votes proportional to \( p_{c} \) for the expression label and votes for the spatio-temporal center of each class \( c \) according to a 3D Gaussian Parzen window estimate of the center set vectors \( D_{c} \). Votes from all patches are integrated into a 4D Hough accumulator, exemplified in the left part of Fig. B.2 for a sequence expressing anger. The dark spots correspond to the probabilistic votes that have been cast by the patches and accumulated in the four-dimensional space (x and y location, time, and class label). As the track has already been localized in space, we marginalize the votes into a 2D accumulator for only class label and time. The local maximum in the remaining Hough accumulator finally leads to the classification prediction, as displayed in Fig. B.2, right. For a formal description of the voting process, we refer the reader to [Yao et al., 2010a].

Time-scale invariance could be achieved by up-sampling or down-sampling the tracks, and then applying the same Hough forest to label expressions displayed at different speeds. However, the system has some tolerance built in through the variation in speed observed in the training data and we therefore did not consider multiple time scales.

\[
 t_{B,f,p,q,\tau}(I) = \begin{cases} 
 0 & \text{if } I^f(p) < I^f(q) + \tau \\
 1 & \text{otherwise} 
\end{cases} 
\] (B.1)

\[
 t_{B,f,p,q,\tau}(I) = \begin{cases} 
 0 & \text{if } I^f(p) < I^f(q) + \tau \\
 1 & \text{otherwise} 
\end{cases} 
\] (B.1)
Figure B.2: Left: An example of a 4D Hough image, output of the voting for a clip displaying anger. The dark dots represent clusters of votes. Right: Example Hough voting space reduced to the two dimensions expression class and time. The maximum (in dark) is taken as the expression label and temporal location.

Figure B.3: Left: automatic face and eye tracking employed for the normalization of the facial images. Right: example log-Gabor responses extracted from a normalized expressive face.
B.4 Building the Expression Tracks

In order to arrange the data in the required normalized expression tracks, we align the faces based on the locations of the eyes. Faces are rotated and scaled so that the eyes lie on the same horizontal line and have the same inter-ocular distance. The invariance to rotation, an addition to the work of [Yao et al., 2010a], is necessary for the task of expression recognition, which are more subtle and harder to recognize than human actions. When ground-truth annotation of the eye locations is not available, we employ a completely automatic method, i.e., the first part of the system described in [Fanelli et al., 2009]: after tracking the face by means of an online-boosting method [Grabner et al., 2006], the eyes are localized thanks to their unique shape [Valenti & Gevers, 2008] and tracked using a pair of Kalman filters. The automatic procedure is shown in the left part of Fig. B.3.

B.4.1 Feature Extraction

For classification, Yao et al. [Yao et al., 2010a] used simple features such as color, greyscale intensity, spatial gradients along the x and y axis, and frame to frame optical flow. In our approach, inspired by [Schindler & Van Gool, 2008], we extract features separately representing the form and the motion of the face in the expression track. The information about form comes from the responses of a bank of log-Gabor filters. In comparison to standard (linear) filters, log-Gabor filters show an improved spectrum coverage with fewer scales [Field, 1987]. The response $g$ at position $(x,y)$ and spatial frequency $w$ is:

$$g^w(x,y) = \frac{1}{\mu} e^{- \frac{\log(w(x,y))/\mu}{2 \log \sigma}}, \quad (B.2)$$

where $\mu$ is the preferred frequency and $\sigma$ a constant used to achieve an even coverage of the spectrum. We use a bank with 3 scales ($\mu \in \{2, 4, 8\}$ pixels) and 6 equally spaced orientations ($\phi \in \{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ\}$), keeping only the response’s magnitude $\|g^w(x,y)\|$ as descriptor. Example responses of the filters applied to one frame of an expression track are shown in the right part of Fig. B.3.

To encode motion information, dense optic flow is computed at every frame by template matching, using the $L_1$-norm, considering 4 directions. Assuming that our expression tracks always start with a neutral face, we compute the optical flow both with respect to the previous frame (frame2frame) and to the first frame of the track (frame2first). In order to increase robustness to translation and to reduce the dimensionality of the feature space, both the shape and motion feature images are down-sampled by max-pooling, also known as winner-takes-all [Fukushima, 1980]:

$$h(x,y) = \max_{(i,j) \in \mathcal{G}(x,y)} g(i,j), \quad (B.3)$$

where $\mathcal{G}(x,y)$ denotes the neighborhood of pixel $(x,y)$. We use a window of size $(3 \times 3)$. 
B.5 Experiments

We trained and tested our facial expression recognition system on the Cohn-Kanade database [Kanade et al., 2000] and the MMI database [Pantic et al., 2005]. Both datasets contain videos of posed facial expressions, with subjects facing the camera and under controlled lighting conditions.

The Cohn-Kanade database consists of greyscale video sequences of 100 university students, 65% of which were female. The videos always start with a neutral face and end at the apex, i.e., the maximum intensity of the expression. For our study, we selected sequences which can be labeled as one of the basic emotions and which are longer than 13 frames, for a total of 344 videos depicting 97 subjects, each performing 1 to 6 facial expressions.

![Sample frames from Cohn-Kanade and MMI databases](image)

**Figure B.4**: Sample frames extracted from sequences depicting surprise in the Cohn-Kanade database (top) and MMI database (bottom). Note how the MMI database contains not only the transition from the neutral face to the apex of the expression, but also the offset leading back to the neutral state at end of the sequence.

The MMI database [Pantic et al., 2005] is a constantly growing, web-searchable set of color videos containing both posed and spontaneous emotions. We selected the subset of (posed) videos labeled as one of the six basic emotions, while discarding all others labeled only in terms of Action Units. The resulting set is comprised of 176 videos of 29 people displaying 1 to 6 expressions. The subjects differ in sex, age, and ethnic background; moreover, facial hair and glasses are sometimes present. The main difference between the MMI and Cohn-Kanade databases is that the MMI sequences do not end at the expression’s apex, but return to a neutral face. An example sequence from both dataset is shown in Fig. B.4, with the Cohn-Kanade at the top and MMI database at the bottom.

As explained in section B.4, both databases have been aligned to the eye center locations. For the Cohn-Kanade database, ground truth manual annotations are provided by [Lipori, 2010], while no such labeling is available for the MMI database, on which we use the eye tracking method of [Fanelli et al., 2009]. In both cases, the facial images are normalized to an inter-ocular distance of 25 pixels, resulting in $55 \times 45$ pixels images. Expression
tracks need to be labeled with both spatial and temporal center of the expression. The center in the image plane is assumed to correspond to the center of the face. The temporal center should ideally be located at the expression apex, therefore we take the last frame for the Cohn-Kanade database and the middle frame in the case of the MMI database. We train and test on all frames from the Cohn-Kanade dataset, which has an average sequence length of 18 frames, while selecting only 20 frames in the middle of each sequence for the MMI database, which has an average length of 79 frames.

For all of the following experiments, we performed subject-independent 5-fold cross validations, i.e., making sure that the same subjects did not occur in both training and test sets, and present here the results averaged over all five iterations. Forests always contained only 5 trees; indeed, adding more trees improved the results only slightly.

![Figure B.5](image)

**Figure B.5:** Left: Influence of the patch size on the overall recognition rate. Larger, rectangular patches, give the best results. Right: Recognition accuracy as a function of the number of \((20 \times 50 \times 2)\) patches sampled from each sequence during training.

![Figure B.6](image)

**Figure B.6:** Left: Confusion matrix for the Cohn-Kanade database. Expressions such as disgust and surprise are well recognized, while most of the confusion arises from the anger/disgust and fear/happiness classes. Right: Recognition rate for the Cohn-Kanade database, as a function of the percentage of occlusion.

Among the parameters of our proposed method are the size and shape of the patches. We ran some experiments varying the patches’ spatial size and shape, while keeping the number of patches fixed to 100 and the temporal dimension to 2 frames. In Fig. B.5, left,
the bars represent the recognition rate as a function of the size and shape of the sampled patches, as achieved on the Cohn-Kanade database. As can be noted, larger patches produce better results than smaller ones and rectangular shapes outperform squared ones. The best results (86.7%) are achieved with $20 \times 50$ patches, i.e., vertical rectangles covering almost half of the face.

Increasing the number of training patches per sequence did not influence much the recognition accuracy. Fig. B.5, right, shows that the accuracy increases only when moving from 100 to 200 patches, while it actually slightly decreases when more patches are used. We also tested the influence of the temporal length of the patches, but did not experience significant changes in the expression recognition accuracy. All results shown in the rest of the section are achieved by sampling 200 patches of size $20 \times 50 \times 2$.

Fig. B.6 left shows the confusion matrix obtained by our method when applied to the Cohn-Kanade dataset. On average, we recognize the correct expression 87.1% of the time; in particular, disgust is always correctly recognized. Fear and anger are the most confused labels, and are mainly mistaken for happiness, respectively disgust.

To assess the robustness of the method to partial occlusions, we removed (set to zero) the information in each feature channel falling under a cuboid. The cuboids are as long as the sequences, and cover a specific percentage of the image plane. For each sequence, the cuboid location on the 2D image plane was randomly chosen. We ran 5 trials for each percentage of occlusion, and present the averaged results in Fig. B.6, right. It can be noted how the performance decreases slowly as the occlusion becomes greater. At 15% occlusion, the accuracy is still around 70%, falling below 50% only when more than 30% of the face is removed. Sample frames help visualizing the amount of occlusion introduced.

Table B.1 lists our results next the performance of other methods which used the Cohn-Kanade database and which published their recognition rates for each label. As can be seen, the results are comparable.

<table>
<thead>
<tr>
<th></th>
<th>Our approach</th>
<th>Yeasin et al.</th>
<th>Buenaposada et al.</th>
<th>Aleksic &amp; Katsaggelos</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURPRISE</td>
<td>97.3%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>HAPPINESS</td>
<td>98.9%</td>
<td>96.6%</td>
<td>98.8%</td>
<td>98.4%</td>
</tr>
<tr>
<td>SADNESS</td>
<td>92.4%</td>
<td>96.2%</td>
<td>82.0%</td>
<td>96.2%</td>
</tr>
<tr>
<td>ANGER</td>
<td>62.2%</td>
<td>100.0%</td>
<td>78.4%</td>
<td>70.6%</td>
</tr>
<tr>
<td>FEAR</td>
<td>71.7%</td>
<td>76.4%</td>
<td>73.9%</td>
<td>88.2%</td>
</tr>
<tr>
<td>DISGUST</td>
<td>100.0%</td>
<td>62.5%</td>
<td>87.9%</td>
<td>97.3%</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>87.1%</td>
<td>90.9%</td>
<td>89.1%</td>
<td>93.6%</td>
</tr>
</tbody>
</table>

Table B.1: The results of our method are comparable with other works on automatic expression recognition. The accuracy is given for each expression class separately and on average.
Figure B.7: Left: Average recognition accuracy on the Cohn-Kanade database plotted against the single image features and their combinations. The optical flow between the current and the first frame gives the best results, followed by the Gabor filter responses and the frame to frame optical flow. Best results are achieved by the combination of all three kinds of features. Right: Accuracy for each class label, as recognized from the tracks created thanks to the ground truth annotation (cyan bars on the left) and automatically extracted by the eye tracker (magenta, right).

Figure B.8: Left: confusion matrix for the MMI database. The higher rate of confusion with respect to the results obtained on the Cohn-Kanade database can be partly explained by the fact that manual annotations of the eye locations were not available. Right: Results obtained on the MMI database using the set of features originally employed by [Yao et al., 2010a], with the addition of frame to frame and frame to first optical flow.

In an attempt to assess the contribution of each feature channel to the recognition, the left part of Fig. B.7 plots the accuracy achieved on the Cohn-Kanade database when each feature is used separately and in all their possible combinations. As can be seen, frame to first optical flow alone gives the best results, followed by log-Gabor responses and by optical flow computed between consecutive frames. The combination of all three features leads to the best results. In Fig. B.7, right, the performance for each class is plotted, depending on whether the tracks were extracted using the manual ground truth annotations of the eye locations (cyan bars on the left) or automatically, using the eye tracker (magenta, on the right). Results clearly worsen when the fully automatic method is employed, but not in
the same extent for each class: surprise, happiness, and sadness are less affected by errors in the tracking than the other classes.

When training and testing on the MMI database, again in a 5-fold cross-validation fashion and with 200 patches of size $20 \times 50 \times 2$, we get the confusion matrix shown in Fig. B.8, left. There is a higher rate of misclassification compared to the results achieved on the Cohn-Kanade database, especially for fear. This could be partly explained by the fact that manual annotations of the eye centers were not available, but also by the lack of a precise annotation of the expression center in the sequences. Also, the expressions in the MMI database are more subtle than in the Cohn-Kanade dataset. On average, our method achieves a recognition rate of 76% on the MMI database and, as far as we know, we are the first ones to attempt at classifying the expressions directly (rather than Action Units) on this dataset. The right side of Fig. B.8 shows the average results obtained on the MMI sequences when using the features originally proposed by [Yao et al., 2010a], with the addition of the two kinds of optical flow. The poor results of the original feature set serves as convincing support for the introduction of the log-Gabor filter responses, as explained in section B.4.1

**B.6 Conclusions**

In this paper, we investigated the use of a Hough forest voting method for facial expression recognition. Our system extends previous work aimed at action recognition to the field of facial expression recognition, which are more subtle and hard to classify. We chose features encoding separately form and motion of the face, which allow us to capture the subtle differences in the facial expressions which a standard action recognition system could not. We evaluated the system on two standard databases, Cohn-Kanade and MMI, and achieved results comparable to the state of the art. Future work includes the investigation of additional features and the application of the method to the recognition of more naturalistic facial expression videos.
C

Group Action Recognition

C.1 Introduction

Recognizing human actions from video has received much attention in the computer vision community, though designing algorithms that can detect and classify actions from unconstrained videos and in realistic settings still remains a challenge. Much of the work in action recognition so far has been focused on single persons. In applications such as intelligent surveillance, where the goal is to detect unusual or dangerous events, however, the classification of group interactions becomes more critical as situations can only be understood by considering the relationship between persons.

We present here a variation on a Hough-voting framework for action recognition, previously introduced in [Yao et al., 2010a] to classify group actions for the ICPR 2010 Contest on Semantic Description of Human Activities. We classify group actions by combining the classification results of single individuals to strengthen the group action response.

The rest of the paper is organized as follows. In Section C.2, we give a short summary of the Hough-voting framework described in [Yao et al., 2010a]. In Section C.3, we describe the combination of the classifier outputs of multiple people into group actions by using classifier combination rules and extending the model of decision profiles [Kuncheva et al., 2001]. In Section C.4, we show the classification results on group action recognition. Finally, Section C.5 summarizes the main results.

C.2 Hough-Voting Framework

The Hough-voting framework in [Yao et al., 2010a] takes a two-staged approach. In an initial localization stage, the person performing the action is tracked. Then, in a secondary classification stage, 3D feature patches from the track are used to cast votes for the action center in a spatio-temporal action Hough space. In [Yao et al., 2010a], a tracking-by-detection approach was used, though any other tracking method can be used as well since
the tracking stage is disjoint from the classification stage. For classifying the action, random trees are trained to learn the mapping between the patches and the corresponding votes in the action Hough space.

### C.2.1 Training

We train a random forest, which we term a “Hough forest”, to learn the mapping between action tracks and a Hough space. Each tree is constructed from a set of patches \( \mathcal{P}_i = (I_i, c_i, d_i) \), where

- \( \mathcal{P}_i \) is a 3D patch (e.g. of \( 16 \times 16 \times 5 \) pixels) randomly sampled from the track.
- \( I_i \) are extracted features at a patch and can be multi-channeled to accommodate multiple features, i.e. \( I_i = (I^1_i, I^2_i, \ldots, I^F_i) \in \mathbb{R}^F \), where each \( I^f_i \) is feature channel \( f \) at patch \( i \) and \( F \) is the total number of feature channels.
- \( c_i \) is the action label.
- \( d_i \) is a 3D displacement vector from the patch center to the action track center.

From the set of patches, the tree is built from the root by selecting a binary test \( t \), splitting the training patches according to the test results and iterating on the children nodes until either the maximum depth of the tree is reached or there are insufficient patches remaining at a node. Each leaf node stores \( p_c \), the proportion of the patches per class label reaching that leaf, and \( D_c = \{d_i\}_{c_i=c} \), the patches’ respective displacement vectors.

The binary tests compare two pixels at locations \( p \in \mathbb{R}^3 \) and \( q \in \mathbb{R}^3 \) in feature channel \( f \) with some offset \( \tau \), i.e.

\[
t_{f,p,q,\tau} (I) = \begin{cases} 
0 & \text{if } I^f(p) < I^f(q) + \tau \\
1 & \text{otherwise}
\end{cases}
\]  

(C.1)

First, a pool of binary tests with random values of \( f, p, q \) and \( \tau \) are generated; the test which splits the patches with minimal class or offset uncertainty between the split is chosen. By switching randomly between the two uncertainty measures, the leaves tend to have low variation in both class label and center displacement.

### C.2.2 Classifying and Localizing Actions

During test time, we extract densely sampled patches from the tracks and pass them through the trees in the Hough forest. Each patch arriving at a leaf votes into the action
C.3 Combining Classifiers for Group Action Recognition

In our setting of group action recognition, we distinguish between symmetric or asymmetric interactions. Symmetric interactions are those in which all individuals perform the same movements, such as shaking hands. Asymmetric interactions, on the other hand, are those in which the individuals behave differently. For example, when one person pushes another, there is an offender and a victim. We assume for simplicity that victims of all asymmetric actions behave in a similar way and add one generic victim class.

For each individual participating in an action, we get a single-person classification, and then combine them into group classifications using combination rules such as product rule, sum rule, min rule and max rule to strengthen the overall group response. A theoretical framework of these combination rules is given in [Kittler et al., 1998]. A convenient and compact representation of multiple classifier outputs is the decision profile matrix [Kuncheva et al., 2001] as the combination rules can be applied directly to the matrix. In the following, we review the model of decision profiles and extend them to handle both symmetric actions and asymmetric actions.

C.3.1 Decision Profiles

We define $c + 1$ single action labels, corresponding to $c$ group interactions and an additional victim label $v$. For each person $l$ in a group interaction of $L$ people, we have a single action classifier $D_l$, giving for each time instance $t$

$$D_l(t) = [d_{l,1}, \ldots, d_{l,c}, d_{l,v}],$$

where each $d$ corresponds to the support for a single action class. To combine the single action classifier outputs into group actions, we formulate a decision profile, $DP$, in matrix notation:
\[ DP(t) = \begin{bmatrix} D_1(t) \\ \vdots \\ D_l(t) \\ \vdots \\ D_L(t) \end{bmatrix} = \begin{bmatrix} d_{1,1} & \cdots & d_{1,c} & d_{1,v} \\ \cdots & \cdots & \cdots & \cdots \\ d_{l,1} & \cdots & d_{l,c} & d_{l,v} \\ \cdots & \cdots & \cdots & \cdots \\ d_{L,1} & \cdots & d_{L,c} & d_{L,v} \end{bmatrix}. \] (C.3)

For the combination of the single actions, the product, sum, min and max rule are directly applied to each column of the decision profile [Kuncheva et al., 2001].

### C.3.2 Extension for Asymmetric Group Actions

In our case, as we have added a victim class, we extend the above DP by dividing it into a symmetric and asymmetric block:

\[ DP(t) = [DP_{sym}(t) \mid DP_{asym}(t)], \] (C.4)

with \( DP_{sym}(t) \) as defined in Equation (C.3), but for single action labels belonging to symmetric group interactions only. To handle the asymmetric group interactions, we consider each combination of single actions which could form the interaction. Equation (C.5) describes the combination for a two-person scenario, but can be easily adapted for more people. Assuming \( m \) asymmetric group actions with classifier outputs \( d_{l,1}, \ldots, d_{l,m} \) and one victim class \( v \) with classifier output \( d_{l,v} \), the asymmetric decision profile would be a \( 2 \times 2 \cdot m \) dimensional matrix defined as follows:

\[ DP_{asym}(t) = \begin{bmatrix} d_{1,1} & d_{1,v} & d_{1,2} & d_{1,v} & \cdots & d_{1,m} & d_{1,v} \\ d_{2,v} & d_{2,1} & d_{2,v} & d_{2,2} & \cdots & d_{2,m} & d_{2,v} \end{bmatrix}. \] (C.5)

While Equation (C.5) is a redundant representation of the single action classifications, we choose this formulation as the same classifier combination rules can be directly applied to each column of the decision profile.

### C.4 Experiments

We demonstrate our approach of group action recognition on the UT-Interaction dataset [Ryoo & Aggarwal, 2010], consisting of six classes of two-person interactions shown in profile view: shake (hands), hug, kick, point, punch and push. We consider shake and hug as symmetric and the others as asymmetric interactions. For each class, there are two settings: set 1 recorded from a parking lot with a stationary background and set 2, recorded on a lawn with some slight background movement and camera jitter.
C.4.1 Single Person Actions

We first use the above-described Hough-voting method to classify the actions of single individuals. Tracks of the people were built with a Hough forest trained for people detection [Gall et al., 2011b] and a particle filter was used to assemble detections across time.

For simplification, only one classifier was trained for both the left and the right person; during testing, the classifier was applied to both the original and flipped version of the tracks and determined based on the higher response of the classifier if the person in the track stands on the left or right. Classification results for the seven single action classes are shown in Table C.1.

<table>
<thead>
<tr>
<th>Action</th>
<th>Set 1</th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left Track</td>
<td>Right Track</td>
</tr>
<tr>
<td>Shake</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>Hug</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>Kick</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Point</td>
<td>0.8</td>
<td>0.63</td>
</tr>
<tr>
<td>Push</td>
<td>0.33</td>
<td>0.72</td>
</tr>
<tr>
<td>Punch</td>
<td>0.66</td>
<td>0.86</td>
</tr>
<tr>
<td>Victim</td>
<td>0.77</td>
<td>0.73</td>
</tr>
<tr>
<td>Average</td>
<td>0.74</td>
<td>0.75</td>
</tr>
</tbody>
</table>

**Table C.1:** Classification performance of single actions according to track

C.4.2 Group Interactions

For evaluation of the group interactions, we use a leave-one-out cross validation for each set individually. Performance of the different combination rules are compared in Table C.2. Confusion matrices of the min-rule for set 1 and set 2 are shown in Figures C.1(a) and (b) respectively. Average performance of the best group classifier compared to the best single person classifier was higher by 13% in set 1 and 7% in set 2. The min rule performs well for both sets. The product and sum rule have similar performance in both sets, but are more affected by a weaker individual classifier as is the case in set 2 for right individual.

C.5 Discussion

The Hough-voting framework for action recognition, previously introduced in [Yao et al., 2010a], was applied to two very different action recognition scenarios and showed flex-
### Table C.2: Classification performance of group interactions for different fusion methods.

<table>
<thead>
<tr>
<th></th>
<th>Set 1</th>
<th></th>
<th>Set 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Product</td>
<td>Sum</td>
</tr>
<tr>
<td>Shake</td>
<td>0.5</td>
<td>0.4</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>Hug</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Kick</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Point</td>
<td>1.0</td>
<td>0.6</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Push</td>
<td>0.7</td>
<td>0.2</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Punch</td>
<td>0.8</td>
<td>0.1</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Average</td>
<td>0.83</td>
<td>0.55</td>
<td>0.87</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Bibliography


Chen, C.-C., Ryoo, M. S., & Aggarwal, J. K. (2010). UT-Tower Dataset: Aerial View Activity Classification Challenge. [http://cvrc.ece.utexas.edu/SDHA2010/Aerial_View_Activity.html](http://cvrc.ece.utexas.edu/SDHA2010/Aerial_View_Activity.html). 3.4.1, 3.5.4


CMU Mocap Database. [http://mocap.cs.cmu.edu/](http://mocap.cs.cmu.edu/). 5.3.2, 8.4.1


Lawrence, N. (2007). Learning for larger datasets with the gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 5.1, 5.3.2


Li, R., Tian, T.-P., Sclaroff, S., & Yang, M.-H. (2010). 3D human motion tracking with a coordinated mixture of factor analyzers. International Journal of Computer Vision (IJCV), 87, 170–190. 2.7, 4.1, 4.2, 5.1, 5.3.4, 5.3.4, 5.1, 5.3.4, 6.1, 7.1


Liu, J., Luo, J., & Shah, M. (2009b). Recognizing realistic actions from videos ”in the wild”. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2.4, 3.1, 3.5.1, 3.5.1, 8.1


Sigal, L., Balan, A., & Black, M. (2010). Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision (IJCV),* 87(1-2), 4–27. 4.1, 4.6.1, 4.6.3, 4.6.3, 5.1, 5.3.4, 5.3.4


Taylor, G., Sigal, L., Fleet, D., & Hinton, G. (2010). Dynamical binary latent variable models for 3D human pose tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.7, 4.2, 5.1, 5.3.4, 5.3.4, 5.3.4, 5.2, 5.3.4, 6.1, 7.1


Tenorth, M., Bandouch, J., & Beetz, M. (2009). The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *IEEE Workshop on THEMIS.* 1.1, 3.4.1, 3.5.5, 3.5.5, 4.1, 4.6.1, 6.3.4, 7.2


UCF50. http://server.cs.ucf.edu/~vision/data.html#UCF50. 2.4


Xu, X., & Li, B. (2007). Learning motion correlation for tracking articulated human body with a rao-blackwellised particle filter. In *Proceedings International Conference on Computer Vision (ICCV)*. 5.1, 5.3.4, 5.3.4


List of Publications

Journal Publications


Refereed Conference Proceedings


**Others**


Curriculum Vitae

Personal Data

Name Yingjie Angela Yao
Date of birth February 18th, 1983
Citizenship Canadian

Education

2008 - 2012 Doctor of Science ETH
Computer Vision Laboratory, ETH Zurich, Switzerland
2006 - 2008 Master of Science ETH in Biomedical Engineering
ETH Zurich, Switzerland
2001 - 2006 Bachelor of Applied Science in Engineering Science
University of Toronto, Canada
1999 - 2001 International Baccalaureate Diploma
Old Scona Academic High School, Edmonton, Canada

Work Experience

2008 - 2012 Computer Vision Laboratory, ETH Zurich, Switzerland
Research assistant
2004 - 2005 Honeywell Aerospace Engineering, Canada
Electrical engineering intern