DIMENSIONALITY REDUCTION FOR 3D
ARTICULATED BODY TRACKING AND HUMAN
ACTION ANALYSIS

By
Leonid Raskin
raskinl@cs.technion.ac.il

Research Supervisors:
Prof. Ehud Rivlin, Dr. Michael Rudzsky
and
Prof. Michael Lindenbaum

Submitted in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
at
Technion IIT - Israel Institute of Technology
Haifa, Israel
March 2010

© Copyright by Leonid Raskin, 2010
# Table of Contents

Table of Contents ........................................ ii  
List of Tables ........................................... vi  
List of Figures ........................................... viii  
Acknowledgements .......................................... xviii  
Glossary of Notation and Acronyms ......................... 1  

1 Introduction ............................................. 3  

2 Tracking and Action Classification: State of the Art .... 7  
2.1 Tracking ............................................... 7  
    2.1.1 Human Body Tracking ............................. 8  
    2.1.2 Human Body Models ............................... 11  
    2.1.3 Tracking Features ................................. 15  
    2.1.4 Dimensionality Reduction ......................... 17  
2.2 Action Recognition and Behavior Analysis ............. 19  
    2.2.1 Appearance-based action classification .......... 20  
    2.2.2 Model-based action classification ............... 27

3 Gaussian Process Annealed Particle filter .............. 39  
3.1 Filtering ............................................... 40  
    3.1.1 Particle Filter .................................. 40  
    3.1.2 Annealed Particle Filter ......................... 42  
3.2 Gaussian Process Models ................................ 44  
    3.2.1 Gaussian Process ................................. 44  
    3.2.2 Gaussian Process Latent Variable Model ........ 44
7 Conclusions and Future Work

7.1 Conclusions ................................................................. 121
7.2 Future Work .............................................................. 123
  7.2.1 Learning .............................................................. 123
  7.2.2 Generalization: Tracking New Actions Using the Existing Latent Space ............................................. 124
  7.2.3 Complex Action Detection and Analysis .................. 125
7.3 Application .............................................................. 125

Bibliography ................................................................. 127
List of Tables

3.1 The average errors for HumanEvaI (S1, walking1), HumanEvaII (S2) and HumanEvaII (S4) sequences produced by APF and GPAPF. The error measures the average distance in millimeters between the joint locations.

3.2 The average errors of the three algorithms: the top row shows the results of APF, the second row shows the results of GPAPF with two independent latent spaces (GPAPF1), and the third row shows the results of GPAPF with a joint latent space (GPAPF2). The columns represent the motion types: handshaking, kicking, and hugging.

4.1 Classification accuracy, using GPLVM based tracker, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

4.2 Classification accuracy, using GPDM based tracker, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

4.3 Classification accuracy, using the combined approach, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.
4.4 Classification accuracy for interactions between two people (using GPDM latent space) for 6 different activities: Hand shaking, Kicking, Hugging, Punching, Pushing, and Object passing. The rows represent the correct motion type; the columns represent the classification results.

5.1 The average errors for HumanEvaI(S1, walking1), HumanEvaII(S2) and HumanEvaII(S4) sequences produced by APF, GPAPF and HAPF. The error measures the average distance in millimeters between the joint locations.

5.2 The average errors of four algorithms: the top row shows the results of APF, the second row shows the result of GPAPF with two independent latent spaces (GPAPF1), the third row shows the result of GPAPF with a joint latent space (GPAPF2), and, finally, the last row shows the results of HAPF tracking algorithm. The columns represent the motion types: handshaking, kicking, and hugging.

6.1 The accuracies of the classification, using the hierarchical classification, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

6.2 Classification accuracy for interactions between two people (using GPDM latent space) for 6 different activities: Hand shaking, Kicking, Hugging, Punching, Pushing, and Object passing. The rows represent the correct motion type; the columns represent the classification results.

6.3 The accuracies of the classification, using the weighted classification, for different irregular activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.
List of Figures

2.1 Top: Description of algorithm for a torso-arm person model for 3 frames. We search for candidates in (a), enforce constant appearance by clustering the patches in (b). By connecting good clusters with valid kinematics, we learn the appearance of the torso and arm, which is used to find new candidates in (c) (reprinted from [118]).

2.2 Inferring attractive people: results for a walking cycle. Left: Initialization samples drawn from noisy simulated part detectors. Right: detector’s output (reprinted from [145]).

2.3 Human body models with kinematic model. Top: 2D model (reprinted from [66, 3]); bottom: 3D model (reprinted from [145, 38]).

2.4 (a) Graphical model. The model in (a) encodes the fact that each image instance of a segment has the same appearance (encoded in $C_{\text{seg}}$), but appears in different places. In (b), the undirected form of the model. A complete person model (c) contains additional kinematic dependencies. The full imaging model for a limited torso and arm subset is shown in (d). (reprinted from [118]).

2.5 Depth ambiguities when using monocular silhouettes [63].

2.6 Human body shape models. Commonly used volumetric models consist of (a) cylinders (reprinted from [135]), (b) tapered super-quadratics (reprinted from [50]); (c) surface-based models for the entire human body (reprinted from [23]).

2.7 (1) Probabilistic dependencies of actions. (Reprinted from [83, 176]).
2.8 Semantic interpretation of two-person interaction for hugging sequence (reprinted from [111]). 25

2.9 Graphical representation of (a) HMM model (b) linear-chain CRF (if the dotted lines exit, it will represent a model with a context of 3 observation timesteps); (c) two-chain factorial CRF (FCRF) including links between cotemporal labels, explicitly modeling limited probabilistic dependencies between two different label sequences; (d) HCRF (multi-class) model; (e) LPCRF model; (f) LDCRF model; (g) unrolled Graphical Model of the SFD-CRF for Pose Tracking and Recognition. 28

2.10 Using motion history for action classification. The two actions are recorded by multiple cameras, spatially integrated into their visual hulls (a), and temporally integrated into motion history volumes (b)(c). Invariant motion descriptors in Fourier space (d) are used for comparing the two actions. (Weinland et al. [31]) 31

2.11 Examples of actions from databases WEIZMANN (top) and KTH (bottom). Note that even a single frame is often sufficient to recognize what a person is doing. (reprint from [32]) 33

2.12 Overview of the embedding method: Two action sequences $Y$ (walk) and $Y'$ (jump forward on one leg) are matched against a set of silhouette exemplars $x_i$. For each exemplar the best matching frame in the sequence is identified (exemplar displayed on top of the corresponding frame; light colors correspond to high matching distances; dark colors to low matching distances). The resulting matching distances $d_i^*$ form vector $D^*$, which is interpreted as an embedding of the sequences into a low dimensional space $\mathbb{R}_n$. The final classifier is learned over $\mathbb{R}_n$, where each point represents a complete sequence. (Reprinted from [175]) 34
2.13 Silhouette contour of the projection from manifold space to image space. With the exception of a few degraded cases each motion sequence is recognizable despite only the first and last silhouettes of each sequence falling exactly on a projected data point. (Reprinted from [17])

2.14 (a) An illustration of 2-dimensional manifold embedding using LSTDE method for human silhouette sequences of multiple action classes; (reprint from [71]) (b) Trained Self Organizing feature Map (SOM) for action kick. (Reprinted from [104])

3.1 Annealed particle filter illustration for $M=5$. Initially the set contains many particles that represent very different poses and therefore can fall into local maxima. In the last layer, all the particles are close to the global maximum and therefore represent the correct pose.

3.2 Time series graphical model [167]. (a) GPLVM model. (b) Nonlinear latent-variable model for time series (GPDM). Because the mapping parameters $A$ and $B$ have been marginalized over, all latent coordinates $X = [x_1, x_2, ..., x_N]^T$ are jointly correlated, as are all poses $Y = [y_1, y_2, ..., y_N]^T$.

3.3 The 3D body model (a) and the samples drawn for the weighting function (b). In the right-hand image the blue samples are used to evaluate the edge matching, the cyan points are used to calculate the foreground matching, and the rectangles with edges on the red points are used to calculate the part-based body histogram.

3.4 The reference histograms of the torso: (a) red, (b) green and (c) blue colors of the reference selection.

3.5 The latent space that is learned from different poses during the walking sequence. (a) the 2D space; (b): the 3D space. On the image (a): the brighter pixels correspond to more precise mapping.
3.6 The latent space that is learned from different motion types. (a) 2D latent space from 3 different motions: lifting an object (red), kicking with the left (green) and the right (magenta) legs. (b) 3D latent space from 3 different motions: hand waving (red), lifting an object (magenta), kicking (blue), sitting down (black), and punching (green).

3.7 Losing and finding the tracked target despite the miss-tracking on the previous frame. Top: camera 1, Bottom: camera 4.

3.8 The latent spaces that were generated for the 3 different actions (left leg kicking, right leg kicking and object lifting) by (a) GPLVM and (b) GPDM with back constraints.

3.9 The dynamics prior generated by GPDM: (a) the walking sequence dynamics; (b) 3-action (kicking with the left leg, kicking with the right leg, and picking up an object) dynamics.

3.10 The average errors for the walking sequences for both GPLVM and GPDM algorithms with (a) 500 particles (100 per layer) and (b) 100 particles (20 per layer). The red crosses represent the GPLVM based tracker errors and the blue circles represent the GPDM based tracker.

3.11 GPAPF with additional annealing layer. The black solid arrows represent the dependencies between state and visual data; the blue arrows represent the dependencies between the latent space and the data space; dashed magenta arrows represent the dependencies between sequential annealing layers; the red arrows represent the dependencies of the additional annealing layer. The green arrows represent the dependency between sequential frames.

3.12 GPAPF algorithm with (top) and without (bottom) adding additional annealed layer.

3.13 The errors of the GPAPF tracker with additional annealing layer (blue circles) and without it (red crosses) for a walking sequence captured at 30 fps.
3.14 The errors of the annealed tracker (red crosses) and the GPAPF tracker (blue circles) for a walking sequence captured at 30 fps.


3.16 GPAPF tracking results. Sample frames from the boxing2 sequence from the HumanEvaI (S1) dataset.

3.17 GPAPF tracking results. Sample frames from the combo1 sequence from the HumanEvaII (S2) dataset.

3.18 GPAPF tracking results. Sample frames from the running, leg movements, and object lifting sequences.

3.19 The errors of the annealed tracker (red crosses) and the GPAPF tracker (blue circles): (top) HumanEvaI (S1, walking1, frames 6-590); (middle) HumanEvaII (S2, frames 1-1202) (marked by blue stars); (bottom) HumanEvaII (S4, frames 2-1258) (marked by red circles).

3.20 Successive GPAPF tracking results for 2 persons interaction. Sample frames from the *kicking* sequence. Each row corresponds to a different scene view (different camera).

3.21 Failure of GPAPF tracking results for 2 persons interaction. Sample frames from the *hugging* sequence. Each row corresponds to a different scene view (different camera).

3.22 The error graphs for *handshaking* (top), *kicking* (middle), and *hugging* (bottom) actions. The errors produced by the APF tracker are marked by green crosses, the errors of the GPAPF1 tracker are marked by blue circles and the errors of the GPAPF2 tracker are marked by magenta squares.
4.1 Visualization of the classification task with (a) GPDM and (b) GPLVM latent spaces. The latent space contains 3 different action templates: lifting an object (red), kicking with the left (yellow) and the right (magenta) legs. The blue line represents the tracker output; the cyan dots represent the actual locations in the latent space, as estimated by the tracker. The classification task is to find the template closest to the tracker’s result.

4.2 Graphical illustration of the action template (black line), denoted by $\Psi_k = \{\psi_0, ..., \psi_l\}$ and a sequence produced by the tracker, (red line), denoted by $\Upsilon = \{\varphi_0, ..., \varphi_l\}$.

4.3 Tracking trajectories in the latent space for different activities: (a) lifting an object, kicking with (b) the left and (c) the right legs. The first row represents the GPLVM based tracker; the second row represents the GPDM based tracker; the third row represents the combined approach: the GPDM based tracker locations are transformed to the data space and back-projected on the GPLVM latent space. In each image the black lines represent the templates and colored lines represent the trajectories produced by the GPAPF tracker.

5.1 A hierarchy for modeling interactions between two subjects. $Y_1$ and $Y_2$ is the data associate with each subject and correspond to latent variables $X_1$ and $X_2$. These latent variables are controlled by $X_3$, which model the interaction between the subject.

5.2 The hierarchy structure for a single subject (on the left) and the corresponding latent spaces (on the right). Each latent space corresponds to a set of body parts (called partial pose) and is learned for five different motion types: hand waving (red), lifting an object (blue), kicking (green), sitting down (yellow), and punching (cyan).
5.3 The hierarchy structure for two-subjects modeling. The hierarchy has 4 layers: the top (root) layer represents both subjects; the second layer describes each subject individually; the third layer describes the groups of body parts of each subject (for example the upper body of subject 1); the last layer describes a single body part for each subject.


5.5 The errors of the annealed tracker (green crosses), the GPAPF tracker (blue circles) and HAPF tracker (red stars) for a walking sequence captured at 30 fps.

5.6 HAPF tracking results. Sample frames from the ThrowCatch1 sequence (throwing part) from the HumanEvaI (S2) dataset.

5.7 HAPF tracking results. Sample frames from the ThrowCatch1 sequence (throwing part) from the HumanEvaI (S2) dataset.

5.8 HAPF tracking results. Sample frames from the combo2 sequence (walking example) from the HumanEvaII (S4) dataset.

5.9 HAPF tracking results. Sample frames from the combo2 sequence (jogging example) from the HumanEvaII (S4) dataset.

5.10 HAPF tracking results. Sample frames from the combo2 sequence (balancing example) from the HumanEvaII (S4) dataset.

5.11 The errors of the annealed tracker (green crosses), the GPAPF tracker (blue circles), and the HAPF tracker (red stars): (top) HumanEvaI (S1, walking1, frames 6-590); (middle) HumanEvaII (S2, frames 1-1202); (bottom) HumanEvaII (S4, frames 2-1258).
5.12 Sample frames showing the motion type transition (from jogging to balancing) in HumanEvaII(S4) dataset.  
5.13 HAPF tracking results for irregular action ”punching with raised leg”.  
5.14 HAPF tracking results for irregular action ”waving with both arms and with raised leg”.  
5.15 HAPF tracking results for irregular action ”kicking and raising up the knee with raised arm”.  
5.16 Tracking results of HAPF tracker. Sample frames from the handshaking sequence.  
5.17 Tracking results of HAPF tracker. Sample frames from the kicking sequence.  
5.18 Tracking results of HAPF tracker. Sample frames from the hugging sequence.  
5.19 The error graphs for handshaking (top), kicking (middle), and hugging (bottom) actions. The errors produced by APF tracker are marked by green crosses, the errors of GPAPF1 tracker are marked by blue circles, the errors of GPAPF2 tracker are marked by magenta squares, and the errors of the HAPF tracker are marked by red stars.  
6.1 Tracking trajectories of sitting down action projected to different latent spaces: (a)hierarchy level 1, latent space 1; (b) hierarchy level 2, latent space 3; (c)hierarchy level 3, latent space 5. The green line represents the correct model (a model of sitting down action), the red lines represent the incorrect models (models of waving, punching, kicking and picking an object actions), and the colored lines represent the trajectories produced by the tracker. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker.
6.2 The latent and tracking trajectories of *punching* action projected to these latent spaces: (a) hierarchy level 3, latent space 1 (representing the left leg); (b) hierarchy level 3, latent space 2 (representing right arm). The green line represents the correct model (a model of *punching* action), the red lines represent the incorrect models (models of *waving*, *sitting down*, *kicking* and *picking an object* actions), and the colored lines represent the trajectories produced by the tracker during irregular performance of the punching action. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker.
Acknowledgements

This research thesis was done under the supervision of Prof. Ehud Rivlin and Dr. Michael Rudzsky in the Department of Computer Science, Technion.

First, I would like to thank my supervisors. Above all, Prof. Ehud Rivlin and Dr. Michael Rudzsky who have encouraged me throughout my Ph.D. and gave me the freedom to form my research in the direction interesting for me. Specially, I would like to thank Michael for helping me to develop my skills during these years, for teaching me all about computer vision and machine learning. The importance of the discussions that we had, the reading material that he has supplied and countless hours he has spent with me cannot be overestimated. I would like to thank Ehud for taking me as a fresh graduate and changing and developing my research vision. I am grateful to both my supervisors for their patient guidance and help, and for believing in me throughout the ups and downs of this research. In addition I have to thank Ehud for a generous financial support throughout all these years.

I wish to express my deepest gratitude to Prof. Micha Lindenbaum, who agreed to become my advisor during Ehud’s leave.

I would like to thank the many people who worked and taught with me throughout these years. I am deeply grateful to Prof. Hagit Attiya, Prof. Roy Friedman, Prof. Assaf Schuster and Dr. Erez Hadad for helping me to be a better teaching assistant and later a lecturer. Specially, I want to thank Hagit for believing in me in the first
place. Also I want to thank Dr. Gabi Kliot, Dr. Alexander Landau, Anastasia Braginsky, and many others with whom I have had pleasure to work during the last five years.

The generous financial support of the Technion, Vivian Konigsberg, Sandor Szego, and the Swartz Fund is gratefully acknowledged.

I would like to thank my wife, Assya Raskin. Without her editorial and mental interference I would not be able to finish this Ph.D. I would like to thank her for supporting me all these years, including taking care of our son, Eran, which allowed me to study. Last but not least, I want to thank my parents who have been supporting me all my life. This thesis is devoted to them.
Glossary of Notation and Acronyms

We include here the notation and abbreviations used in this work.

- $x_n$ - A hidden state vector.
- $y_n$ - A measurement in time $n$.
- $w_n(y_n, x)$ - A weighting function.
- $\Lambda$ - 3D location: rotation and translation.
- $\Omega$ - Pose vector.
- $\Omega_{h,l}$ - $l$-th subspace in the hierarchy level $h$.
- $\Theta_{h,l}$ - $l$-th latent space in the hierarchy level $h$.
- $\wp_{(h,l)}$ - Mapping function from $\Theta_{h,l}$ to $\Omega_{h,l}$.
- $\theta_{h,l}$ - A coordinate in the $l$-th latent space in the hierarchy layer $h$.
- $\omega_{h,l}$ - Partial pose vector corresponding to $l$-th subspace in the hierarchy level $h$.
- $\lambda_{h,l,n}$ - Location in hierarchy layer $h$ in the latent space $l$ in the frame $n$.
- $\omega_{h,l,n}$ - Full pose in hierarchy layer $h$ in the latent space $l$ in the frame $n$.
- $\theta_{h,l,n}$ - Latent coordinates in hierarchy layer $h$ in the latent space $l$ in the frame $n$.
- $\pi_{h,l,n}^{(i)}$ - Model configuration data.
- $S_{h,l,n}^{(i)}$ - Weighted particle on frame $n$ in hierarchy layer $h$ in the latent space $l$.
- $\pi_{h,l,n}$ - The particle weight on frame $n$ in hierarchy layer $h$ in the latent space $l$.
- $M_k$ - A model of $k$-th action.
- $M_{h,l,k}$ - A model of $k$-th action in hierarchy layer $h$ in the latent space $l$. 
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF</td>
<td>Annealed Particle Filter</td>
<td>2.1</td>
</tr>
<tr>
<td>BN</td>
<td>Bayesian Networks</td>
<td>2.2.1</td>
</tr>
<tr>
<td>CHMM</td>
<td>Coupled Hidden Markov models</td>
<td>2.2.2</td>
</tr>
<tr>
<td>CRF</td>
<td>Conditional Random Fields</td>
<td>2.2.1</td>
</tr>
<tr>
<td>DoF</td>
<td>Degrees of Freedom</td>
<td>2.1.2</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
<td>2.2.2</td>
</tr>
<tr>
<td>FCRF</td>
<td>Factorial Conditional Random Fields</td>
<td>2.2.1</td>
</tr>
<tr>
<td>GPAPF</td>
<td>Gaussian Process Annealed Particle Filter</td>
<td>3.3</td>
</tr>
<tr>
<td>GPLVM</td>
<td>Gaussian Process Latent Variable Models</td>
<td>3.2.2</td>
</tr>
<tr>
<td>GPDVM</td>
<td>Gaussian Process Dynamical Model</td>
<td>3.2.3</td>
</tr>
<tr>
<td>HAPF</td>
<td>Hierarchical Annealed Particle Filter</td>
<td>5.3</td>
</tr>
<tr>
<td>HCRF</td>
<td>Hierarchical Conditional Random Fields</td>
<td>2.2.1</td>
</tr>
<tr>
<td>HGPLVM</td>
<td>Hierarchical Gaussian Process Latent Variable Models</td>
<td>5.2</td>
</tr>
<tr>
<td>HHBM</td>
<td>Hierarchical Human Body Model</td>
<td>5.3</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
<td>2.2.1</td>
</tr>
<tr>
<td>HVT-HMM</td>
<td>Hierarchical Variable Transition Hidden Markov Model</td>
<td>2.2.1</td>
</tr>
<tr>
<td>KNN</td>
<td>K-Nearest Neighbor</td>
<td>2.2.2</td>
</tr>
<tr>
<td>KPCA</td>
<td>Kernel Principal Component Analysis</td>
<td>2.2.1</td>
</tr>
<tr>
<td>LD-CRF</td>
<td>Latent-Dynamic Conditional Random Fields</td>
<td>2.2.1</td>
</tr>
<tr>
<td>LLE</td>
<td>Locally Linear Embedding</td>
<td>2.1.4</td>
</tr>
<tr>
<td>LPCRF</td>
<td>Latent-Pose Conditional Random Fields</td>
<td>2.2.1</td>
</tr>
<tr>
<td>LPP</td>
<td>Locality Preserving Projections</td>
<td>2.2.2</td>
</tr>
<tr>
<td>LSTDE</td>
<td>Local Spatio-Temporal Discriminant Embedding</td>
<td>2.2.2</td>
</tr>
<tr>
<td>MEMM</td>
<td>Maximum Entropy Markov Model</td>
<td>2.2.1</td>
</tr>
<tr>
<td>MHV</td>
<td>Motion History Volumes</td>
<td>2.2.2</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
<td>2.2.2</td>
</tr>
<tr>
<td>MM</td>
<td>Markov Model</td>
<td>2.2.1</td>
</tr>
<tr>
<td>NLDS</td>
<td>Non-Linear Dynamical Systems</td>
<td>2.2.2</td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbor</td>
<td>2.2.2</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
<td>2.1.4</td>
</tr>
<tr>
<td>PCFG</td>
<td>Probabilistic Context-Free Grammar</td>
<td>2.2.1</td>
</tr>
<tr>
<td>PF</td>
<td>Particle Filter</td>
<td>3.1.1</td>
</tr>
<tr>
<td>PPCA</td>
<td>Probabilistic Principal Component Analysis</td>
<td>3.2.2</td>
</tr>
<tr>
<td>RVM</td>
<td>Relevance Vector Machines</td>
<td>2.2.2</td>
</tr>
<tr>
<td>SFD-CRF</td>
<td>Shape,Flow,Duration-Conditional Random Fields</td>
<td>2.2.2</td>
</tr>
<tr>
<td>SGPDM</td>
<td>Switching Gaussian Process Dynamical Model</td>
<td>2.1.4</td>
</tr>
<tr>
<td>SMILE-SVM</td>
<td>Multiple Instance LEarning Support Vector Machines</td>
<td>2.2.2</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines</td>
<td>2.2.2</td>
</tr>
<tr>
<td>VTHMM</td>
<td>Variable Transition Hidden Markov Model</td>
<td>2.2.1</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction

Tracking humans, understanding their actions and interpreting them are crucial to a great variety of applications. Tracking is used in automated surveillance, human-computer interface applications and in security applications.

During the last decade extended Analysis of human interactions is a complicated and challenging task for several reasons. First, the large number of body parts makes it hard to detect each part separately. Second, the variability in clothing style and illumination conditions add to the already great variety of images. In order to achieve a satisfactory understanding of pose, one has to resolve the ambiguity caused by human body articulation. In addition, there are difficult technical issues, such as tracking in occluded environment, image segmentation and feature extraction, which increase the number of aspects that have to be considered in this kind of analysis. Finally, the camera synchronization task is hard to be solved by software. Therefore, in many cases this problem is solved by using special hardware equipment, which allows receiving synchronized streams from several cameras. The focus of our research is to provide a robust algorithm for 3D body part tracking and action classification.

The human poses can be represented in high dimensional spaces. Despite the high dimensionality of the problem, many poses can be presented in a low dimensional space by dimensionality reduction. The human actions can be described as curves in this space. This space can be obtained by learning different motion types [61, 37, 159, 123].
We develop an algorithm for 3D human body part tracking. In this approach we apply a nonlinear dimensionality reduction using Gaussian Process Dynamical Model (GPDM) \cite{GPDM} on annealed particle filter \cite{APF}. GPDM comprises a mapping from a latent space to the data space, and a dynamical model in the latent space, and, therefore, is more suitable for tracking than the linear methods such as PCA, which create a mapping from the data space to the latent space (for the discussion about the advantages of non linear methods see Section 2.1.4). This method generates a mapping function from the low dimensional latent space to the full data space based on learning from previously observed poses from different motion types. For the tracking purpose we separate model state into two independent parts: one contains information about 3D location and orientation of the body and the second one describes the pose. We learn latent space that describes poses only. Our method is a variant of the particle filter (see Section 3.1). The tracking algorithm consists of two stages. Firstly, the particles are generated in the latent space and are transformed into the data space by using learned a priori mapping function. Secondly, we add rotation and translation parameters to obtain valid poses. The likelihood function calculated in order to evaluate how well a pose matches the visual data. The resulting tracker estimates the locations in the latent space that represents poses with the highest likelihood.

As the latent space is learned from sequences of poses from different motion types. The action is represented by a template: the ordered set of the coordinated in this latent space. The classification of an action can be done based on the comparison of the sequences of latent coordinates, produced by the tracker, and the templates that represent different motion types. We suggest using a modified Fréchet distance \cite{Fréchet} in order to compare the sequences of the latent coordinates. The Fréchet distance of two curves measures the resemblance of the curves taking into consideration their direction. This method is quite tolerant of position errors and enables comparing the portion of a motion to the model. This approach allows introducing a different action from the ones we have used for learning by exploiting the curve that represents it.

The drawback of the GPAPF algorithm is that a latent space is not capable of
describing all possible poses. The particles generated in the latent space describe only the poses that resemble those used during the learning process. However, if a person performs a new movement which differs from those already learned, then the new poses will be represented less accurately by the latent space.

In order to be able to deal with all the stated above difficulties we propose Hierarchical Anneal Particle Filter (HAPF). Instead of using a single latent space to describe the human body motions we use a hierarchy of the latent spaces. The root node describes the whole body, while the leaves describe a individual body parts. This hierarchical decomposition of the body makes the algorithm capable of a recovery of novel poses that are not present in the training set. The algorithm estimates pose by using different hierarchy levels, rather then a single latent space describing the whole body. This allows us to generate particles that represent not only the poses similar to the ones used in the training set, but also their combinations, such as "waving and kicking", "running with a raised arm" etc.

The classification using the hierarchical representation of human body can be done in two ways. The first way is to compare the sequences of the latent coordinates on each latent space and to find the action with the closest average distance. However this approach is not robust for the variations in the motion types and thus it would not be able to classify irregular motion described in the previous paragraph. Therefore we suggest additional classification algorithm, that assigns weight for each root of the hierarchy. This allows to model the dependency of the motion on a movement of a single or a group of body parts.
Chapter 2

Tracking and Action Classification: State of the Art

2.1 Tracking

The goal of the human body poses tracking poses is to find a set of pose parameters that minimizes the error between the observation and the body model (see Section 2.1.2). We can identify two main classes for body pose estimation. The first one is the body detection and estimation, which is based on a single frame [167, 67, 96, 91, 7, 59]. The second approach is the body pose tracking which approximates body pose based on a sequence of frames [2, 3, 141, 33, 143]. A variety of methods have been developed for articulated human body tracking from single views [117, 53, 149], as well as multiple views [38, 35]. Additional difficult problem in human pose tracking, that has been addressed in previous years, is tracking in an occluded environment [51, 39, 94].

Tracking is used in order to ensure temporal coherence between sequential pose estimations. Traditional tracking algorithms, such as Kalman filtering and local optimization methods [21, 164, 72], were performed using a single hypothesis of the human body. Cham et al. [25] use a set of Kalman filters, which results a more reliable tracking. Kalman filter is the optimal linear estimator of the Gaussian problems, which is not necessary the case in the human body part tracking, where there is a non-linear dependency between the parameters. Moreover, using a single hypothesis the estimation is likely to lose the track, for example in case of ambiguity which is
caused by the occlusion. Therefore, most recent works propagate multiple hypothesis. One of the common multiple hypothesis approaches for tracking is using a Particle Filtering. This method uses multiple predictions, obtained by drawing samples of pose and location priors and then propagating them using the dynamic model, which are refined by comparing them with the local image data, calculating the likelihood (see, for example Isard and MacCormick [69], Bregler and Malik [20] or Perez et al. [112]). Mei and Ling [88] represent tracking target candidate in the space spanned by target templates. The sparsity is achieved by solving an L1-regularized least squares problem. The candidate with the smallest projection error is taken as the tracking target. Particle filter (PF) is used for propagating sample distributions over time. The prior is typically quite diffused (because motion can be fast) but the likelihood function may be very peaky, containing multiple local maxima, which are hard to account for in detail. For example, if an arm swings past an "arm-like" pole, the correct local maximum must be found to prevent the track from drifting (Sidenbladh et al. [142]). Annealed particle filter (APF) (Deutscher and Reid [38, 37], Balan et al. [12]) or local searches are the ways to attack this difficulty. An alternative is to apply a strong model of dynamics (Mikolajczyk et al. [92]). Mikami et al. [90] proposed the Memory-based Particle Filter (M-PF) capable of tracking moving objects with complex dynamics. They eliminate the Markov assumption from the particle filtering framework and predict the prior distribution of the target state from the long-term dynamics. The method can handle nonlinear, time-variant, and non-Markov dynamics, which is not possible within the PF frameworks.

2.1.1 Human Body Tracking

One can identify two main approaches to the body pose estimation problem: the top-down and bottom-up.

The top-down approaches match a projection of the human body pose vector with the observations. A local search can be performed close to the initial estimation [50, 21], but it is computationally expensive. One way to deal with this problem is to perform a gradient descent [164]. Dellamare et al. [36] use gradient descent to achieve best matching with the projected and extracted silhouettes to refine pose estimation. A drawback of the top-down approaches is that they require manual initialization.
In addition, top-down approaches cause problems with occlusion or self occlusion. Moreover, errors are propagated through the kinematics chain [38]. To overcome this problem Drummond et al. [40] introduce constrains between linked body parts in the kinematics chains, which allow lower part to effects the parts, which are higher in the chain. Fossati et al. [47] exploit the fact that the trajectories of a persons feet or hands strongly constrain body pose in motions such as skating, skiing, or golfing. They treat these as a latent variables and learn a mapping between them and sequences of body poses. In this manner, they can reliably guess initial poses over whole sequences and, then, refine them.

In the bottom-up approaches individual body parts locations are estimated and then assembled into a full body. The body parts are usually described by 2D templates [116]. There is a need for a part detector for most of the body parts. The drawback of such detectors is that they produce many false positive detections. Assembling process takes in consideration physical and temporal constrains. Bottom-up approaches have the advantage that they do not require manual initialization. For example Mori et al. [97] perform image segmentation, based on the contours and shape, which are classified later as body parts, forming a partial configuration for the body pose. Global constrains, such as part proximity, color similarity, lengths, are used to detect missing body parts. A similar in spirit approach was introduced by Ren et al. [133]. They use pairwise edges to find the segments. Ramanan [118] use a simple appearance based detector to detect the body parts. In [119] Ramanan et al. train the models that maximize the likelihood for joint position of the body parts, instead a separate model for each body part. Ioffe and Forsyth [67] model an appearance of each body part. Inference is used on a mixture of the trees. Sigal et al. [145] describe human body as a graphical model. Each node of the model represents a parameterized body part and the spatial constrains are modeled as arcs. Each node has associated likelihood function, that measures the probability to observe the part, based on image observations. The body estimation is, therefore, formulated as inference in the graphical model. In [143] a temporal constrains are also taken in consideration. In [144] Sigal and Black introduce occlusion-sensitive likelihood functions.

By combining these two techniques the drawbacks of both of them can be solved. Navaratnam et al. [100] uses search-space decomposition method. Body parts lower
Figure 2.1: Top: Description of algorithm for a torso-arm person model for 3 frames. We search for candidates in (a), enforce constant appearance by clustering the patches in (b). By connecting good clusters with valid kinematics, we learn the appearance of the torso and arm, which is used to find new candidates in (c) (reprinted from [118]).

Figure 2.2: Inferring attractive people: results for a walking cycle. Left: Initialization samples drawn from noisy simulated part detectors. Right: detector’s output (reprinted from [145]).
in the kinematics chain are found using the part detectors, within the image region that is defined by the parent in the kinematics chain. Though this method is more efficient, it highly depends on the part detectors. Hua et al. [65] incorporates bottom-up information in a statistical framework. The human body modeled as a Markov network. 2D poses are inferred using Monte Carlo belief propagation approach. Sminchisescu et al. [148] learn top-down and bottom-up functions in alternate steps. The top-down process uses samples from the bottom-up one, which in its turn is optimized to produce estimations close to the ones produced by the top-down step. Baak et al. [10] introduced an iterative tracking approach that dynamically integrates motion priors retrieved from a database to stabilize tracking. The idea is to pursue a joint bottom-up and top-down strategy in the sense of starting with a rough initial tracking which is then improved by incorporating high-level motion cues. The resulting 3D motion sequences, which may be corrupted due to tracking errors, are locally classified according to available motion categories. Depending on the classification result, a retrieval system supplies suitable motion priors, which are then used to regularize and stabilize the tracking in the next iteration step.

2.1.2 Human Body Models

Human body models describe both the kinematics properties of the body (the skeleton) and the appearance (clothes).

**Kinematics Models**

Most of the models describe human body as a kinematics tree, consisting of segments that are linked by joints. Every joint has a certain number of degrees of freedom (DoF). The pose representation is formed by all the DoF of all the joints of the model.

The number of DoF that are used varies between the studies. However, we can distinguish two major classes of the kinematics models: 2D models and 3D models. The first ones are used for describing a planar motion. For example, Haritaoglu et al. [58] use a so-called Cardboard model to model the limbs as planar patches. Each segment has 7 parameters that allow it to rotate and scale according to the 3D motion. Huang and Huang [66] add an extra patch width parameter to account for scaling
during in-plane motion. Aggarwal and Triggs [1], as shown in Figure 2.3.a, the human body is described by a 2D scaled prismatic model. They assume a constant angle of view of the subject. However, because of the restricting assumptions the resulting trackers are not capable of tracking general human poses.

Consequently, many researchers choose a general kinematics model allowing three (orthogonal) rotations per joint. 3D approaches usually model the body as a set of rigid segments, which allow up to 3 DoF (rotational) for each joint. For each DoF kinematics constrains can be imposed. For instance, Sigal et al. [145] model the relationships between body parts as conditional probability distributions. Bregler et al. [21] introduce a twist motion model and exponential maps which simplify the comparison between image motion and model motion. The kinematic DOF can be recovered robustly by solving simple linear systems under scaled orthogonal projection. The parameters of the kinematic model such as limb lengths are sometimes assumed fixed. However, due to the large variability among people, this will lead to inaccurate
pose estimations. Alternatively, these parameters can be recovered in an initialization step where the observed person is to adopt a specified pose [23, 14]. While this approach works well for many applications, it restricts use in surveillance or automatic annotation systems. Online adjustment of these parameters is possible by relying on statistical priors [53] or specific but common key poses [16]. Some models [37] use 30 DoF to describe a body, while the others [1] use more than 50 DoF. However, even in the models with a limited number of DoF, the number of possible poses is very high. Therefore, applying kinematics constants is an effective way to eliminate the infeasible poses. Typical constrains are joint angle limits [164] angular velocity and acceleration limits [178]. For example, Rohr [136] restricts the range of movement of the subject. The assumption is that the subject is performing a specific action. While it can help exclude impossible postures, it cannot solve the ambiguities that stem from occlusion. As it is shown in Fig. 2.5 there are many different poses that produce very similar 2D projection, which makes a monocular tracking less effective. This problem can be partially solved by using several cameras, which allows to obtain different 2D projections of a human body. The assumption is that while two different poses may have same or very similar projection on some 2D plane, they are expected to have different projections on other 2D planes. Having multiple cameras in the system results being able to use projections of the same pose to different planes and
Figure 2.5: Depth ambiguities when using monocular silhouettes [63].

thus being able to distinguish between poses that may appear similar in a monocular system.

**Shape Models**

In addition to the kinematics models, the human shape is also modeled. In 3D models the segments are either volumetric or surface-based. Commonly used volumetric models consist of cylinders [60, 135, 141] or tapered super-quadratics [50, 74, 36]. Instead of modeling each segment as a separate rigid shape [23], surface-based models usually use a single surface models for the entire human body and typically consist of a mesh of polygons [72, 13]. While in some models [37, 120] the parameters like limbs length and width are assumed to be invariant, in other models these parameters are estimated during the initialization [23]. Gall et al. [49] proposed a two-pass approach: in the first pass, a skeleton is geometrically fit into the visual hull for each frame; the second pass deforms a template model according to the estimated skeleton and refines the template to fit the silhouettes.

Most of the tracking techniques [142, 143, 124] require a knowledge of the prior for a human body motion. Unfortunately, it is not feasible to manually design such models for several reasons. The most important one is that because of the high dimensionality of the problem it is hard to understand the dependencies between the
Figure 2.6: Human body shape models. Commonly used volumetric models consist of (a) cylinders (reprinted from [135]), (b) tapered super-quadratics (reprinted from [50]); (c) surface-based models for the entire human body (reprinted from [23]).

Several algorithm rely on an assumption of a smoothed movements. For example, a particle filter based tracker suggests that the location of the object of interest between two sequential frames can differ only to a certain degree. Usually such priors are derived using first or second order Markov models. However, because a human motion often consist of abrupt orientation and speed changes, such models a practically not effective for body parts tracking. As mentioned above these models can be further improved using the physiological limits between the body joints.

2.1.3 Tracking Features

In order to be able to track an object in general and human body in particular one has to use different image features, rather than use the image as it is. Often these image descriptors include one or several of the ones described in this section.
Silhouettes

One of the most popular features is the silhouettes. It can be extracted relatively easily, it is robust to color and texture variations and gives very important information for the 3D reconstruction of the body [1]. However, the quality of the extraction is also limited, because of shadows, changes in the lighting conditions and occlusions. The other drawback is that it contains ambiguities due to the lack of the depth information [63].

Another popular feature is the edges. They can be extracted robustly and are invariant to light conditions and changes of the illumination, but unsuitable in an occluded environment. A textured clothing makes it hard to use this feature as well. Usually, the edges are used either in a region of the interest or inside the extracted foreground image [73, 164].

The drawback of the silhouettes and the edges is that they do not contain any depth information, which makes it hard to operate in an occluded environment. As mentioned above, one way to handle the occlusions is to use multiple cameras and to use projections on several different 2D planes. The other approach, which also requires usage of multiple cameras, is a 3D reconstruction. It is done using the silhouettes that are extracted from different views. Two common approaches are voxel-based [22, 29] and volume intersection [19]. In addition one can reconstruct the 3D information based on the stereometry [114, 58]. Ukita et al. [156] proposed a method for estimating the pose of a human body using its approximate 3D volume (visual hull) obtained in real time from synchronized videos. They learn the probabilistic dynamical model of human volumes from training temporal volumes refined by error correction. The dynamical model of a body pose (joint angles) is also learned with its corresponding volume. By comparing the volume model with an input visual hull and regressing its pose from the pose model, pose estimation can be realized.

The additional feature that can be used for the human body part tracking is the color and texture information. The motivation behind this feature is the ability to distinguish between different parts, which is important in the cases of self occlusion.
The appearance of each part can be described using part-based color histograms [121] or a mixture of Gaussian color distributions [177]. Skin color can be used for finding hands and head [146].

2.1.4 Dimensionality Reduction

Despite the high dimensionality of the problem, many poses can be represented in a low dimensional space using dimensionality reduction [161]. This space can be obtained by learning from different motion types [61, 37, 159]. When reducing dimensionality, the basic assumption is that body part movements are mutually dependent and therefore lie in a low-dimensional manifold. For example, leg and arm movements during walking are highly correlated. Tracking in such low dimensional manifolds can be done more effectively, as it requires fewer particles. There exist several possible strategies for reducing the dimensionality of the configuration space. Several works attempted to learn subspace models from an activity. For example, Ormoneit et al. [108] use PCA on the cyclic motions. Another way to cope with high-dimensional data space is to learn low-dimensional latent variable models [171]. Elgammal et al. [43] use locally linear embedding (LLE) to learn activity based manifolds from silhouette data. They then use nonlinear regression methods to learn mappings from manifolds back to silhouette space and to 3D data. Jenkins et al. [70] use ST-Isomap to learn embedding of multi-activity human motion data. However, methods like Isomap [152] and LLE [138] do not provide a mapping between the latent space and the data space. Sminchisescu et al. [147] used spectral embedding techniques to learn an embedding of 3D motion capture data. They also learn a mapping back to pose space separately, which requires a large amount of training data. Tian et al. [153] use Gaussian Process Latent Variable Models (GPLVM) (Lawrence [79]) for 2D pose estimation. Particle filtering is used for the tracking. Li et al. [82] use Locally Linear Coordination (LLC) [151] for learning the mapping. Urtasun et al. [160, 159, 157] use a form of probabilistic dimensionality reduction with a Gaussian Process Dynamical Model (GPDM) (Wang et al. [167]) to formulate the tracking as a nonlinear least-squares optimization problem. Urtasun et al. [158] focused on learning human motion models with interpretable latent directions enabling style/content separation, and generalization beyond the learning data set. Similar to our approach, Andriluka et al. [8] use Hierarchical Gaussian Process Latent Variable Model (HGPLVM) to
model prior on possible articulations and temporal coherency within a *walking* cycle.

While for many actions it is intuitive that a motion can be represented in a low dimensional manifold, this is not the case for a set of different motions. Let us use the *walking* motion as an example. One can notice that for this motion type the locations of the ankles are highly correlated with the location of the other body parts. Therefore, it seems natural to be able to represent the poses from this action in a low dimensional space. However, when several different actions are involved, the possibility of a dimensionality reduction, especially a usage of 2D and 3D spaces, is less intuitive. Mitchelson and Hilton [93] suggested a method for dynamic estimation of pose of multiple people using multiple video cameras. They performed tracking using a model-based approach and a set of cues which exploit both shape and color information. Zhang et al. [179] use an algorithm that makes use of a learned pose-based transition model for detection and tracking of multiple humans. Han and Liang [56] used HGPLVM for hand gesture recognition by means of a Hierarchical CRF (HCRF). Urtasun et al. [158] suggested to perform learning of latent spaces with multiple motions by adding a prior that is based on local neighborhood distances. Gupta et al. [55] extended the GPLVM to include an embedding from observation space (the space of image features) to the latent space, which allowed them to learn and estimate poses from actions which exhibit large systematic variation in joint angle space due to difference in contextual variables. In [62] Hou et al. used EM clustering algorithm in order to segment complex motion, such as ballet dancing, into primitive motions, each of which was modeled by GPLVM. Chen et al. [27] propose a switching GPDM (SGPDM) model, which combines the switching dynamical system and GPDM. The proposed switching variables enable the SGPDM to capture diverse motion dynamics effectively, and also allow to identify the motion class.

We apply a nonlinear dimensionality reduction using GPLVM and GPDM [167] (which are explained in section 3.1) with an annealed particle filter [101, 38, 48]. Both GPLVM and GPDM comprise a mapping from a latent space to the data space and, therefore, are more suitable for tracking than the linear methods such as PCA. In addition, GPDM generates a dynamical model in the latent space, which is used to predict the next pose. Our method generates a mapping function from the low
dimensional latent space to the full data space. The method relies on learning from previously observed poses of different motion types. For tracking purposes we separate the model state into two independent parts: one contains information about 3D location and body orientation and the second one describes the pose. We learn the latent space that describes only poses. The tracking algorithm consists of two stages. First, the particles are generated in the latent space and are transformed into the data space by using a learned a priori mapping function. Next, we add rotation and translation parameters to obtain valid poses. The likelihood function is then calculated in order to evaluate how well a pose matches the visual data. The resulting tracker estimates the locations in the latent space that represent poses with the highest likelihood. We will address the advantages of each of the dimensionality reduction methods later.

2.2 Action Recognition and Behavior Analysis

The extensive research conducted in the last decade sought to understand and analyze the processes of human interactions in general and human motion in particular. Automatic video annotation is crucial for a great variety of applications, such as casino game search and finding tackles in soccer game recordings. It was clear that in order to identify moving body poses and its reciprocal interactions a somewhat scene analysis was inevitable.

There are several reasons the problem is difficult. First, the collection of possible activities appears to be very large, and no straightforward vocabulary is known. Second, may vary significantly if done by different humans or in different times. Third, there is little evidence about what needs to be measured to obtain a good description of activity.

Many approaches have been proposed to classify the human actions. In this section we will present them and provide a brief comparison between them.

The first approach is to determine the body pose model in terms of kinematics and dynamics before analyzing the scene. Then, the scene is segmented to ”influence regions” and further analysis performed [181]. Most of the works that were done in
the area of behavior analysis of humans were done using a single camera. At first, the research concentrated on a single subject analysis, secluded from its environment [137, 11]. Then, it focused on examination and segregation of a finite set of body parts, as hands, legs and head [42, 140]. Appearance based approach is taken when no preliminary model is available, then the motion analysis performed using heuristic assumptions [4, 110, 109, 180]. Several works were done in attempt to combine both approaches [137]. Traditionally, action classification is divided into template matching [126] and state-space approaches. Recently, many different approaches have been proposed. In this review we will try to group them according to the techniques they use.

2.2.1 Appearance-based action classification

Graphical models consist nodes, representing states, connected by edges. These states edges represent probabilities of the transitions between states. For action recognition, an observation can represent the image representation at a given frame. Usually one model is trained per action class. This way the model states correspond to phases of the performed action. Graphical models are either generative or discriminative. While they share many characteristics, they are conceptually different. generative model is a model for randomly generating observable data, typically given some hidden parameters. It specifies a joint probability distribution $P(x, y)$ over observation $x$ and label sequences $y$. Generative models are used in machine learning for either modeling data directly (i.e., modeling observed draws from a probability density function), or as an intermediate step to forming a conditional probability density function. A conditional distribution can be formed from a generative model through the use of Bayes’ rule. In contrast, discriminative models are a class of models used in machine learning for modeling the dependence of an unobserved variable $y$ on an observed variable $x$. Within a statistical framework, this is done by modeling the conditional probability distribution $P(y|x)$, which can be used for predicting $y$ from $x$. Discriminative models differ from generative models in that they do not allow one to generate samples from the joint distribution of $x$ and $y$.

Generative graphical models
**Hidden Markov models:** Hidden Markov models (HMM) are the most well-known generative graphical statistical models in which the system being modeled is assumed to be a Markov process with unobserved state. An HMM can be considered as the simplest dynamic Bayesian network. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output dependent on the state is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by a HMM gives some information about the sequence of states. In the classification algorithms the hidden states usually correspond to different stages of an action and the model state transition probabilities, and observation probabilities. Ahmad and Lee [5] represent Human action as a set of multidimensional combined local-global (CLG) optical flow and shape flow feature vectors in the spatial-temporal action boundary. Actions are modeled by using a set of multidimensional HMMs for multiple views using the combined features. Training of an HMM can be done efficiently using the Baum-Welch algorithm. The Viterbi algorithm is used to determine the probability of observing a given sequence. When using a single HMM per action, action recognition becomes finding the action HMM that could generate the observed sequence with the highest probability.

In approach suggested by Lv and Nevatia [83] each action class is modeled as a chain of the extracted key poses (Figure 2.7.1.a). Each node (state) in this graph model corresponds to one key pose. A back-link connects backward two states in the same model (Figure 2.7.1.b). Back-links are useful to model periodic actions such as walking and actions with unknown number of repeated motions such as waving hands. An inter-link links one action to another (Figure 2.7.1.c). Inter-links are used to rule out unlikely transitions between different action models. Both back-links and inter-links are determined based on the similarity between two 3D key poses. By connecting different action models a more complex graph model called an Action Net is constructed (Figure 2.7.1.d). Action Net provides long-term contextual constraints for action recognition and segmentation in that the links within an action model specify the order of the key poses and the links across action models constrain the possible action transitions. The Action Net is "unrolled" to model changes in the
actors orientation (Figure 2.7.1.e). Given the input, silhouette matching between the input frames and the key poses is performed first using an enhanced Pyramid Match Kernel algorithm. The best matched sequence of actions is then tracked using the Viterbi algorithm.

Feng and Perona [45] represented a pose in space-time, called ’movelet’. A movelet is a collection of shape, motion and occlusion of image patches corresponding to the main parts of the body. The (infinite) set of all possible movelets is quantized into codewords obtained by the vector quantization. For every pair of frames each codeword is assigned a probability. Recognition is performed using HMMs by estimating the most likely sequence of codewords and the action that took place in a sequence. Weinland et al. [176] model actions using three dimensional occupancy grids, built from multiple viewpoints, in an exemplar-based HMM. 3D exemplars are used to produce 2D image information that is compared to the observations. Parameters that describe image projections are added as latent variables in the recognition process. In addition, the temporal Markov dependency applied to view parameters allows them to evolve during recognition as with a smoothly moving camera. Figure 2.7.2 shows the probabilistic dependencies of actions: an action is modeled as a hidden state sequence $Q$, e.g. a motion sequence in a pose space. At each time step $t$, a 3D exemplar $x_t$, i.e. a visual hull, is drawn from the motion sequence $Q$. Observations $y_t$, i.e. silhouettes, result then from a geometric transformation of exemplars that is defined by 2 sets of parameters: $\hat{l}$ are observed parameters, e.g. camera parameters determined in a preliminary step, and $\tilde{l}$ are latent parameters, e.g. body orientation determined during recognition. Shaded nodes in the graph correspond to observed variables. Messing et al. [89] presented a system that tracks a set of keypoints in a video sequence, extracts their velocity histories, and uses a generative mixture model to learn a velocity-history language and classify video sequences. They model activity classes as a weighted mixture of bags of augmented trajectory sequences. Each mixture component models a velocity history feature with a Markov chain, and each activity class has a distribution over those mixture components.
Using body parts: In order to simplify the learning process and in order to detect new actions, which are not part of the training set, one has to model the human body as a collection of body parts and learn the motion patterns (using HMM) for each part separately. In addition the dependencies in movements of different parts have to be investigated. Chakraborty et al. [24] detect the human body parts and then learn the changes of those body parts for action recognition. They incorporate sub-classifiers for the head, arms and the leg detection. Each sub-classifier detects body parts under a specific range of viewpoints. Then, view-invariance is fulfilled by combining the results of these sub classifiers. A HMM is designed for each action, which is trained based on the detected body parts. Peursum et al. [113] break each action down into a two-level hierarchy of phases (sub-actions) and motion within each phase. The hierarchy is tractably modeled with a Hierarchical Hidden Markov Model (HHMM) by factoring the states of the lower level (which model the actual pose). Each action is then modeled by a different instance of this factored-state HHMM (FS-HHMM), and the most likely model for a given sequence provides the action label and posture sequence. Natarajan and Nevatia [98] introduce the Hierarchical Variable Transition Hidden Markov Model (HVT-HMM) which is a three-layered
extension of the Variable Transition Hidden Markov Model (VTHMM). The topmost layer of the HVT-HMM represents the composite actions and contains a single Markov chain, the middle layer represents the primitive actions which are modeled using a VTHMM whose state transition probability varies with time and the bottommost layer represents the body pose transitions using a HMM. Niebles and Li [102] use hierarchical representation of a human body model, that is used for the action categorization. Han et al. [57] suggested using Hierarchical Gaussian Process Latent Variable Model (HGPLVM) to describe and classify the human actions. In their work human actions are inferred from human body joint motions and human bodies are decomposed into several physiological body parts according to inherent hierarchy (e.g. right arm, left arm and head all belong to upper body).

**Grammars:** Grammars specify explicitly in which order parts of an action can be observed. In the work of Fihl et al. [46] human whereabouts at are extracted by double difference images and represented by four features. In each frame the primitive, if any, that best explains the observed data is identified. This leads to a discrete recognition problem since a video sequence is converted into a string containing a sequence of symbols, each representing a basic movements. After pruning the string a probabilistic Edit Distance classifier is applied to identify which action best describes the pruned string. Turaga et al. [155] describe activities as a cascade of dynamical systems which significantly enhances the expressive power of the model while retaining many of the computational advantages of using dynamical models. They derive methods to incorporate view and rate-invariance into these models so that similar actions are clustered together irrespective of the viewpoint or the rate of execution of the activity. This cascade can be regarded as a grammar that describes the production rules for each action in terms of a sequence of action prototypes. Park et al. [111] presented the string comparison method, which performs a detailed two person identification using a nearest neighbor classifier. They also proposed a recognition method based on model tracking and deterministic finite state automata [109]. Hongeng et al. [61] presented probabilistic finite state automata (FSA) for low level human interactions. Wada et al. [165] used nondeterministic finite state automata that utilize state product space. Aggarwal et al. [109] used Bayesian Networks (BN) for extracting features such as hand and torso position, and then applied sequence matching in order to classify the
Figure 2.8: Semantic interpretation of two-person interaction for hugging sequence (reprinted from [111]).

scene. Ogale et al. [105] extract actions and their constituent atomic poses from a set of multiview multiperson video sequences by an automatic keyframe selection process, and are used to automatically construct a probabilistic context-free grammar (PCFG). Given a new single viewpoint video, they parse it to recognize actions and changes in viewpoint simultaneously. Kojima et al. [76] have introduced an approach where natural language is used to describe single person activities.

**Discriminative graphical models**

HMMs, shown in Fig. 2.9.a, and stochastic grammars are generative models, assigning a joint probability to paired observation and label sequences; the parameters are trained to maximize the joint likelihood of training examples. HMMs make strict assumption that observations are conditionally independent given class labels, and cannot represent multiple interacting features and long-range dependencies of the observations. Thus, it largely limits the applicability of the HMM models. Therefore, discriminative graphical models, such as [149, 172, 170, 41], have been proposed, that learn a conditional distribution of action labels, given the observations.

Conditional random fields (CRFs) (shown in Fig. 2.9.b) are a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. Much like a Markov random field, a CRF is an undirected graphical model in which each vertex represents a random variable whose distribution is to be inferred, and each edge represents a dependency between two random variables. Sminchisescu
et al. [149] showed that CRFs solve the classical version of the label bias problem, and, more significantly, that CRFs perform better than HMMs and maximum entropy Markov models (MEMM) when the true data distribution has higher-order dependencies than the model, as is often the case in practice. Wang and Suter [170] presented framework that combines kernel principal component analysis (KPCA) based feature extraction and factorial conditional random field (FCRF) (shown in Fig. 2.9.c) based motion modeling. Silhouette data is represented by nonlinear dimensionality reduction that explores the underlying structure of the articulated action space and preserves explicit temporal orders in projection trajectories of motions. FCRF models temporal sequences in multiple interacting ways, thus increasing joint accuracy by information sharing. They demonstrated the superiority of FCRF to both HMM and general CRF. Wang et al. [172] introduced hidden state conditional random field (HCRF) (shown in Fig. 2.9.d) model for gesture recognition, which has the ability of CRFs to use long range dependencies, and the ability of HMMs to model latent structure. By regarding the sequence label as a random variable they trained a single joint model for all the gestures and share hidden states between them and showed that HCRFs outperform both CRFs and HMMs for certain gesture recognition tasks. Morency et al. [95] presented Latent-Dynamic Conditional Random Field (LDCRF) (shown in Fig. 2.9.e) model which is a discriminative approach for gesture recognition. LDCRF model discovers latent structure that best differentiates visual gestures and can distinguish subtle motion patterns. The LDCRF model thus combines the strengths of CRFs and HCRFs by capturing both extrinsic dynamics and intrinsic sub-structure. It learns the extrinsic dynamics by modeling a continuous stream of class labels, and it learns internal sub-structure by utilizing intermediate hidden states. Ning et al. [103] proposed the latent pose conditional random fields (LPCRF) (Fig. 2.9.f) to bridge the gap between the high dimensional observations and the random fields. In LPCRF model, the observation layer of the random fields is replaced with a latent pose estimator that learns to compress the high dimensional visual features into a compact representation (like human pose). The structure of the model also enables transfer learning to utilize the existing knowledge and data on image-to-pose relationship. Natarajan and Nevatia [99] represent poses in a Conditional Random Field (CRF) whose observation potentials are computed using shape similarity and the transition potentials are computed using optical flow. They present
Shape, Flow, Duration-Conditional Random Field (SFD-CRF) (shown in Fig. 2.9.g), which is capable of representing spatial and temporal constraints. Then the authors find the best sequence of actions using Viterbi search in the SFD-CRF. Wang and Mori [174] proposed an algorithm based on the cutting-plane method and the decomposed dual optimization. They represent a frame in a video by a global motion feature extracted from the whole frame and a set of salient local patches. The model consists of a root filter and a constellation of several hidden parts. The root filter models the compatibility of the action label and the global motion feature of the whole frame. A hidden part assigns a latent part label to a local patch. The constraints among the patches are captured by pairwise potential functions defined in the model.

2.2.2 Model-based action classification

Discriminative classifiers

Discriminative classifiers distinguish between classes without explicitly modeling each. The image representation is simply regarded as a feature vector. Many methods are those based on Support Vector Machines (SVM). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. Intuitively, an SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. Laptev et al. [77] formulate the problem of motion recognition as a matching of corresponding events in image sequences. To enable the matching, they present a set of motion descriptors. Oikonomopoulos et al. [106] introduced a sparse representation of image sequences as a collection of spatiotemporal events that are localized at points that are salient both in space and time. The spatiotemporal salient points are detected by measuring the variations in the information content of pixel neighborhoods not only in space but also in time. An appropriate distance metric between two collections of spatiotemporal salient points is introduced. This metric is used by Relevance Vector Machine (RVM) for the classifications human-action recognition. Hu et al. [64] treat the candidate regions of an action as a bag of instances. They proposed
Figure 2.9: Graphical representation of (a) HMM model (b) linear-chain CRF (if the dotted lines exit, it will represent a model with a context of 3 observation timesteps); (c) two-chain factorial CRF (FCRF) including links between cotemporal labels, explicitly modeling limited probabilistic dependencies between two different label sequences; (d) HCRF (multi-class) model; (e) LPCRF model; (f) LDCRF model; (g) unrolled Graphical Model of the SFD-CRF for Pose Tracking and Recognition.
Simulated annealing Multiple Instance LEarning Support Vector Machines (SMILE-SVM), which aims to obtain a global optimum via simulated annealing method, for learning human action detector based on imprecise action locations. Gilbert et al. [52] use compound 2D corner features. The simple corners detected form an overcomplete set of features are encoded with respect to their spatial and temporal neighborhood. Compound corner groups are formed using a multi-stage grouping. These compound features are used to produce localized likelihoods of actions. The whole sequence is then classified as the action with the maximized likelihood.

Nearest neighbor classification

One of the simplest and yet commonly used technique for the action classification is k-Nearest Neighbor (KNN). The basic idea to have some representation of the labelled sequences (usually called models) from the training set. The given sequence is compared to these models and the most common label among the k most similar sequences is chosen as the classification. The ability to cope with variations in spatial and temporal performance, viewpoint and image appearance depends on the representation and the distance metric that is applied. NN classification can be either performed at the frame level, or for whole sequences. In the latter case, issues with different frame lengths need to be resolved.

The early approaches for human body representation were a moving bounding box or a silhouette and the research focused on detecting body parts defined preliminary to the analysis process. It was soon understood that such simple representation of the human body is not enough for small details recognition such as eyes or fingers. Also, the silhouette representation suffers from lack of precession needed in order to handle the occlusion or joint body parts. For example, Blank et al. [18] apply 1-NN using Euclidean distance between global features, Batra et al. [15] use Euclidean distance between histograms. Later attempts to analyze human motion, when a model of an articulated structure is preliminary defined, used Coupled Hidden Markov models (CHMM) [107]. Tran et al. [154] use a learned discriminative distance metric in their NN classification. Reddy et al. [132] presented a framework involving the feature-tree to index large scale motion features using Sphere/Rectangle-tree (SR-tree). The recognition consists of two steps: a) recognizing the local features by non-parametric
nearest neighbor (NN), b) using a simple voting strategy to label the action.

When individual sequences are used for comparison, there is the risk that outliers will have a large impact on the final classification. Also, the computational performance of the nearest neighbor classifier is linear in the number of training sequences. This might cause problems when there are many training sequences available. Instead, action class prototypes can be used with 1-NN classification. To solve this problem Wang and Suter [168] generate models by averaging over sequences with similar class labels. Similarly, Poppe and Poel [115] learn discriminative feature transforms and distinguish between pairs of classes. Weinland et al. [31] introduces Motion History Volumes (MHV) as a free-viewpoint representation for human actions in the case of multiple calibrated, and background-subtracted, video cameras. They presented algorithms for computing, aligning and comparing MHVs of different actions performed by different people in a variety of viewpoints.

One drawback of action class prototypes is that they are not able to model more complex class distributions. Rodriguez et al. [134] suggested approach based on a Maximum Average Correlation Height (MACH) filter. MACH is capable of capturing intra-class variability by synthesizing a single Action MACH filter for a given action class. They generalized the traditional MACH filter to video (3D spatiotemporal volume), and vector valued data. Chaudhry et al. [26] proposed the Histogram of Oriented Optical Flow (HOOF) features to represent human activities. They modeled the proposed HOOF features as outputs of Non-Linear Dynamical Systems (NLDS) and use the Binet-Cauchy kernels for NLDS to perform human activity recognition.

Dynamic Time Warping (DTW) is an algorithm for measuring similarity between two sequences which may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. In general, DTW is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain
Figure 2.10: Using motion history for action classification. The two actions are recorded by multiple cameras, spatially integrated into their visual hulls (a), and temporally integrated into motion history volumes (b)(c). Invariant motion descriptors in Fourier space (d) are used for comparing the two actions. (Weinland et al. [31])
non-linear variations in the time dimension. This sequence alignment method is often used in the context of hidden Markov models. Veeraraghavan et al. [163] used Kendall's definition of shape for feature extraction. Since the shape feature rests on a non-Euclidean (spherical) manifold, they proposed a nonparametric model, based on Dynamic Time-Warping, on the tangent space and demonstrate the ability of these models to capture the nature of shape deformations using experiments on gait-based human recognition. In later work [162], they also address the alignment of sequences by considering the space of warping functions for a given activity.

paragraphKeyframes: Visual recognition of human actions in video clips has been an active field of research in recent years. However, most published methods either analyze an entire video and assign it a single action label, or use relatively large look-ahead to classify each frame. However, human vision proves that simple actions can be recognized almost instantaneously: humans can correctly recognize actions from very short sequences (often even from single frames), and without temporal look-ahead. The main message is that very short snippets (1-7 frames), are sufficient for basic action recognition, with rapidly diminishing returns, as more frames are added.

For example, Sullivan and Carlsson [150] recognize forehand and backhand tennis strokes by matching edge representations to stored key poses. Similarly Wang et al. [173], attempt to discover the set of action classes present in a large collection of training images. These action classes are used to label test images. The distance between a pair of images is computed using a linear programming relaxation technique. Spectral clustering is then performed using the resulting distances. Weinland et al. [32] performed segmentation and clustering of action classes based on a motion descriptor which can be extracted efficiently from reconstructed volume sequences. This is convenient when the frame contains a key pose, but in general will generate many false matches. Schindler and Van Gool [139] presented a system for action recognition from very short sequences of 1-10 frames. They showed that even local shape and optic flow for a single frame are enough to achieve correct recognitions, and snippets of 5-7 frames (0.3-0.5 seconds of video) are enough to achieve a performance similar to the one obtainable with the entire video sequence. Dedeoglu et al. [34] use histograms of matches to manually selected key poses. The histogram is normalized and 1-NN is used for classification. Weinland and Boyer [175] represent
Figure 2.11: Examples of actions from databases WEIZMANN (top) and KTH (bottom). Note that even a single frame is often sufficient to recognize what a person is doing. (reprint from [32])

motion sequences with respect to a set of discriminative static key-pose exemplars and without modeling any temporal ordering (shown on Figure 2.12). The proposed representation is equivalent to embedding actions into a space defined by distances to key-pose exemplars.

Laptev and Pérez [78] introduced "keyframe priming" that combines discriminative models of human motion and shape within an action. Zhao and Elgammal [182] presented an approach for human action recognition by finding the discriminative key frames from a video sequence and representing them with the distribution of local motion features and their spatiotemporal arrangements. In this approach, the key frames of the video sequence are selected by their discriminative power and represented by the local motion features detected in them and integrated from their temporal neighbors. In the key frames representation, the spatial arrangements of the motion features are captured in a hierarchical spatial pyramid structure.
Figure 2.12: Overview of the embedding method: Two action sequences $Y$ (walk) and $Y^*$ (jump forward on one leg) are matched against a set of silhouette exemplars $x_i$. For each exemplar the best matching frame in the sequence is identified (exemplar displayed on top of the corresponding frame; light colors correspond to high matching distances; dark colors to low matching distances). The resulting matching distances $d_i^*$ form vector $D^*$, which is interpreted as an embedding of the sequences into a low dimensional space $\mathbb{R}_n$. The final classifier is learned over $\mathbb{R}_n$, where each point represents a complete sequence. (Reprinted from [175])
**Dimensionality reduction:**

As mentioned above, despite the high dimensionality of the feature space, many poses lie in a low dimensional manifold and often can be presented in a low subspace using dimensionality reduction. This can be used for human pose recovery. Similar idea can be applied for the action classification. Given a new sequence one can calculate the distance of each frame to the manifold representing a certain action. In the past ten years, a number of successful approaches based on nonlinear manifold learning have been proposed. Masoud and Papanikolopoulos [87], for example, use PCA on feature images to determine the manifold. Chin et al. [28] learn manifolds using Local Linear Embedding (LLE) on silhouette images. They performed comparison between different methods for the extrapolation of learned manifolds within the context of activity recognition. Wang and Suter [169] learn explicit representations for dynamic shape manifolds of moving humans. They exploit locality preserving projections (LPP) for dimensionality reduction, leading to a low-dimensional embedding of human movements. A sequence of moving silhouettes associated to an action video, by LPP, is projected into a low-dimensional space to characterize the spatiotemporal property of the action, as well as to preserve much of the geometric structure. To match the embedded action trajectories, the median Hausdorff distance or normalized spatiotemporal correlation is used for similarity measures. Action classification is then achieved in a NN framework. Blackburn and Ribeiro [17] used Isomap for dimensionality reduction in human motion recognition and show how an adapted dynamic time warping algorithm (DTW) can be successfully used for matching motion patterns of embedded manifolds.

Jia and Yeung [71] proposed Local Spatio-Temporal Discriminant Embedding (LSTDE). The proposed method projects data points (silhouette-based image frames of human action sequences) in a local neighborhood into the embedding space where data points of the same action class are close while those of different classes are far apart. LSTDE finds an optimal embedding which maximizes the principal angles between those temporal subspaces associated with data points of different classes. Orrite-Uruuela et al. [104] create Self Organizing feature Map (SOM) for every action, grouping viewpoint (spatial) and movement (temporal) in a principal manifold.
Figure 2.13: Silhouette contour of the projection from manifold space to image space. With the exception of a few degraded cases each motion sequence is recognizable despite only the first and last silhouettes of each sequence falling exactly on a projected data point. (Reprinted from [17])
Action recognition is carried out by comparing a 2D motion template, built from observations, with learned models of the same type captured from a wide range of viewpoints. Every new 2D motion template gives a distance to the map, related to the probability that motion feature belongs to that particular action. Action recognition is accomplished by a Maximum Likelihood (ML) classifier over all specific-action SOMs.

We also presented several algorithms for human motion classification using a single latent space [126, 127, 128], generated by GPLVM or GPDM, or a hierarchy of low dimensional latent spaces [130, 131, 129], generated by HGPLVN. The algorithms uses the trajectories generated in the latent space by our tracker algorithms. Fréchet distance is used to calculate the distance between between the models, representing different actions, and these trajectories. The Nearest Neighbor technique to determine the correct action. We demonstrate the algorithm’s robustness in classifying different actions performed by different subjects. We show that the method can classify interactions between people, which is a much harder task due to the high rate of occlusion. We introduce body parts weights, that allow the classification algorithm
to detect irregular actions, that combine several motion types, and classify them correctly.
Chapter 3

Gaussian Process Annealed Particle filter

In this chapter we will present the Gaussian Process Annealed Particle filter (GPAPF). We apply a nonlinear dimensionality reduction using GPLVM and GPDM [167] (which are explained in section 3.1) with an annealed particle filter [38]. Both GPLVM and GPDM comprise a mapping from a latent space to the data space. Therefore these methods are more suitable for tracking than the methods such as PCA, where the mapping function is from the data space to the latent space. In addition, GPDM generates a dynamical model in the latent space, which is used to predict the next pose. Our method generates a mapping function from the low dimensional latent space to the full data space. The method relies on learning from previously observed poses of different motion types. For tracking purposes we separate the model state into two independent parts: one contains information about 3D location and body orientation and the second one describes the pose. We learn the low dimensional latent space that describes only poses. The tracking algorithm consists of two stages. First, the particles are generated in the latent space and are transformed into the data space by using a learned a priori mapping function. Next, we add rotation and translation parameters to obtain valid poses. The likelihood function is then calculated in order to evaluate how well a pose matches the visual data. The resulting tracker estimates the locations in the latent space that represent poses with the highest likelihood. In this chapter we will also address the advantages of each of the dimensionality reduction methods.
The chapter is organized as follows. Section 3.1 provides a short description of the particle filter and the annealed particle filter algorithm, while Section 3.2 introduces the concept of Gaussian Process models. This standard material has been provided for completeness. Sections 3.3 presents our algorithms for tracking. Finally, Section 3.5 contains experimental results of GPAPF tracking, as well as a comparison of our GPAPF tracker with the annealed particle filter and particle filter trackers.

3.1 Filtering

3.1.1 Particle Filter

The particle filter was designed for tracking objects, using the Bayesian inference framework. This algorithm represents the posterior and prior distribution numerically using sets of weighted particles, and uses importance sampling in order to ensure the representation remains sufficiently accurate. Importance sampling is a general technique for estimating the statistics of a random variable. The estimation is based on samples of this random variable generated from an easy-to-sample distribution, called the proposal distribution.

Let us denote $x_n$ as a hidden state vector and $y_n$ as a measurement in time $n$. The algorithm builds an approximation of a maximum posterior estimate of the filtering distribution: $p \left(x_n | y_{1:n} \right)$, where $y_{1:n} ≡ (y_1, ..., y_n)$ is the history of the observation. This distribution is represented by a set of pairs $\{x^{(i)}_n; \pi^{(i)}_n\}_{i=1}^{N_p}$, where $\pi^{(i)}_n \propto p \left(y_n | x^{(i)}_n \right)$. Using Bayes’ rule, the filtering distribution can be calculated using two steps [9]:

Prediction step:

$$p \left(x_n | y_{1:n-1} \right) = \int p \left(x_n | x_{n-1} \right) p \left(x_{n-1} | y_{1:n-1} \right) dx_{n-1}$$  \hfill (3.1.1)

Filtering step:

$$p \left(x_n | y_{1:n} \right) \propto p \left(y_n | x_n \right) p \left(x_n | y_{1:n-1} \right)$$  \hfill (3.1.2)

Therefore, starting with a weighted set of samples $\{x^{(i)}_0; \pi^{(i)}_0\}_{i=1}^{N_p}$, the new sample set
\( \{ x_n^{(i)}; \pi_n^{(i)} \}_{i=1}^{N_p} \) is generated according to the distribution, which depends on the previous set \( \{ x_{n-1}^{(i)}; \pi_{n-1}^{(i)} \}_{i=1}^{N_p} \) and the new measurements \( y_n \): \( x_n^{(i)} \sim q \left( x_n^{(i)} | x_{n-1}^{(i)}, y_n \right), i = 1, ..., N_p \). The new weights are calculated using the following formula [9]:

\[
\pi_n^{(i)} = k \pi_{n-1}^{(i)} \frac{p \left( y_n | x_n^{(i)} \right) p \left( x_n^{(i)} | x_{n-1}^{(i)} \right)}{q \left( x_n^{(i)} | x_{n-1}^{(i)}, y_n \right)} \tag{3.1.3}
\]

where

\[
k = \left( \sum_{i=1}^{N_p} \pi_{n-1}^{(i)} \frac{p \left( y_n | x_n^{(i)} \right) p \left( x_n^{(i)} | x_{n-1}^{(i)} \right)}{q \left( x_n^{(i)} | x_{n-1}^{(i)}, y_n \right)} \right)^{-1} \tag{3.1.4}
\]

and \( q \left( x_n^{(i)} | x_{n-1}^{(i)}, y_n \right) \) is the proposal distribution. The disadvantage of particle filtering is the need to approximate \( p \left( x_n | y_{1:n} \right) \), which for high dimensional spaces is a very computationally inefficient and hard task.

Often a weighting function \( w_n^i \left( y_n, x \right) \) can be constructed according to the likelihood function, as in the CONDENSATION algorithm of Isard and Blake [68], which provides a good approximation of the \( p \left( y_n | x_n \right) \), but is also relatively easy to calculate. Therefore, the new problem is to find a configuration \( x_k \) that maximizes the weighting function \( w_n^i \left( y_n, x \right) \). Particle filtering produces robust tracking in low-dimensional configuration spaces (up to about 10 DOF) in the presence of significant clutter. CONDENSATION was successfully used for short human motion capture sequences (see [141, 37]).

However, in the high-dimensional configuration spaces in the human body part motion capture domain, CONDENSATION is unable to use a manageable size particle set to populate the space and represent the density function, while maintaining its ability to detect the maxima. Furthermore, the complicated nature of the observation process when capturing human motion causes the posterior density \( p \left( y_n | x_n \right) \) to be non-Gaussian and multi-modal, as was shown experimentally by Deutscher and Reid [38]. It was also shown by MacCormick [84] that \( N \geq \frac{D}{\alpha^2} \), where \( N \) is the number of particles required, \( d \) is the dimensionality of the data space, \( D \) and \( \alpha \) are constant, and \( (\alpha \ll 1) \). This suggests that a sufficient particle set for human
motion capturing problems should contain thousands of particles, making CONDENSATION intractable for multidimensional problems such as articulated body tracking.

### 3.1.2 Annealed Particle Filter

Deutscher and Reid [38] proposed APF, which optimizes a multi-modal weighting function using an approach similar to that of simulated annealing, suggested by Kirkpatrick et al. [75].

**Algorithm 1**: The annealed particle filter algorithm [38]

**Initialization:** \( \left\{ x_{n,M}^{(i)} : \frac{1}{N} \right\}_{i=1}^{N_p} \)

**for each**: frame \( n \)

**for** \( m = M \) **down to 0** **do**

1. Calculate the weights:
   \[
   \pi_n^{(i)} = k \frac{w_m(y_n, x_{n,m}^{(i)}) p(x_{n,m}^{(i)}|x_{n,m-1}^{(i)})}{q(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}, y_n)},
   \]
   where
   \[
   k = \left( \sum_{i=1}^{N_p} \frac{w_m(y_n, x_{n,m}^{(i)}) p(x_{n,m}^{(i)}|x_{n,m-1}^{(i)})}{q(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}, y_n)} \right)^{-1}.
   \]

2. Draw \( N \) particles from the weighted set \( \left\{ x_{n,m}^{(i)} : \pi_n^{(i)} \right\}_{i=1}^{N_p} \) with replacement and with distribution \( p(x = x_{n,m}^{(i)}) = \pi_n^{(i)} \).

3. Calculate \( x_{n,m-1}^{(i)} \sim q(x_{n,m-1}^{(i)}|x_{n,m}^{(i)}, y_n) = x_{n,m} + n_m \), where \( n_m \) is a Gaussian noise \( n_m \sim \mathcal{N}(0, P_m) \).

**end for**

- The optimal configuration can be calculated using the following formula: \( x_n = \sum_{i=1}^{N_p} \pi_n^{(i)} x_{n,0}^{(i)} \).
- The unweighted particle set for the next observation is produced using \( x_{n+1,M}^{(i)} = x_{n,0}^{(i)} + n_0 \), where \( n_0 \) is a Gaussian noise \( n_m \sim \mathcal{N}(0, P_0) \).

**end for each**

The main idea of [38] is to use an annealing schedule to try to concentrate more particles around the global maximum. To this end, they proposed a set of weighting functions \( \left\{ w_m (y_n, x) \right\}_{m=0}^{M} \), where \( w_{m-1} (y_n, x) \) differs only slightly from \( w_m (y_n, x) \) and
Figure 3.1: Annealed particle filter illustration for $M=5$. Initially the set contains many particles that represent very different poses and therefore can fall into local maxima. In the last layer, all the particles are close to the global maximum and therefore represent the correct pose.

represents a smoothed version of it. The original weighting function $w(y_n, x)$ might be peaky, and thus a large number of particles are required to find the global maxima. Therefore, $w_M(y_n, x)$ is designed to be a very smoothed version of $w_0(y_n, x)$. The smoothed weighting function is usually constructed using $w_m(y_n, x) = (w(y_n, x))^{\beta_m}$, where $1 = \beta_0 > ... > \beta_M$ and $w_0(y_n, x)$ is equal to the original weighting function.

Each iteration of the annealed particle filter algorithm consists of $M$ steps, in each of which the appropriate weighting function is used and a set of pairs $\{x^{(i)}_{n,m}, \pi^{(i)}_{n,m}\}_{i=1}^{N_p}$ is constructed. Tracking is described in Algorithm 1. Fig. 3.1 illustrates the 5-layered annealed particle filter. Initially the set contains many particles that represent very different poses and therefore can fall into local maxima. In the last layer, all the particles are close to the global maximum and therefore represent the correct pose.

One of the disadvantages of annealing is the loss of the multi-modality of the standard particle filter. The other problem with APF is that it still requires a minimum number of particles for tracking, no matter how many layers are used. In practice
this means that for high dimensional state spaces APF still requires many particles. In their work, Deutscher and Reid [38] used 4000 particles for each frame. We will show that GPAPF requires only 500 particles and produces better results than the original APF tracker.

3.2 Gaussian Process Models

3.2.1 Gaussian Process

Gaussian Processes is an approach to learn a mapping $y_i = g(x_i, B)$ from a training set $\{x_i, y_i\}$, where $y_i$ is an input training point in a $D$-dimensional data space, $x \in \mathbb{R}^d$ denotes a latent coordinate in a $d$-dimensional space, which corresponds to the training point $y_i$ and $g()$ a mapping function from the latent space to the data (observation) space. The mapping function depends on $B$, which stand for the parameters for the reconstruction mapping which are obtained during the learning process. Gaussian Processes arise from a Bayesian formulation of the problem, in which one marginalizes over a family of functions for $g$. For mapping function $g$, which is expressed as a linear combination of non-linear functions $\{\chi_j\}$

$$y_i = \sum_j b_j \chi_j(x) + n_y$$

with IID Gaussian Noise $n_y$ and a Gaussian prior over $\{b_j\}$ the marginalization procedure is a Gaussian process model [85]. Fig. 3.2.a gives the graphical presentation of Gaussian process model.

3.2.2 Gaussian Process Latent Variable Model

As mentioned above the Gaussian Processes is a regression, which obtains a mapping $y_i = g(x_i, B)$ from a training set $\{x_i, y_i\}$. However, in general case the latent coordinates $\{x_i\}$ are not given. The Gaussian Process Latent Variable Model (GPLVM)[79] is used to learn the unknown latent positions $\{x_i\}$ and the mapping function $y_i = g(x_i, B)$, given input training points $y_i$. This resembles probabilistic PCA (PPCA). However, while PPCA marginalizes over latent variables to find
the mapping, GPLVM marginalizes over mapping functions and optimizes the latent variables. The other important difference between the PPCA and GPLVM is that GPLVM is not restricted to the linear functions.

Let $Y \equiv [y_1, y_2, ..., y_N]^T$ be a matrix, where row $i$ is equal to the $i$-th training data vector. The assumption is that $y_i$ are zero means. Marginalizing over $B$ we get the following expression:

$$p(Y|M) = \int_B p(Y, B|M) dB = \int_B p(Y|B, M)p(B|M) dB$$  \tag{3.2.2}$$

Under the Gaussian Process model, the conditional density for the data is multivariate Gaussian:

$$p(Y|M) = \frac{|W|^N}{\sqrt{(2\pi)^D |K_Y|^D}} \exp \left( -\frac{1}{2} tr (K_Y^{-1}YW^2Y^T) \right)$$  \tag{3.2.3}$$

where $M \equiv \{ \{ x_i \}, \{ \beta_i \}, \{ w_j \}_{j=1}^D \}$ is a vector of the unknown GPLVM model parameters. The matrix $K_Y$ is called the kernel matrix [79]. The elements of the kernel matrix are given by a kernel function $(K_Y)_{i,j} = k_Y(x_i, x_j)$, which depends on the form of the basis function in 3.2.1. As suggested in [54, 79] RBF kernel function is used, with the parameters $\beta_i$ given by

$$k_Y(x_i, x_j) = \beta_1 \exp \left( -\frac{\beta_2}{2} \|x_i - x_j\|^2 \right) + \delta_{x_i, x_j} \beta_3.$$  \tag{3.2.4}$$

Here $\delta_{x_i, x_j}$ is the Kronecker delta function, $\beta_1$ and $\beta_2$ represent the over all scale and width of the RBF kernel and $\beta_3$ is the variance of the additive noise as it appears in [79]. $W \equiv \text{diag}(w_1, w_2, ..., w_D)$ is a diagonal matrix containing a scale factor for each dimension of the data variable space as described by Grochow et al. [54]. Finally the data likelihood in 3.2.3 and prior distributions for the latent positions and the kernel hyperparameters are combined. The resulting posterior density over the model $M$:

$$p(M|Y) \propto p(Y|M)p(M) = p(Y|M)p(X)p(\beta)p(W).$$  \tag{3.2.5}$$

As suggested in [79] we use an IID zero-mean Gaussian prior with unit covariance over the latent positions, an inverse prior over the kernel hyperparameters and a uniform prior over the weights

$$p(X) = \prod_i N(x_i|0, I); p(\beta) = p(\beta_{1:3}) \propto \prod_{i=1}^3 \frac{1}{\beta_i}. $$  \tag{3.2.6}$$
**Learning:** The objective of the training is to calculate the model parameters are obtained by maximization of the posterior \( p(M|Y) \). In this stage the mapping from the latent space to the full data space is obtained. In addition the latent positions that correspond to the training data vectors are obtained. The maximization of the posterior, given the priors in 3.2.6 is equivalent to minimization of the negative log posterior \( -\ln p(M|Y) \), which is given by:

\[
L = \frac{D}{2} \ln |K_Y| + \frac{1}{2} tr \left( K_Y^{-1}YW^2Y^T \right) \\
+ \frac{1}{2} \sum_i \|x_i\|^2 + \frac{1}{2} \sum_{i=1}^3 \ln (\beta_i) - N \ln |W| 
\]

(3.2.7)

**Learning**

Learning procedure requires initialization, i.e. an initial choice of the latent position that correspond to the full data vectors that are used for the training. Another disadvantage is that one need to specify the dimensionality of the latent space.

**Pose Prior:** Once the model parameters \( M \) are obtained the join density over the learned latent coordinates and the corresponding full state vector is given by [85]:

\[
p(x,y|M,Y) = p(y|x,M,Y)p(x|M,Y) = \frac{p(y,Y|x,M)}{p(Y|x,M)} p(x|M,Y) \tag{3.2.8}
\]

Here \( X \) denotes latent coordinates set \( X = [x_1, x_2, ..., x_N]^T \) that corresponds to the training set \( Y \). Consequently, ignoring the terms that are constant for the optimization process:

\[
p(x,y|M,Y) \propto \frac{|W|^{N+1}}{\sqrt{(2\pi)^{(N+1)D}|\hat{K}_Y|^D}} \exp \left( \frac{1}{2} tr \left( \hat{K}_Y^{-1} \hat{Y} W^2 \hat{Y}^T \right) \right) \exp \left( -\frac{x^T x}{2} \right) 
\]

(3.2.9)

Here \( \hat{Y} \) denotes vector \( \hat{Y} \equiv [Y^T, y]^T \), which consist of the training data points and the the new data vector \( y \). \( \hat{K}_Y \) is the corresponding kernel matrix, which is given by:

\[
\hat{K}_Y = \begin{pmatrix} K_Y & k_Y(x) \\
            k_Y(x)^T & k_Y(x,x) \end{pmatrix}, \tag{3.2.10}
\]
where \( k_Y(x) = [k_Y(x, x_1), k_Y(x, x_2), ..., k_Y(x, x_N)] \) and \( k_Y(x, x') \) is the kernel function.

According to the [54, 85] it is possible to obtain a more simple expression for the negative log likelihood \(-\ln p(x, y|M, Y)\), which up to an additive constant is equal to:

\[
L(x, y) = \frac{\|W(y - \mu_Y(x))\|^2}{2\sigma_Y^2(x)} + \frac{D}{2} \ln \sigma_Y^2(x) + \frac{1}{2}\|x\|^2, \tag{3.2.11}
\]

where

\[
\mu_Y(x) = \mu + YK^{-1}k_Y(x) \tag{3.2.12}
\]

\[
\sigma_Y^2(x) = k_Y(x, x) - k_Y(x)^T K^{-1}k_Y(x) \tag{3.2.13}
\]

Here \( \mu_Y(x) \) is the mean of the state vector reconstructed from the latent position \( x \) and \( \sigma_Y^2(x) \) denotes the uncertainty of the reconstruction. Minimizing \( L(x, y) \) minimizes the reconstruction error by produces \( y \) close to the \( \mu_Y(x) \) and keeps the latent position close to the training data. The third term of the 3.2.11 is the result of the broad prior over latent position. This term has usually little influence on the latent positions.

### 3.2.3 Gaussian Process Dynamical Model

The Gaussian Process Dynamical Model (GPDM) comprises a mapping from a latent space to the data space, and a dynamical model in the latent space [171]. Given training vectors the latent embedding, the latent dynamics and the mapping function are obtained. These mappings are typically nonlinear. The GPDM is obtained by marginalizing over the parameters of the two mappings, and optimizing the latent coordinates of training data. Considering a latent-variable mapping with first-order Markov dynamics:

\[
x_t = f(x, A) + n_{x,t} \tag{3.2.14}
\]

\[
y_t = g(x, B) + n_{y,t} \tag{3.2.15}
\]

Here, \( f() \) and \( g() \) are nonlinear mappings parameterized by \( A \) and \( B \), respectively, \( x_t \in \mathbb{R}^d \) denotes the \( d \)-dimensional latent coordinates at time \( t \) and \( n_{x,t} \) and \( n_{y,t} \) are zero-mean, white Gaussian noise processes. Fig. 3.2.b depicts the graphical model.
As suggested by Wang et al. [171] $f()$ and $g()$ are linear combinations of basis functions:

$$x_t = \sum_i a_i \varphi_i (x_{t-1}) + n_{x,t}$$  \hspace{1cm} (3.2.16) \\
y_t = \sum_j b_j \chi_j (x) + n_{y,t}$$  \hspace{1cm} (3.2.17)

for basis functions $\varphi_i$ and $\chi_j$ and weights $A = [a_1, a_2, ...]$ and $B = [b_1, b_2, ...]$. If $\varphi_i$ and $\chi_j$ are chosen to be linear functions, then the equations 3.2.16 and 3.2.16 represent an AR model. However, when there is no such constrain and nonlinear basis functions are used, the model is significantly richer.

In traditional regression one should fix the number of the basis function and then fit the model parameters. However, from a Bayesian perspective, the specific forms and the numbers of basis functions are incidental, and should therefore be marginalized over. With isotropic Gaussian prior over $b_j$, one can marginalize over $B$ to yield a multivariate Gaussian data likelihood of the form [76, 86]:

$$p(Y|X, \beta) = \frac{|W|^N}{\sqrt{(2\pi)^{ND}|K_Y|^D}} \exp \left( -\frac{1}{2} tr \left( K_Y^{-1} Y W^2 Y^T \right) \right),$$  \hspace{1cm} (3.2.18)
where \( Y = [y_1, y_2, ..., y_N]^T \) is the training data vectors, \( X = [x_1, x_2, ..., x_N]^T \) are the corresponding latent positions and \( K_Y \) is the kernel matrix. The elements of the kernel matrix are defined by a kernel function \((K_Y)_{i,j} = k_Y(x_i, x_j)\), where the RBF kernel is used:

\[
k_Y(x_i, x_j) = \beta_1 \exp \left( -\frac{\beta_2}{2} \|x_i - x_j\|^2 \right) + \frac{\delta_{x_i, x_j}}{\beta_3}.
\]

(3.2.19)

The scaling matrix \( W \equiv [w_1, w_2, ..., w_D] \) is used to account for differing variances in the different data dimensions. The vector \( \overline{\beta} \equiv [\beta_1, \beta_2, ..., W] \) comprises the kernel hyperparameters. Hyperparameter \( \beta_1 \) represents the overall scale of the output function, while \( \beta_2 \) corresponds to the inverse width of the RBFs. The variance of the process noise term \( n_y,t \) is given by \( \beta_3^{-1} \).

In a similar way the joint probability density over the latent coordinates and the dynamics weights \( A \) are formed and then the weights \( A \) are marginalized out:

\[
p(X|\overline{\alpha}) = \int_A p(X|A\overline{\alpha}) p(A|\overline{\alpha}) \, dA
\]

(3.2.20)

where \( \overline{\alpha} \) is a vector of kernel hyperparameters. Incorporating the Markov property gives:

\[
p(X|\overline{\alpha}) = p(x_1) \prod_{t=2}^N p(x_t|x_{t-1}, A, \overline{\alpha}) p(A|\overline{\alpha}) \, dA
\]

(3.2.21)

Assuming an isotropic Gaussian prior over the columns of \( a_i \), it can be shown [166] that:

\[
p(X|\overline{\alpha}) = \frac{p(x_1)}{\sqrt{(2\pi)^{(N-1)d}|K_X|^d}} \exp \left( -\frac{1}{2} \text{tr} \left( K_X^{-1} X_{out} X_{out}^T \right) \right)
\]

(3.2.22)

Here \( X_{out} \equiv [x_2, x_3, ..., x_N]^T \), \( K_X \) is the kernel matrix, which is constructed from \( X_m = [x_1, x_2, ..., x_{N-1}]^T \), and \( x_1 \) is assumed to have an isotropic Gaussian prior. The dynamics is modeled using both the RBF kernel, as well as the following "linear + RBF" kernel:

\[
k_X(x_i, x_j) = \alpha_1 \exp \left( -\frac{\alpha_2}{2} \|x_i - x_j\|^2 \right) + \alpha_3 x_i^T x_j + \frac{\delta_{x_i, x_j}}{\alpha_4}.
\]

(3.2.23)
Learning

Learning GPDM means to estimate the kernel hyperparameters and the latent positions that correspond to the training vectors. As Wang et al. [167] suggest a simple prior distributions over the hyperparameters are used:

\[
p(\alpha) \propto \prod_i \alpha_i^{-1} \\
p(\beta) \propto \prod_i \beta_i^{-1}
\]

(3.2.24)

The GPDM posterior becomes

\[
p(X, \alpha, \beta | Y) \propto p(Y | X, \beta) p(X | \alpha) p(\alpha) p(\beta)
\]

(3.2.25)

It can be shown that the hyperparameters and the latent positions can be found by minimizing the negative log posterior, which is up to an additive constant is equal to:

\[
L = \frac{d}{2} \ln |K_X| + \frac{1}{2} \text{tr} \left( K_X^{-1} X_{out} Y_{out}^T \right) - N \ln |W| + \frac{D}{2} \ln |K_Y| \\
+ \frac{1}{2} \text{tr} \left( K_Y^{-1} Y W^2 Y^T \right) + \sum_i \ln (\alpha_i) + \sum_i \ln (\beta_i)
\]

(3.2.26)

The first two terms come from the log dynamics 3.2.22 and the next three terms come from the log reconstruction density 3.2.16.

Multiple sequences

While GPDM is defined for a single input sequence, it can be extended naturally to multiple sequences \([Y_1, Y_M, ..., Y_M]\) with associated latent coordinates \([X_1, X_M, ..., X_M]\) within a shared latent space. For the latent mapping \(g()\) we can conceptually concatenate all sequences within the Gaussian Processes likelihood. A similar concatenation applies for the dynamics \(f()\), ignoring temporal transitions between the sequences, i.e. omitting the first frame of each sequence from \(X_i\), and the final frame of each sequence from the kernel matrix \(K_X\).
3.3 Gaussian Process Annealed Particle Filter in Action

In this section we present the Gaussian Process Annealed Particle Filter (GPAPF) algorithm for body part tracking. In section 3.3.1 we introduce the model of the human body; in section 3.3.2 we specify the weighted function that we used. In section 3.3.4 we explain how we learn the latent space that will later be used for tracking. The tracking algorithm is explained in section 3.4. In section 3.4.1 motion models are introduced. Finally, in section 3.4.2 we explain how the tracker can be improved to achieve more precise results.

3.3.1 The Model

In our work we use a model similar to the one proposed by Deutscher et al. [37], with some differences in the annealing schedule and weighting function. The body model is defined by a pair $M = \{L, \Gamma\}$, where $L$ stands for the limb lengths and $\Gamma$ describes the global location and orientation of the body and the angles between the limbs. The limb parameters are constant and represent the actual size of the tracked person. The angles represent the body pose and, therefore, are dynamic. The state is a vector of dimensionality 31: 3 DoF for the global 3D location, 3 DoF for the global rotation, 4 DoF for each leg, 6 DoF for the torso, 4 DoF for each arm, and 3 DoF for the head (see Fig. 3.3.a). The whole tracking process estimates the angles in such a way that the resulting body pose will match the actual pose. This is achieved by maximizing the weighting function, which is explained next.

3.3.2 The Weighting Function

In order to evaluate how well the tracked body pose matches the actual pose, a weighting function $w(\Gamma, Z)$ has to be defined, where $\Gamma$ is the model’s configuration (i.e. angles) and $Z$ stands for visual content (the captured images). The weighting function that we use is a version of the one suggested by Deutscher et al. [38], with some modifications. Experiments were performed with 3 different features:
Figure 3.3: The 3D body model (a) and the samples drawn for the weighting function (b). In the right-hand image the blue samples are used to evaluate the edge matching, the cyan points are used to calculate the foreground matching, and the rectangles with edges on the red points are used to calculate the part-based body histogram.

edges, foreground silhouette and foreground histogram. Fig. 3.3.b shows the different features projected on the image.

**Edges**

The first feature is the edge map. As Deutscher et al. [38] proposes this feature is the most important one, and provides a good outline for visible parts, such as arms and legs. The other important property of this feature is that it is invariant to the color and lighting condition. The edge maps, in which each pixel is assigned a value dependent on its proximity to an edge, are calculated for each image plane. Each part is projected on the image plane and samples of the $N_e$ hypothesized edges of human body model are drawn. A sum-squared difference function is calculated for these samples:

$$
\Sigma^e(\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{cv}} \sum_{j=1}^{N_e} (1 - p_j^e(\Gamma, Z_i))^2
$$

(3.3.1)

where $N_{cv}$ is a number of camera views, and $Z_i$ stands for the image from the $i$-th camera. The $p_j^e(\Gamma, Z_i)$ are the edge maps. Each part is projected on the image plane...
and samples of the $N_e$ hypothesized edges are drawn.

However, the problem that occurs using this feature is that the occluded body parts will produce no edges. Even the visible parts, such as the arms, may not produce the edges, because of the color similarity between the part and the body. This will cause $p_{ij}^f (\Gamma, Z_i)$ to be close to zero and thus will increase the squared difference function. Therefore, a good pose which represents well the visual context may be omitted. In order to overcome this problem for each combination of image plane and body part we calculate a coefficient, which indicates how well the part can be observed on this image. For each sample point on the model’s edge we estimate the probability being covered by another body part. Let $N_i$ be the number of hypothesized edges that are drawn for the part $i$. The total number of drawn sample points can be calculated using $N_e = \sum_{i=1}^{N_{bp}} N_i$, where $N_{bp}$ is the total number of body parts in the model. The coefficient of part $i$ for the image plane $j$ can be calculated as following:

$$\lambda_{i,j} = \frac{1}{N_i} \sum_{k=1}^{N_i} \left(1 - p_{fg}^f (\Gamma_i, Z_j)\right)^2$$  \hspace{1cm} (3.3.2)

where $\Gamma_i$ is the model configuration for part $i$ and $p_{fg}^f (\Gamma_i, Z_j)$ is the value of the foreground pixel map of the sample $k$. If a body part is occluded by another one, then the value of $p_{fg}^f (\Gamma_i, Z_j)$ will be close to one and therefore the coefficient of this part for the specific camera will be low. We propose using the following function instead of sum-squared difference function as presented in (3.3.1):

$$\Sigma^e (\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{bp}} \sum_{j=1}^{N_{cv}} \lambda_{i,j} \sum (\Gamma_i, Z_j)$$ \hspace{1cm} (3.3.3)

where

$$\sum (\Gamma_{bp}, Z_{cv}) = \sum_{k=1}^{N_i} \left(1 - p_k^b (\Gamma_{bp}, Z_{cv})\right)^2$$ \hspace{1cm} (3.3.4)

Silhouette

The second feature is the silhouette obtained by subtracting the background from the image. The foreground pixel map is calculated for each image plane with background pixels set to 0 and foreground set to 1 and sum squared difference function is
computed:

$$\Sigma^{fg} (\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{cv}} \sum_{j=1}^{N_e} \left(1 - p^{fg}_j (\Gamma, Z_i)\right)^2$$

(3.3.5)

where $p^{fg}_j (\Gamma, Z_i)$ is the value is the foreground pixel map values at the sample points.

**Part based histogram**

The third feature is the foreground histogram. The reference histogram is calculated for each body part. It can be a grey level histogram or three separated histograms for color images, as shown in Fig. 3.4. Then, on each frame a normalized histogram is calculated for a hypothesized body part location and is compared to the referenced one. In order to compare the histograms we have used the squared Bhattacharyya distance [30, 112], which provides a correlation measure between the model and the target candidates:

$$\Sigma^h (\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_{bp}} \sum_{i=1}^{N_{bp}} \sum_{j=1}^{N_{cv}} \left(1 - \rho^{part} (\Gamma_i, Z_j)\right)$$

(3.3.6)

where

$$\rho^{part} (\Gamma_{bp}, Z_{cv}) = \frac{\sum_{i=1}^{N_{bins}} \sqrt{p^{ref}_i (\Gamma_{bp}, Z_{cv}) p^{hyp}_i (\Gamma_{bp}, Z_{cv})}}$$

(3.3.7)

and $p^{ref}_i (\Gamma_{bp}, Z_{cv})$ is the value of bin $i$ of the body part $bp$ on the view $cv$ in the reference histogram, and the $p^{hyp}_i (\Gamma_{bp}, Z_{cv})$ is the value of the corresponding bin on the current frame using the hypothesized body part location.

The main drawback of that feature is that it is sensitive to changes in the lighting conditions. Therefore, the reference histogram has to be updated, using the weighted average from the recent history.

**Accumulative weighting function**

In order to calculate the total weighting function the features are combined together using the following formula:

$$w (\Gamma, Z) = e^{-(\Sigma^e(\Gamma,Z)+\Sigma^{fg}(\Gamma,Z)+\Sigma^h(\Gamma,Z))}$$

(3.3.8)
As was stated above, the target of the tracking process is equal to maximizing the weighting function.

### 3.3.3 Motivation: from APF to GPAPF

The drawback of the particle filter tracker is that a high-dimensional state space causes an exponential increase in the number of particles that have to be generated in order to preserve the same particle density. In our case, the data dimension is 31-D. Balan et al. [12] have shown that the annealed particle filter is capable of tracking body parts with 125 particles using 60 fps video input. However, using a significantly lower frame rate (15 fps) causes the tracker to produce bad results and eventually lose the target.

Another problem of the annealed particle filter tracker is that once a target is lost (i.e., the body pose was wrongly estimated, which can happen for quick and jerky movements), it is highly unlikely that the pose on the following frames will be estimated correctly.

### 3.3.4 Learning

In order to reduce the dimension of the space we introduce Gaussian Process Annealed Particle Filter (GPAPF). We use a set of poses in order to create a low dimensional
latent space. The latent space is generated by applying nonlinear dimension reduction on the previously observed poses of different motion types, such as walking, running, punching and kicking. We divide our state into two independent parts. The first part contains the global 3D body rotation and translation parameters and is independent of the actual pose. The second part contains only information regarding the pose (25 DoF). We use Gaussian Process Dynamical Model (GPDM) in order to reduce the dimensionality of the second part and to construct a latent space, as shown on Fig. 3.5. GPDM is able to capture properties of high dimensional motion data better than linear methods such as PCA. This method generates a mapping function from the low dimensional latent space to the full data space. This space has a significantly lower dimensionality (we have experimented with 2D or 3D). Unlike Urtasun et al. [157], whose latent state variables include translation and rotation information, our latent space includes solely pose information and is therefore rotation and translation invariant. This allows using the sequences of the latent coordinates in order to classify different motion types.

Figure 3.5: The latent space that is learned from different poses during the walking sequence. (a) the 2D space; (b): the 3D space. On the image (a): the brighter pixels correspond to more precise mapping.

We have also made several experiments on constructing a latent space that describes several different motion types. While for many actions it is intuitive that a motion can be represented in a low dimensional manifold, that is not the case for a set
of different motions. Fig. 3.6 shows two different latent spaces that were constructed for two different sets of motion types. As addressed in Section 3.6 that work has to be extended and a deep research has to be performed.

![Figure 3.6: The latent space that is learned from different motion types. (a) 2D latent space from 3 different motions: lifting an object (red), kicking with the left (green) and the right (magenta) legs. (b) 3D latent space from 3 different motions: hand waving (red), lifting an object (magenta), kicking (blue), sitting down (black), and punching (green).](image-url)

We use a 2-stage algorithm. In the first stage a set of new particles is generated in the latent space. Then we apply the learned mapping function that transforms latent coordinates to the data space. As a result, after adding the translation and rotation information, we construct 31 dimensional vectors that describe a valid data state, which includes location and pose information, in the data space. In order to estimate how well the pose matches the images the likelihood function, as described in the previous section, is calculated.

The main difficulty in this approach is that the latent space is not uniformly distributed. Therefore we use the dynamic model, as proposed by Wang et al. [167], in order to achieve smoothed transitions between sequential poses in the latent space. However, there are still some irregularities and discontinuities. Moreover, while in a regular space the change in the angles is independent on the actual angle value, in a latent space this is not the case. Each pose has a certain probability to occur and...
thus the probability to be drawn as a hypothesis should be dependent on it. For each particle we can estimate the variance that can be used for generation of the new ones. In Fig. 3.5.(a) the lighter pixels represent lower variance, which depicts the regions of the latent space that produce more likely poses.

Figure 3.7: Losing and finding the tracked target despite the miss-tracking on the previous frame. Top: camera 1, Bottom: camera 4.

Another advantage of this method is that the tracker is capable of recovering after several frames, from poor estimations. The reason for this is that particles generated in the latent space are representing valid poses more authentically. Furthermore because of its low dimensionality the latent space can be covered with a relatively small number of particles. Therefore, most of possible poses will be tested with emphasis on the pose that is close to the one that was retrieved in the previous frame. So if the pose was estimated correctly the tracker will be able to choose the most suitable one from the tested poses. However, if the pose on the previous frame was miscalculated the tracker will still consider the poses that are quite different. As these poses are expected to get higher value of the weighting function the next layers of the annealing
process will generate many particles using these different poses. As shown in Fig. 3.7 in this way the pose is likely to be estimated correctly, despite the miss-tracking on the previous frame.

In addition the generated poses are, in most cases, natural. The large variance in the data space causes the generation of unnatural poses by the CONDENSATION or by annealed particle filtering algorithms. In the introduced approach the poses that are produced by the latent space that correspond to points with low variance are usually natural as the whole latent space is constructed based on learning from a set of valid poses. The unnatural poses correspond to the points with the large variance (black regions on Fig. 3.5.(a)) and, therefore, it is highly unlikely that it will be generated. Therefore the effective number of the particles is higher, which enables more accurate tracking.

As shown in Fig. 3.5 is that the latent space is not continuous. Two sequential poses may appear not too close in the latent space; therefore there is a minimal number of particles that should be drawn in order to be able to perform the tracking.

The other drawback of this approach is that it requires more calculation than the regular annealed particle filter due to the transformation from the latent space into the data space. However, as it is mentioned above, if same number of particles is used, the amount of the effective poses is significantly higher in the GPAPF then in the original annealed particle filter. Therefore, we can reduce the number of the particles for the GPAPF tracker, and by this compensate for the additional calculations.

### 3.4 Tracking algorithm

As explained above, the state of the tracker consists of 2 statistically independent parts. The first part describes the body’s 3D location: the rotation and the translation (6 DoF). The second part describes the actual pose: the latent coordinates of the corresponding point in the Gaussian Space (that was generated using the methods
introduced in Section 3.2). This second part usually has very few DoFs (we experimented with 2- and 3-dimensional latent spaces). We use a 2-stage algorithm. The first stage is the generation of new particles. In the second stage we apply the learned transform function that transforms latent coordinates to the data space (25 DoF). After the translation and rotation information is added, a 31-dimensional vectors are constructed. These vectors describe a valid data state, which includes location and pose information, in the data space. Then the state is projected onto the cameras in order to estimate how well it fits the images.

Suppose we have $M$ annealing layers. The state is defined as a pair $\Gamma = \{\Lambda, \Omega\}$, where $\Lambda$ is the location information and $\Omega$ is the pose information. We also define $\omega$ as the latent coordinates that correspond to the data vector $\Omega$: $\Omega = \varphi(\omega)$, where $\varphi$ is the mapping function learned by the GPDM. $\Lambda_{n,m}$, $\Omega_{n,m}$ and $\omega_{n,m}$ are the location, pose vector and corresponding latent coordinates in frame $n$ and annealing layer $m$. For each $1 \leq m \leq M - 1$ and for each frame $n$, we generate $\Lambda_{n,m}$ and $\omega_{n,m}$ by adding a multi-dimensional Gaussian random variable to $\Lambda_{n,m+1}$ and $\omega_{n,m+1}$ respectively. Then $\Omega_{n,m}$ is calculated using $\omega_{n,m}$. The full body state $\Gamma_{n,m} = \{\Lambda_{n,m}, \Omega_{n,m}\}$ is projected to the cameras and the likelihood $\pi_{n,m}$ is calculated using the likelihood function, as explained in Section 3.3.2 (see Algorithm 2).

In the original annealed particle filter algorithm, the optimal configuration is achieved by calculating the weighted average of the particles in the last layer. However, as the latent space is not Euclidian, applying this method on $\omega$ will produce poor results. The other method is to choose the particle with the highest likelihood as the optimal configuration $\omega_n = \omega_{n,0}^{(i_{\text{max}})}$, where $i_{\text{max}} = \arg \min_i \left(\pi_{n,0}^{(i)}(\omega_{n,0}^{(i)})\right)$. However, this method of calculating the optimal pose is unstable: in order to ensure the existence of a particle that represents the correct pose, we have to use a large number of particles. Therefore, we propose to calculate the optimal configuration in the data space and then project it back to the latent space. First, we apply the $\varphi$ on all the particles to generate vectors in the data space. Then we calculate the average on these vectors in the data space and project it back to the latent space. This can be written as follows: $\omega_n = \varphi^{-1} \left( \sum_{i=1}^{N} \pi_{n,0}^{(i)}(\omega_{n,0}^{(i)}) \right)$.

The advantage of the GPAPF tracker is that it can recover from poor estimations
Algorithm 2: The GPAPF algorithm

Initialization: \( \left\{ \Lambda^{(i)}_{n,M}; \omega^{(i)}_{n,M}; \frac{1}{N} \right\}_{i=1}^{N_p} \)

for each frame \( n \)
for \( m = M \) down to 1 do
1. Calculate \( \Omega^{(i)}_{n,M} = \varphi(\omega^{(i)}_{n,M}) \) applying the prelearned by GPLVM mapping
   function \( \varphi \) on the set of particles \( \left\{ \omega^{(i)}_{n,M} \right\}_{i=1}^{N_p} \).
2. Calculate the weights of each particle:
   \[
   \pi^{(i)}_n = k \frac{w^m(y_n, \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}) p\left( \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m} | \Lambda^{(i)}_{n,m-1}, \omega^{(i)}_{n,m-1} \right)}{q\left( \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m} | \Lambda^{(i)}_{n,m-1}, \omega^{(i)}_{n,m-1}, y_n \right)},
   \]
where \( k \) is a normalization factor needed for satisfaction of constraint \( \sum_{i=1}^{N_p} \pi^{(i)}_n = 1 \).
Now the weighted set is constructed, which will be used to draw particles for the next layer.
3. Draw \( N \) particles from the weighted set \( \left\{ \Lambda^{(i)}_{n,m}; \omega^{(i)}_{n,m}; \pi^{(i)}_{n,m} \right\}_{i=1}^{N_p} \) with replacement and with distribution \( p\left( \Lambda = \Lambda^{(i)}_{n,m}, \omega = \omega^{(i)}_{n,m} \right) = \pi^{(i)}_{n,m} \).
4. Calculate \( \left\{ \Lambda^{(i)}_{n,m-1}; \omega^{(i)}_{n,m-1} \right\} \sim q\left( \Lambda^{(i)}_{n,m-1}; \omega^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}, y_n \right) \), which can be rewritten as \( \Lambda^{(i)}_{n,m-1} \sim q\left( \Lambda^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, y_n \right) = \Lambda^{(i)}_{n,m} + n^\Lambda_m \) and \( \omega^{(i)}_{n,m-1} \sim q\left( \omega^{(i)}_{n,m-1} | \omega^{(i)}_{n,m}, y_n \right) = \omega^{(i)}_{n,m} + n^\omega_m \), where \( n^\Lambda_m \) and \( n^\omega_m \) are multivariate Gaussian random variables.
end for

- The optimal configuration can be calculated using the following formula: \( \Lambda_n = \sum_{i=1}^{N_p} \Lambda^{(i)}_{n,1} \) and \( \omega_n = \omega^{(i)\min}_{n,1} \), where \( i_{\text{max}} = \arg\min_i \pi^{(i)}_{n,1} \).

- The unweighted particle set for the next observation is produced using \( \Lambda^{(i)}_{n+1,M} = \Lambda^{(i)}_{n,1} + n^\Lambda_1 \) and \( \omega^{(i)}_{n+1,M} = \omega^{(i)}_{n,1} + n^\omega_1 \), where \( n^\Lambda_1 \) and \( n^\omega_1 \) are multivariate Gaussian random variables.

end for
after several frames. This is because particles generated in the latent space are more likely to represent valid poses. Furthermore, because of its low dimensionality, the latent space can be covered with a relatively small number of particles. The particle generation is based on modification of the pose retrieved in the previous frame. Most of these particles resemble this pose, but some differ greatly. Although there are only a few such particles, still, due to the low dimensionality of the problem, some of them will resemble the correct pose on the current frame. If the retrieved pose was estimated correctly, the tracker will choose the particle that most resembles the pose on the current frame. If the retrieved pose was miscalculated, the tracker will consider particles that bear little resemblance to the pose from the previous frame. However, because the weighting function is expected to assign higher values for such particles, the next layers of the annealing process will generate many particles that resemble the correct pose. In this way, the pose is likely to be estimated properly, despite the mistracking in the previous frame. This is shown in Fig. 3.7.

Algorithms such as CONDENSATION or the annealed particle filter algorithm suffer from large variance in data space, which may cause the generation of unnatural poses. In the proposed latent space approach, particles correspond to natural poses with low variance. This due to the latent space being learned from natural poses. The unnatural poses correspond to the points with large variance (the black regions in Fig. 3.5.(a)). It is thus highly unlikely that a particle corresponding to such a location will be generated. Therefore the effective number of particles is higher, which enables more accurate tracking.

The drawback of this approach is that it requires more calculation than the regular annealed particle filter due to the transformation from the latent space into the data space. However, as mentioned above, if same number of particles is used, the number of effective poses is significantly higher in the GPAPF than in the original annealed particle filter. Therefore, we can reduce the number of particles for the GPAPF tracker and compensate for the additional calculations.
3.4.1 Prediction

Although the reduction performed by the GPLVM significantly reduces the dimensionality of the state space, it is a static model and does not allow prediction of the next latent location. Moreover, it does not produce a smooth path in the latent space even if smooth sequences in the pose space were used during learning. Therefore, we need to be able to incorporate the dynamic aspects of the motions into the latent space. This can be done using the Gaussian Process Dynamic Model (GPDM) [167], which can provide the tracker with a motion prior.

A further problem is that in the latent space generated by either GPLVM or GPDM (see Fig. 3.5), points close in latent space are constrained to be close in data space, but the converse is not true. Therefore, we have used back constraints, introduced by Lawrence and Candela [80], in order to force the latent points to be a smooth function of the data points. This means that points that are close in data space are constrained to be close in latent space. Fig. 3.8 shows the latent spaces that were generated for the 3 different actions (left leg kicking, right leg kicking and object lifting) by GPLVM (left) and GPDM with back constraints (right). In the latent space shown in Fig. 3.8(a), each point has information only about the corresponding pose and therefore can appear in the sequences of different motion types. For example, any magenta point represents a pose which can appear during lifting and putting down an object. In the latent space in Fig. 3.8(b), each latent location represents not only the actual pose but also the dynamics (the dependency between sequential poses). Therefore, the same poses will have different locations depending on the poses that appeared in the past. Consequently, the same poses that appear during lifting and putting down an object will now have different locations in the latent space. This is a very important property, and we will address it later for the classification task.

As mentioned above, GPDM produces a dynamics prior. Fig. 3.9 shows the dynamics prior that is generated for the walking sequence and the 3-action latent space. This dynamics provides a motion prior for the tracker. Instead of generating a new particle by adding a multivariate Gaussian random variable, we can shift the particle...
Figure 3.8: The latent spaces that were generated for the 3 different actions (left leg kicking, right leg kicking and object lifting) by (a) GPLVM and (b) GPDM with back constraints.

using this prior and then add a random variable with a lower variance. Not only do this prior and the continuous latent space allow for better tracking, they also make it possible to track the body parts with a significantly lower number of particles. Fig. 3.10 shows the average errors for the walking sequences for both the GPLVM and GPDM algorithms. The left graph shows the results for the GPAPF tracker with 500 particles (100 per layer) and the right graph shows the error of the GPAPF tracker with 100 particles (20 per layer). While both algorithms have similar errors when a large number of particles is used, this is not the case for a small number. We can see that the performance of the GPDM based tracker is better.

3.4.2 Towards precise tracking

The problem with a 2-stage approach is that a Gaussian field is not capable of describing all possible poses. As mentioned above, this approach resembles using probabilistic PCA in order to reduce the data dimensionality. However, for tracking issues, we want the pose estimation to be as close as possible to the actual one. Therefore, we add an additional annealing layer as the last step. This layer consists of only one stage. We generate particles in the full data space using the particles from the previous 2-stage annealing layer, described in the previous section. This is done with
Figure 3.9: The dynamics prior generated by GPDM: (a) the walking sequence dynamics; (b) 3-action (kicking with the left leg, kicking with the right leg, and picking up an object) dynamics.

very low variances in all the dimensions, which are almost equal for all actions, as the purpose of this layer is to make only slight changes in the final estimated pose. Thus it does not depend on the actual frame rate. This is in contrast to the original annealed particle filter tracker, where the model parameters (the variances for each layer) must be updated.

The final scheme of each step is shown in Fig. 3.11 and described in Algorithm 3. Suppose we have $M$ annealing layers, as explained in Section 3.4. Then we add one more single-stage layer. In this last layer $\Omega_{n,0}$ is calculated using only $\Omega_{n,1}$ without calculating $\omega_{n,0}$. We should note that the last layer has no influence on the tracking quality in the following frames, as $\omega_{n,1}$ is used for the initialization of the next layer.

Fig. 3.12 shows the difference between the version without the additional annealing layer and the results after adding it. We used five 2-stage annealing layers in both cases. For the second tracker we added an additional single-stage layer. Fig. 3.13 shows the error graphs produced by two trackers. The error was calculated by comparing the trackers’ output with the result of the MoCap system. This comparison was suggested by A. Balan [12]. The error measure is computed by calculating the 3D distance between the locations of the different joints (hips, knees, and so forth),
Algorithm 3: The GPAPF algorithm with the additional layer

Initialization: \( \left\{ \Lambda^{(i)}_{n,M}, \omega^{(i)}_{n,M}, \frac{1}{N} \right\}_{i=1}^{N_p} \)

for each: frame \( n \)

for each: frame \( n \)

1. Calculate \( \Omega^{(i)}_{n,M} = \phi \left( \omega^{(i)}_{n,M} \right) \) applying the prelearned by GPDM mapping function \( \phi \) on the set of particles \( \left\{ \omega^{(i)}_{n,M} \right\}_{i=1}^{N_p} \).

2. Calculate the weights of each particle:
   \[
   \pi^{(i)}_n = k \frac{\omega^{(i)}_{n,M} p(\Lambda^{(i)}_{n,M}, \omega^{(i)}_{n,M} | \Lambda^{(i)}_{n,M-1}, \omega^{(i)}_{n,M-1})}{q(\Lambda^{(i)}_{n,M}, \omega^{(i)}_{n,M} | \Lambda^{(i)}_{n,M-1}, \omega^{(i)}_{n,M-1}, y_n)},
   \]
   where \( k \) is a normalization factor needed for satisfaction of constraint \( \sum_{i=1}^{N_p} \pi^{(i)}_n = 1 \).

3. Draw \( N \) particles from the weighted set \( \left\{ \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}, \pi^{(i)}_m \right\}_{i=1}^{N_p} \) with replacement and with distribution \( p(\Lambda = \Lambda^{(i)}_{n,m}, \omega = \omega^{(i)}_{n,m}) = \pi^{(i)}_m \).

4. Calculate \( \Lambda^{(i)}_{n,m-1}, \omega^{(i)}_{n,m-1} \sim q(\Lambda^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}, y_n) \), which can be rewritten as \( \Lambda^{(i)}_{n,m-1} \sim q(\Lambda^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}, y_n) = \Lambda^{(i)}_{n,m} + n^\Lambda_m \) and \( \omega^{(i)}_{n,m-1} \sim q(\omega^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \omega^{(i)}_{n,m}, y_n) = \omega^{(i)}_{n,m} + n^\omega_m \), where \( n^\Lambda_m \) and \( n^\omega_m \) are multivariate Gaussian random variables.

end for

- The optimal configuration can be calculated by the following steps:

1. Calculate \( \left\{ \Lambda^{(i)}_{n,m-1}, \Omega^{(i)}_{n,m-1} \right\} \sim q(\Lambda^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \Omega^{(i)}_{n,m}, y_n) \), which can be rewritten as \( \Lambda^{(i)}_{n,m-1} \sim q(\Lambda^{(i)}_{n,m-1} | \Lambda^{(i)}_{n,m}, \Omega^{(i)}_{n,m}, y_n) = \Lambda^{(i)}_{n,m} + n^\Lambda_m \) and \( \Omega^{(i)}_{n,m-1} \sim q(\Omega^{(i)}_{n,m-1} | \Omega^{(i)}_{n,m}, y_n) = q(\Omega^{(i)}_{n,m-1} | \Omega^{(i)}_{n,m}, y_n) = \phi(\Omega^{(i)}_{n,m}) + n^\Omega_m \), where \( n^\Lambda_m \) and \( n^\Omega_m \) are multivariate Gaussian random variables.

2. Draw \( N \) particles from the weighted set \( \left\{ \Lambda^{(i)}_{n,m}, \Omega^{(i)}_{n,m}, \pi^{(i)}_m \right\}_{i=1}^{N_p} \) with distribution \( p(\Lambda = \Lambda^{(i)}_{n,m}, \Omega = \Omega^{(i)}_{n,m}) = \pi^{(i)}_m \). Calculate the weight of each particle.

3. The optimal configuration is \( \Lambda_n = \sum_{i=1}^{N_p} \pi^{(i)}_n \Lambda^{(i)}_{n,0} \) and \( \Omega_n = \sum_{i=1}^{N_p} \pi^{(i)}_n \Omega_{n,0} \).

- The unweighted particle set for the next observation is produced using \( \Lambda^{(i)}_{n+1,M} = \Lambda^{(i)}_{n,1} + n^\Lambda_0 \) and \( \omega^{(i)}_{n+1,M} = \omega^{(i)}_{n,1} + n^\omega_0 \), where \( n^\Lambda_0 \) and \( n^\omega_0 \) are multivariate Gaussian random variables.

end for each
Figure 3.10: The average errors for the walking sequences for both GPLVM and GPDM algorithms with (a) 500 particles (100 per layer) and (b) 100 particles (20 per layer). The red crosses represent the GPLVM based tracker errors and the blue circles represent the GPDM based tracker.

as estimated by the MoCap system and by the trackers’ results. The distances are summed and multiplied by the weight of the corresponding particle. Then the sum of the all weighted distances is calculated. We can see that the error produced by the GPAPF tracker with the additional layer (blue circles on the graph) is lower than the one produced by the original GPAPF algorithm without the additional annealing layer (red crosses on the graph) for the walking sequence taken at 30 fps. However, the improvement is not dramatic. This is explained by the fact that the difference between the estimated pose using only latent space annealing and the actual pose is not very large. This suggests that the latent space accurately represents the data space.

We can also see that the improved GPAPF has fewer peaks on the error graph. The peaks stem from the fact that the \textit{argmax} function, used to find the optimal configuration, is very sensitive to the location of the best fitting particle. In the improved version, we calculate the weighted average of all the particles. We observed in our experiments that many particle weights are close to their optimal value. Therefore, the result is less sensitive to the location of some particular particle. Rather it depends on the entire particle set.
Figure 3.11: GPAPF with additional annealing layer. The black solid arrows represent the dependencies between state and visual data; the blue arrows represent the dependencies between the latent space and the data space; dashed magenta arrows represent the dependencies between sequential annealing layers; the red arrows represent the dependencies of the additional annealing layer. The green arrows represent the dependency between sequential frames.

We have tried to use the results produced by the additional layer in order to initialize the state in the next time step. This was done by applying the inverse function $\varphi^{-1}$, suggested by Lawrence [80], on the particles generated in previous annealing layer. However, this approach did not produce any valuable improvement in the tracking results. The computationally heavy inverse function greatly increased the calculation time. Therefore, we decided not to experiment with it further.

3.5 Results

3.5.1 Single Person

We tested the GPAPF tracking algorithm using the HumanEva, HumanEvaI and HumanEvaII datasets [144]. The sequences show different activities, such as walking, boxing, and jogging, which were captured by several synchronized and mutually calibrated cameras; we used only 4 video inputs in our evaluation. For the HumanEva and HumanEvaI datasets we used 4 grayscale image streams and for HumanEvaII we used 4 color streams. The sequences were captured using the MoCap system, which provides the correct 3D locations of body joints such as shoulders and knees, for evaluation of the results and comparison to other tracking algorithms. Given the
locations of $N$ joints estimated by the MoCap system $G = \{g_i\}_{i=1}^N$, and their locations estimated by a tracker $P = \{p_i\}_{i=1}^N$, the average error $\varepsilon$ can be calculated using:

$$\varepsilon(G, P) = \frac{1}{N} \sum_{i=1}^{N} \| p_i - g_i \|^2 \quad (3.5.1)$$

We also used an online evaluation system for these datasets (http://www.cs.brown.edu/ ls/ehum2/submit.html).

The first sequence shows a person walking in a circle. The video was captured at 60 fps. We tested the annealed particle filter based body tracker, implemented by A. Balan et al.[12], and compared the results with the ones produced by the the GPAPF tracker. The error was calculated by comparing the tracker’s output with that of the MoCap system, using the average distance in millimeters between 3-D
joint locations, as explained in Section 3.4. Fig. 3.14 shows the error graphs produced by the GPAPF tracker (based on the GPDM latent space) (blue circles) and by the annealed particle filter (red crosses) for the walking sequence taken at 30 fps. As can be seen, the GPAPF tracker produces a more accurate estimation of the body location. Similar results were achieved for 15 fps. Fig. 3.15 presents sample images with the actual pose estimation for this sequence. The poses were projected to the first and second cameras. The first 2 rows show the results of the GPAPF tracker. The third and fourth rows show the results of the annealed particle filter. In order to more accurately compare GPAPF and APF, we added a simple dynamic model. For both trackers we used 5 annealing layers with 100 particles per layer. We calculated average velocities for each part and added them during the propagation between sequential frames. However, because of differences in acceleration during the different stages of the same action, such a simple model was not helpful. The main reason for the APF failure was that the range of possible part locations significantly increased when the sequence was downsampled from 60 fps (used by Balan et al. [12]) to 30 fps. In order to compensate for this, we had to increase the variance of the Gaussian noise to allow the pose to be meaningfully different from the one observed in the previous frame. In practice, this meant that we had to increase the number of particles. While the GPAPF tracker was able to maintain robust results using the same number of particles, we had to increase both the number of layers and particles per
In order to achieve a similar error rate, we had to use (as suggested by Deutscher and Reid [38]) 10 annealing layers with 400 particles in each. This means that APF used 4000 likelihood evaluations (8 times more than GPAPF).

In our experiments we varied the number of particles from 100 up to 1000. For the setup that uses 100 particles per layer with 5 layers on a Pentium IV 3GHz, the computational cost was 30 seconds per frame. Using the same number of particles and layers in the annealed particle filter algorithm takes 20 seconds per frame. The annealed particle filter algorithm could not track the body pose with a low number of particles for the 30 fps and 15 fps videos. Therefore, we had to increase the number of particles used in the annealed particle filter to 500 in order to make the comparison. Both algorithms were implemented in Matlab and were executed with no optimization packages.

We also tried to compare the performance of the GPAPF algorithm to that of the CONDENSATION algorithm. But because CONDENSATION yielded very poor results or a very large number of particles was required to implement it, this algorithm was computationally ineffective and we do not report the results here.

Next we used GPDM in order to create a latent space that combines several
different behaviors: *walking, jogging, and balancing*. We used this latent space to track the body parts on the images from the HumanEvaI and HumanEvaII datasets. Fig. 3.16 shows the results of tracking the boxing sequence from the HumanEvaI(S1) dataset and Fig. 3.17 shows the results for HumanEvaII(S2). Fig. 3.19 shows the errors of the annealed tracker (red crosses) and the GPAPF tracker (blue circles): (top) HumanEvaI (S1, walking1, frames 6-590); (middle) HumanEvaII(S2, frames 1-1202) (marked by blue stars); (bottom) HumanEvaII (S4, frames 2-1258) (marked by red circles). The average errors for these sequences are presented in Table 3.5.1. The errors were calculated using the online evaluation system. We can see that the GPAPF tracker is very stable.

We also conducted several experiments with the sequences that were captured in our lab. We captured different activities, performed by different actors. Some of the activities (boxing and jogging, for instance) are present in the HumanEva datasets, while others (such as object lifting and sitting down) are not. The frame rate of these videos is 15 fps. We manually marked some of the sequences in order to produce the needed training sets for GPDM. The results of the tracking of different types of motions are shown in Fig. 3.18.
Figure 3.17: GPAPF tracking results. Sample frames from the combo1 sequence from the HumanEvaII (S2) dataset.

Figure 3.18: GPAPF tracking results. Sample frames from the running, leg movements, and object lifting sequences.
Figure 3.19: The errors of the annealed tracker (red crosses) and the GPAPF tracker (blue circles): (top) HumanEvaI (S1, walking1, frames 6-590); (middle) HumanEvaII (S2, frames 1-1202) (marked by blue stars); (bottom) HumanEvaII (S4, frames 2-1258) (marked by red circles).

<table>
<thead>
<tr>
<th>Dataset Sequence</th>
<th>HumanEvaI</th>
<th>HumanEvaII</th>
<th>HumanEvaII</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF</td>
<td>95.4</td>
<td>163.8</td>
<td>172.1</td>
</tr>
<tr>
<td>GPAPF</td>
<td>86.3</td>
<td>86.6</td>
<td>89.0</td>
</tr>
</tbody>
</table>

Table 3.1: The average errors for HumanEvaI (S1, walking1), HumanEvaII (S2) and HumanEvaII (S4) sequences produced by APF and GPAPF. The error measures the average distance in millimeters between the joint locations.
3.5.2 Two Person

We performed experiments involving two subjects. We created a database with two humans performing different actions, such as handshaking, kicking, hugging, and passing an object. We used four cameras with a 30 fps sample rate to capture these interactions. In addition, we used the MoCap system to obtain the ground truth location of the body joints of both people in the scene. This data was used for the HGPLVM training and for error calculation. The error was calculated by comparing the tracker’s output to the ground truth, using the average (per joint) distance in millimeters between the 3D joint locations of both people.

Figure 3.20 demonstrates some sample frames from the kicking action. Despite small errors in the locations of the small body parts, like the lower part of arms, the tracking results look are correct. However, when a more complex actions are used,
Figure 3.21: Failure of GPAPF tracking results for 2 persons interaction. Sample frames from the hugging sequence. Each row corresponds to a different scene view (different camera).
Table 3.2: The average errors of the three algorithms: the top row shows the results of APF, the second row shows the results of GPAPF with two independent latent spaces (GPAPF1), and the third row shows the results of GPAPF with a joint latent space (GPAPF2). The columns represent the motion types: *handshaking*, *kicking*, and *hugging*.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Handshaking</th>
<th>Kicking</th>
<th>Hugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF</td>
<td>142.1</td>
<td>147.3</td>
<td>200.3</td>
</tr>
<tr>
<td>GPAPF1</td>
<td>102.6</td>
<td>115.9</td>
<td>163.6</td>
</tr>
<tr>
<td>GPAPF2</td>
<td>102.4</td>
<td>120.2</td>
<td>152.2</td>
</tr>
</tbody>
</table>

like hugging which has a very high rate of occlusion, the tracking results are less reliable. We compared the results produced by the GPAPF and APF trackers for several motion types involving two humans. As before we used 500 particles per layer for the APF algorithm. The GPAPF algorithm was tested in two different ways. The first (denoted by GPAPF1) was to use a separate latent space for each subject (two latent spaces in all) with five annealing layers and 100 particles for each subject; the second way (denoted by GPAPF2) was to apply a combined latent space describing both subjects (their poses and relative position) with five annealing layers and 200 particles for each. In both cases we used GPDM for learning.

Table 3.5.2 presents the average error for these algorithms for three different actions: *handshaking*, *kicking*, and *hugging*. Each row represents a different tracking algorithm: the top row shows the results of APF, the second row shows the results of GPAPF with two independent latent spaces, and the third row shows the results of GPAPF with a joint latent space. The columns represent the motion types. Figure 3.22 shows the error graphs for these actions. Not only is the average error of GPAPF significantly lower than that of APF, but the error graphs of GPAPF also have fewer peaks, which emphasizes the relative robustness of the algorithm.
3.6 Discussion

We have introduced an approach that uses GPLVM and GPDM in order to reduce dimensionality and thus improve the ability of the annealed particle filter to track an object even in a high dimensional space. We show that our tracking algorithm is capable of performing simultaneous articulated human body tracking for two subjects. We also show that using GPDM makes it possible to recover from temporal target loss. We demonstrate the tracker’s robustness and ability to track two humans simultaneously.

Tracking of interactions between multiple actors is an interesting problem. The main challenge, which is discussed in chapter 5, is constructing a latent space. As can be inferred from our experiments, while an individual person’s poses can be described using a low dimensional space, this may not be the case for multiple people. The other problem here is that occlusion is very likely when more than one person is being tracked. Furthermore, while for one-person scenes each body part can be seen from at least one camera, that is not the case for crowded scenes. Although our method is capable of tracking multiple subjects, the results are much less accurate and the tracker less robust than for the one person case.
Furthermore, as we have mentioned above, a 2D latent space that combines 5 different actions contains many gaps between sequential poses and thus is less suitable for tracking. However, when we increase the dimensionality to 3D, the latent space becomes significantly smoother, which allows more accurate tracking. How to choose the correct dimensionality of the latent space requires further investigation.
Chapter 4

Action Classification Using GPAPF

4.1 Classification algorithm

Humans in motion can perform different actions. Our aim is to classify these actions into predefined action types. The classification of actions is based on the sequences of poses detected by the tracker during the performed motion. To classify these actions, we generate templates (which will be described later) for each one (walking, kicking, waving, and so forth). Suppose there are $K$ different motion types. Each type $k$ is represented by an action template $\Psi_k$, which consists of a sequence of $l_k + 1$ latent coordinates $\Psi_k = \{\psi_0, ..., \psi_{l_k}\}$. The GPAPF tracker generates a sequence of $l + 1$ latent coordinates: $\Upsilon = \{\varphi_0, ..., \varphi_l\}$ (see Fig. 4.2 for an illustration). This sequence is compared to the template and the one with the smallest distance is chosen to represent the type of action.

4.1.1 GPAPF based on GPLVM vs. GPAPF based on GPDM

To successfully classify the actions, we have to address the differences between the tracking results of the GPAPF tracker that is based on the GPDM latent space and the GPAPF tracker that is based on the GPLVM. As mentioned before, each latent location in the GPDM latent space represents a concrete pose and the motion history (see Fig. 4.1.(a)). However, the GPLVM locations do not contain any information
Figure 4.1: Visualization of the classification task with (a) GPDM and (b) GPLVM latent spaces. The latent space contains 3 different action templates: lifting an object (red), kicking with the left (yellow) and the right (magenta) legs. The blue line represents the tracker output; the cyan dots represent the actual locations in the latent space, as estimated by the tracker. The classification task is to find the template closest to the tracker’s result.

regarding the motion’s dynamics or the previous locations (see Fig. 4.1.(b)). For example, sitting down and standing up are two actions that consist of approximately equal poses in a different order. While the GPDM tracker will produce two different curves, the tracker based on the GPLVM latent space will produce two similar curves with opposite directions. Therefore, given a set of latent locations produced by the GPDM based tracker, we can compare them to the set of the locations that represent the action templates: we need not consider the sequential information. But in order to classify the results of the GPLVM tracker, we need to compare the sequences, which is much a harder task.

4.1.2 Classification using GPAPF tracker based on GPLVM

In order to compare the two sets of points, we have a distance transform of sampled curves, suggested by Felzenszwalb and Huttenlocher [44]. According to this metric, the distance between two sampled curves, represented by $\Sigma = \{\sigma_0, ..., \sigma_n\}$ and $\Gamma = \{
Figure 4.2: Graphical illustration of the action template (black line), denoted by \( \Psi_k = \{\psi_0, ..., \psi_l\} \) and a sequence produced by the tracker, (red line), denoted by \( \Upsilon = \{\varphi_0, ..., \varphi_l\} \).

\( \{\gamma_0, ..., \gamma_l\} \), can be calculated using the following formula [44]:

\[
D(\Sigma, \Gamma) = \max \left( d(\Sigma, \Gamma), d(\Gamma, \Sigma) \right) \tag{4.1.1}
\]

where

\[
d(\Sigma, \Gamma) = \max_{1 \leq i \leq n} \min_{1 \leq j \leq l} \| \sigma_i, \gamma_j \| \tag{4.1.2}
\]

The sequence \( \Upsilon \), produced by the tracker, is compared to each one of the templates \( \Psi_k \). The template with the smallest distance is chosen to represent the type of action.

### 4.1.3 Classification using GPAPF tracker based on GPDM

In order to compare two curves we use a modified Fréchet distance (see Alt et al. [6]). This metric takes into consideration the direction of the compared curves. The method is quite tolerant to position errors, and thus the classification can tolerate the tracking errors.

We define polygonal curve \( P^\Sigma \) and \( P^\Gamma \) as continuous and piecewise linear curves made of segments connecting vertexes \( \Sigma = \{\sigma_0, ..., \sigma_n\} \) and \( \Gamma = \{\gamma_0, ..., \gamma_l\} \) correspondingly. The curves can be parameterized with a parameter \( \alpha \in [0, n] \) and \( \beta \in [0, l] \), where \( P^\Sigma(\alpha) \) refers to a given position on the curve \( P^\Sigma \), with \( P^\Sigma(0) \) denoting \( \sigma_0 \) and \( P^\Sigma(n) \) denoting \( \sigma_n \), and \( P^\Gamma(\beta) \) refers to a given position on the curve
\( P^\Gamma \), with \( P^\Gamma (0) \) denoting \( \gamma_0 \) and \( P^\Gamma (l) \) denoting \( \gamma_l \). The distance between the two curves is defined as [6]:

\[
F \left( P^\Sigma, P^\Gamma \right) = \min_{\alpha [0,1] \rightarrow [0,n], \beta [0,1] \rightarrow [0,l]} f \left( P^\Sigma (\alpha), P^\Gamma (\beta) \right),
\]

(4.1.3)

where

\[
f \left( P^\Sigma (\alpha), P^\Gamma (\beta) \right) = \max_{t \in [0,1]} \| P^\Sigma (\alpha(t)) - P^\Gamma (\beta(t)) \|_2
\]

(4.1.4)

and \( \alpha(t) \) and \( \beta(t) \) represent sets of continuous and increasing functions, with \( \alpha(0) = 0 \), \( \alpha(1) = n \), \( \beta(0) = 0 \), \( \beta(1) = l \). Once again, the curve generated by the tracker is compared to each action template and assigned to the class with the smallest distance.

### 4.1.4 Classification: the combined approach

While in general it is hard to calculate the Fréchet distance, Alt et al. [6] proposed an efficient algorithm to calculate it between two piecewise linear curves. As explained in Section 3, the tracker that is based on the GPDM latent space produces better results than the GPLVM-based tracker. Moreover, as mentioned above, the results produced by the GPDM-based tracker are easier to classify, as we can compare sets rather than sequences. However, the drawback is that the GPDM-based tracker produces poor results if the type of motion changes while the action is being performed. For example, if after picking up an object a person decides to perform a different action, the initial locations in the latent space will be very distant from the corresponding locations in the action template. Therefore we suggest using a combined approach, which results in good tracking with the GPDM-based tracker and robust classification with the GPLVM. We use GPDM for tracking only. For action recognition, we project the GPDM locations to the full data space and then back-project them to the GPLVM latent space. The classification should thus be performed on these back-projections on the GPLVM latent space using the aforementioned Fréchet distance.

### 4.2 Results

The classification algorithm was tested on two different datasets. The first set contained 3 different activities: (1) lifting an object, kicking with (2) the left and (3) the
right leg. For each activity 5 different sequences were captured. We used one sequence for each motion type in order to construct the action templates. We experimented with the GPLVM and GPDM based trackers and compared the results with the combined approach, introduced in Section 4.1.4. Fig. 4.3 shows the trajectories in the latent spaces produced by the trackers. The first row represents the GPLVM based tracker; the second row represents the GPDM based tracker; finally, the third row represents the results of the GPDM based tracker projected on the GPLVM latent space.

Figure 4.3: Tracking trajectories in the latent space for different activities: (a) lifting an object, kicking with (b) the left and (c) the right legs. The first row represents the GPLVM based tracker; the second row represents the GPDM based tracker; the third row represents the combined approach: the GPDM based tracker locations are transformed to the data space and back-projected on the GPLVM latent space. In each image the black lines represent the templates and colored lines represent the trajectories produced by the GPAPF tracker.
Table 4.1: Classification accuracy, using GPLVM based tracker, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th></th>
<th>Hand waving</th>
<th>Object lifting</th>
<th>Kicking</th>
<th>Sitting down</th>
<th>Punching</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand waving</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>70%</td>
</tr>
<tr>
<td>Object lifting</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>75%</td>
</tr>
<tr>
<td>Kicking</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Sitting down</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>70%</td>
</tr>
<tr>
<td>Punching</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>65%</td>
</tr>
</tbody>
</table>

The second set contained 5 different activities: (1) hand waving, (2) lifting an object, (3) kicking, (4) sitting down, and (5) punching. Once again 5 different sequences were captured for each activity. The sequences were classified by cross-validation. Once again, we experimented with the two different trackers and with the combined approach. The classification results of the GPLVM based classifier are shown in Table 4.1; the results of the GPDM based classifier are shown in Table 4.2; finally, the results of the combined approach are shown in Table 4.3. Different types of actions involving hand gestures were difficult to classify because they all look alike. Actions involving sitting down or object lifting were difficult to classify because the high rate of self-occlusions caused it to incorrectly estimate the actual poses.

The third experiment was to classify the interaction between two subjects. As mentioned above, we created a database with two humans performing different actions, such as handshaking, kicking, hugging, pushing, and passing an object, similar to the database used by Park et al. [111]. Classifying interactions is a very difficult task because the image of one person often occludes the other, making it hard to use image-based methods. Moreover, some types of motion closely resemble others. As in the previous experiments, cross-validation was applied. We used the GPDM based GPAPF tracker to produce the trajectories in the latent space and tried to perform classification of these interactions. The classification results are presented in Table 4.4.
Table 4.2: Classification accuracy, using GPDM based tracker, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th>Hand waving</th>
<th>Object lifting</th>
<th>Kicking</th>
<th>Sitting down</th>
<th>Punching</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>75%</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>85%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>80%</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>70%</td>
</tr>
</tbody>
</table>

Table 4.3: Classification accuracy, using the combined approach, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th>Hand waving</th>
<th>Object lifting</th>
<th>Kicking</th>
<th>Sitting down</th>
<th>Punching</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>80%</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>85%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>80%</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>75%</td>
</tr>
</tbody>
</table>
Table 4.4: Classification accuracy for interactions between two people (using GPDM latent space) for 6 different activities: Hand shaking, Kicking, Hugging, Punching, Pushing, and Object passing. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th></th>
<th>Hand shaking</th>
<th>Kicking</th>
<th>Hugging</th>
<th>Punching</th>
<th>Pushing</th>
<th>Object passing</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand shaking</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>80%</td>
</tr>
<tr>
<td>Kicking</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Hugging</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>65%</td>
</tr>
<tr>
<td>Punching</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>4</td>
<td>1</td>
<td>70%</td>
</tr>
<tr>
<td>Pushing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>75%</td>
</tr>
<tr>
<td>Object passing</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>80%</td>
</tr>
</tbody>
</table>

The results are significantly better than the ones presented by Park et al. [111]. There are two reasons for the low success rate for some actions. We can see that the Hugging action was often assigned a wrong motion class, while no other motion was incorrectly classified as Hugging. This is because the tracker was unable to robustly track this action. When two people actually hug one another, the tracker cannot estimate the body part locations correctly. Consequently, the classification was not successful. Punching and pushing were also misclassified due to the limitation of the latent space produced by GPDM. The action templates (trajectories) are very similar, which makes classification difficult.
Chapter 5

Hierarchical Annealed Particle Filter

5.1 Motivation: From GPAPF to HAPF

The drawback of the GPAPF algorithm is that a latent space is not capable of describing all possible poses. The particles generated in the latent space describe only the poses that resemble those used in the learning process. On the one hand, this property contributes to the robustness of the tracker. On the other hand, it reduces the pose detection precision of the tracker. Moreover, the space reduction must capture dependencies between the poses of the different body parts. For example, if there is any connection between the parameters describing the pose of the left hand and those describing the right hand, then we can easily reduce the dimensionality of these parameters. However, if a person performs a new movement which differs from those already learned, then the new poses will be represented less accurately by the latent space.

Therefore, after experimenting with the GPAPF algorithm we concluded that it is hardly possible to improve the quality of the results achieved with GPAPF. We have tried to use sophisticated methods for edge detection, foreground object detection, changed body model representation as well as tried to construct different latent spaces. Non of these methods caused a significant improvement. Specifically, when an action contains several motion types or previously unseen motion the error rate was high. We observed same behavior of GPAPF algorithm for sequences with large
occlusions and self-occlusion such as "sitting down". The other concern is the ability of GPAPF to track the body parts during the transition between different motion types. Even if both motions were used in the learning to produce a common latent space, the error rate during the transition phase was relatively high.

One of the purposes of this research is to develop a robust tracking algorithm, capable of simultaneously tracking of two subjects. We want to benefit from the reduced dimensionality of the state space but, at the same time, to be able of tracking new, previously unobserved motion types and handle situations with combined motions and high occlusion rate. Additionally, the tracker should be computationally efficient. In order to use GPAPF for tracking several humans we have to either define a new state to contain the location and pose information for all the people in the scene or to track each person independently. For example, GPAPF uses 31-dimensional state space for a single person. The first option is to use a 62-dimensional state space, describing two subjects. In this case we had to apply the dimensionality reduction of a larger state space, which is likely to produce poor results. Furthermore, a small change in the relative position or synchronization between the people may cause a significant change in the corresponding latent position. In particular, the series of the latent coordinates that describe the same action may be very different. This will make it hard to perform tracking using such a latent space. As suggested above, the alternative is to track each person independently. In this case, we will have to deal with a very high occlusion rate, which results in a high error rate. In addition, in the scenarios involving several humans, there are situations in which some body parts cannot be seen from any camera. Using the knowledge of the interaction between people in the scene we can estimate the pose of the missing part. Therefore, it is essential to use the information about other people in the scene in order to achieve efficient and robust tracking.

In order to be able to deal with all the stated above difficulties we propose Hierarchical Annealing Particle Filter (HAPF). Instead of using a single latent space to describe the human body we use a hierarchy of the latent spaces. The root node describes the whole body, while the leaves describe a individual body parts. This hierarchical decomposition of the body makes the algorithm capable of a recovery of
novel poses that are not present in the training set. The algorithm estimates pose by using different hierarchy levels, rather than a single latent space describing the whole body. This allows us to generate particles that represent not only the poses similar to the ones used in the training set, but also their combinations, such as walking and waving, running with a raised arm etc.

5.2 Hierarchical Gaussian Process Latent Variable Model

In many situations, when modeling high dimensional data, one wants to express conditional independencies in the data as well as the manifold structure. The HGPLVM [81] is a form of GPLVM with a hierarchical latent representation (e.g., see Figure 5.2). This structure allows to model independently different subsets of the data. The leaves of the latent model represent a GPLVM model for the corresponding subsets of the data. To capture the dependencies between these subsets one can use Gaussian Processes to model the joint distribution. For example in Figure 5.2 the left leg and right leg are coordinated by the lower body (legs) latent variable. Given a lower body latent position, there is a GP mapping to latent positions for each leg, and there are GP mappings from the latent spaces that represent each leg to the joint latent space of the two legs.

Suppose we want to model interaction between two people. In this case we want the latent space to capture the interaction itself. As mentioned above, HGPLVM is capable of modeling each subject independently. The hierarchy that can describe this conditional dependency is shown in Figure 5.1. The joint probability distribution represented by this graph is given by [81]:

\[
P(Y_1, Y_2) = \int (P(Y_1|X_1)) \times \int (P(Y_2|X_2)) \times \int (P(X_3, X_2|X_3))dX_3dX_2dX_1
\]

where each distribution is a Gaussian process. The MAP solutions for finding the values of the latent variables for this model is maximization of [81]:

\[
P(X_1, X_2, X_3|Y_1, Y_2) = \log(P(Y_1|X_1)) + \log(P(Y_2|X_2)) + \log(P(X_3, X_2|X_3))
\]
Figure 5.1: A hierarchy for modeling interactions between two subjects. \( Y_1 \) and \( Y_2 \) is the data associate with each subject and correspond to latent variables \( X_1 \) and \( X_2 \). These latent variables are controlled by \( X_3 \), which model the interaction between the subject.

Figure 5.2: The hierarchy structure for a single subject (on the left) and the corresponding latent spaces (on the right). Each latent space corresponds to a set of body parts (called partial pose) and is learned for five different motion types: hand waving (red), lifting an object (blue), kicking (green), sitting down (yellow), and punching (cyan).
The first two terms correspond to each subject individually and the third one models the interaction between the subjects.

The latent positions in the hierarchy are learned from training data through maximum likelihood. The optimization algorithm [81] of the hierarchical manifold space contains the following steps:
1. The latent variable of the each leaf node is initialized by applying principal component analysis (PCA) on the corresponding observation dataset.
2. The latent variable of the inner node initialized by applying PCA to the concatenated latent variables of their dependent nodes. This procedure repeats till the root nodes are reached. There is one root node for each activity, and the latent model in each root node is a function of the latent variables of its dependents that belong to this activity.
3. The parameters of the kernel matrices for each Gaussian process model and the latent variable positions are jointly optimized.

To prevent overfitting, fixed dynamical priors (using periodic kernels) are added to the root nodes, and the noise variance of each GP (other than leaf nodes) is fixed.

5.3 Learning

As we described in Section 3.3.4 the human body model $\Gamma$ consists of two statistically independent subspaces $\Gamma = \{\Lambda, \Omega\}$. The first subspace $\Lambda \subseteq \mathbb{R}^6$ describes the body’s 3D location: the rotation and the translation. The second one $\Omega \subseteq \mathbb{R}^{25}$ describes the actual pose, which is represented by the angles between different body parts (see [12] for more details about the human body model). We define a Hierarchical Human Body Model (HHBM) such as that shown in Figure 5.2 for a single person case and in Figure 5.3 for the case with two-subjects involved. We denote the number of the hierarchy layers as $H$.

Let us define $\Omega_{h,l} \subset \Omega$ as the $l$-th subspace in hierarchy level $h$. For instance, on the hierarchy depicted by Figure 5.2, the $\Omega_{3,2}$ stands for the subspace that describes
The hierarchy structure for two-subjects modeling. The hierarchy has 4 layers: the top (root) layer represents both subjects; the second layer describes each subject individually; the third layer describes the groups of body parts of each subject (for example the upper body of subject 1); the last layer describes a single body part for each subject.

For each $\Omega_{h,l}$ the HGPLVM algorithm constructs a latent space $\Theta_{h,l}$ and the mapping function $\varphi^{(h,l)}$ that maps this latent space to the partial pose space $\Omega_{h,l}$.

Let us also define $\theta_{h,l} \in \Theta_{h,l}$ as a coordinate in the $l$-th latent space in the $h$-th hierarchy layer. $\omega_{h,l} \in \Omega_{h,l}$ is the partial pose vector that corresponds to $\theta_{h,l}$. Applying Formula (5.3.1) we have $\omega_{h,l} = \varphi^{(h,l)}(\theta_{h,l})$.

An important property of a hierarchical model is that $\Omega_{h,l}$ is a subset of some $\Omega_{h-1,l}$ in the higher layer of the hierarchy. In other words, for any hierarchy level $h \in 2, 3, ..., H$ and for any subspace $l \in 1, 2, ..., L_h$ in this layer, there always exists a subspace $\tilde{l}$ in the hierarchy level $h-1 \in 1, 2, ..., H-1$, such that $\Omega_{h,l} \subset \Omega_{h-1,\tilde{l}}$ (here $\tilde{l}$ is the index of the parent node in the hierarchy tree).

Additional mappings that are learned by the HGPLVM are the mapping functions between the latent spaces, that correspond to the subspaces, which are connected in
the hierarchy tree $\phi^{(h,l)} : \Theta_{h,l} \mapsto \Theta_{h+1,l}$.

Finally, $\lambda_{h,l,n} \in \Lambda$, $\omega_{h,l,n} \in \Omega_{h,l}$ and $\theta_{h,l,n} \in \Theta_{h,l}$ denote the location, full pose space (full pose) vector and latent coordinates in hierarchy layer $h$ in the latent space $l$ in the frame $n$.

## 5.4 Tracking algorithm

In this section we present a Hierarchical Annealed Particle Filter (HAPF) for 3D body part tracking. A HAPF run is performed at frame $n$ using image-observations $y_n$. These observations can be an image obtained from a single camera or from several cameras. In this section we follow the notations used in [38].

The model configuration data consists of translation and rotation values, latent coordinates and the full pose space vectors:

$$s^{(i)}_{h,l,n} = \{ \lambda^{(i)}_{h,l,n}, \omega^{(i)}_{h,l,n}, \theta^{(i)}_{h,l,n} \}$$

In the frame $n$ and the hierarchy layer $h$ in the latent space $l$ the state of the tracker is represented by a set of weighted particles:

$$S^{\pi}_{h,l,n} = \{ (s^{(0)}_{h,l,n}, \pi^{(0)}_{h,l,n}), \ldots, (s^{(N)}_{h,l,n}, \pi^{(N)}_{h,l,n}) \}$$

where $s^{(i)}_{h,l,n}$ stands for the model configuration and $\pi^{(i)}_{h,l,n}$ corresponds to a particle weight. The un-weighted set of particles is denoted by

$$S_{h,l,n} = \{ s^{(0)}_{h,l,n}, \ldots, s^{(N)}_{h,l,n} \}$$

The tracking algorithm consists of two stages. The first is the generation of new particles in the latent space. In the second stage a corresponding mapping function is applied that transforms latent coordinates to the pose space. After the transformation, the translation and rotation parameters are added and the 31-dimensional vectors are constructed. These vectors represent valid poses, which are projected to
the cameras in order to estimate the likelihood.

Each HAPF run has the following steps:

**Step 1-Initialization.** For every frame the run is started at layer \( h = 1 \) and is initialized by a set of un-weighted particles \( S_{1,l,n} = \{ s_{1,l,n}^{(i)} \}_{i=1}^{N_p} = \{ \lambda_{1,l,n}^{(i)}; \omega_{1,l,n}^{(i)}; \theta_{1,l,n}^{(i)} \}_{i=1}^{N_p} \).

**Step 2.** The weight of each particle is calculated:

\[
\pi_{h,l,n}^{(i)} \propto w^m \left( y_n, s_{h,l,n}^{(i)} \right) = w^m \left( y_n, \lambda_{h,l,n}^{(i)}, \omega_{h,l,n}^{(i)} \right) \tag{5.4.4}
\]

where \( w^m \) is the weighting function suggested by Deutscher and Reid [38]. The weights are normalized so that \( \sum_{i=1}^{N_p} \pi_{h,l,n}^{(i)} = 1 \).

**Step 3.** \( N \) particles are drawn randomly with replacements and with a probability equal to their weight \( \pi_{h,l,n}^{(i)} \). For every latent space \( l \) in the hierarchy level \( h + 1 \) the particle \( s_{h+1,l,n}^{(j)} \) is produced using the \( j^{th} \) chosen particle \( s_{h,l,n}^{(j)} \) (\( i \) is the index of the parent node in the hierarchy tree): \( \lambda_{h+1,l,n}^{(j)} = \lambda_{h,l,n}^{(j)} + B_{\lambda_{h+1}} \) and \( \theta_{h+1,l,n}^{(j)} = \phi(\theta_{h,l,n}^{(j)}) + B_{\theta_{h+1}} \), where \( B_{\lambda_{h}} \) and \( B_{\theta_{h+1}} \) are multivariate Gaussian random variables with covariances \( \Sigma_{\lambda_{h}} \) and \( \Sigma_{\theta_{h+1}} \) correspondingly and mean 0. In order to construct a full pose, the vector \( \omega_{h+1,l,n}^{(j)} \) is initialized with the \( \omega_{h,l,n}^{(j)} \): \( \omega_{h+1,l,n}^{(j)} = \omega_{h,l,n}^{(j)} \) and then updated at the coordinates defined by \( \Omega_{h+1,l} \): \( \omega_{h+1,l,n}^{(j)}[\Omega_{h+1,l}] = s_{h+1,l,n}^{(j)}(\theta_{h+1,l,n}^{(j)}) \)

(The notation \( a[B] \) stands for the coordinates of vector \( a \in A \) defined by the subspace \( B \subseteq A \).) The idea is to use a pose that was estimated using the higher hierarchy layer, with small variations in the coordinates described in the \( \Omega_{h+1,l} \) subspace. Finally, the new particle for the latent space \( l \) in the hierarchy level \( h + 1 \) is

\[
s_{h+1,l,n}^{(j)} = \{ \lambda_{h+1,l,n}^{(j)}; \omega_{h+1,l,n}^{(j)}; \theta_{h+1,l,n}^{(j)} \} \tag{5.4.5}
\]

The sets \( S_{h+1,l,n} \) have now been produced that can be used to initialize the layer \( h + 1 \). The process is repeated until we reach the \( H \)-th layer.

**Step 4.** The \( j^{th} \) chosen particle \( s_{H,l,n}^{(j)} \) in every latent space \( l \) in the lowest hierarchy level is used to produce \( s_{1,l,n+1}^{(j)} \) un-weighted particle set for the next observation:

\[
\lambda_{1,l,n+1}^{(j)} = \frac{1}{L} \sum_{l=1}^{L} \lambda_{H,l,n}^{(j)}
\]

\[
\text{for} 1 \leq l \leq L_H \text{ do } \theta_{1,l,n+1} = \varphi_{1,l}^{-1}(\omega^{(j)})
\]

where \( \omega^{(j)} \) is calculated using

\[
\omega_n^{(j)}[\Omega_{H,l}] = \omega_{H,l,n}[\Omega_{H,l}] \tag{5.4.7}
\]
Here $L_H$ denotes the number of subspaces in the last hierarchy layer $H$.

Finally, $s_{1,1,n+1}^{(j)} = \{\lambda_{1,1,n+1}^{(j)}; \omega_{1,1,n+1}^{(j)}\}$.

**Step 5.** The optimal configuration can be calculated using the following method:

$$
\lambda_{n}^{(opt)} = \frac{1}{L_H} \sum_{l=1}^{L_H} \sum_{j=1}^{N} \lambda_{H,l,n}^{(j)} \pi_{j}^{(j)}
$$

$$
\omega_{n}^{(opt)} = \sum_{j=1}^{N} \omega_{j}^{(j)} \pi_{j}^{(j)}
$$

(5.4.8)

where $\pi_{j}^{(j)} \propto w_{m}^{(y_{n}, \lambda_{n}^{(opt)}, \omega_{j}^{(j)})}$ is the normalized weight of the selected particles so that $\sum_{i=1}^{N_p} \pi_{i}^{(i)} = 1$ and $\omega_{j}^{(j)}$ is calculated as in step 5.

## 5.5 Results

### 5.5.1 Single Person

We tested the HAPF tracking algorithm using the HumanEva, HumanEvaI and HumanEvaII datasets [144] as well as on the test sets created in our lab. We compared the results of the HAPF tracker to the results of Annealed Particle filter implemented by A. Balan et al.[12], and compared the results with the ones produced by the the GPAPF tracker (using GPDM latent space). The error was calculated by comparing the tracker’s output with that of the MoCap system, using the average distance in millimeters between 3-D joint locations, as explained in Section 3.5.

The first sequence shows a person walking in a circle. Fig. 5.4 presents sample images with the actual pose estimation for this sequence. The poses were projected to the first and second cameras. The first 2 rows show the results of the HAPF tracker. The third and fourth rows show the results of the GPAPF tracker. Finally, the fifth and sixth rows show the results of the annealed particle filter. Fig. 5.5 shows the error graphs produced by the HAPF tracker (red stars) GPAPF tracker (based on the GPDM latent space) (blue circles) and by the annealed particle filter (green crosses) for the walking sequence taken at 30 fps. As can be seen, the HAPF tracker produces the most accurate estimation of the body pose. Similar results were achieved for 15 fps. Five annealing layers with 100 particles per layer were used for GPAPF tracker and for Annealed Particle Filter tracker. The same number of particles was used for each layer of the HAPF tracker. In order to more accurately compare GPAPF and
APF, we added a simple dynamic model, assuming constant speed. We calculated average velocities for each part and used these velocities for the propagation between sequential frames. However, because of differences in acceleration during the different stages of the same action, such a simple model was not very helpful. The main reason for the APF failure was that the range of possible part locations significantly increased when the sequence was downsampled from 60 fps (used by Balan et al. [12]) to 30 fps. In order to compensate for this, we had to increase the variance of the Gaussian noise to allow the pose to be meaningfully different from the one observed in the previous frame. In practice, this meant that we had to increase the number of particles. While the GPAPF tracker was able to maintain robust results using the same number of particles, for APF we had to increase both the number of layers and particles per layer. In order to achieve a similar error rate, we had to use (as suggested by Deutscher and Reid [38]) 10 annealing layers with 400 particles in each. This means that APF used 4000 likelihood evaluations (8 times more than GPAPF).

We have also tested HAPF and compared the results with GPAPF on other sequences from HumanEval and HumanEvaII datasets. Fig. 5.6 and Fig. 5.7 present sample frames from the ThorwCatch1 sequence from the HumanEval (S2) set.

Fig. 5.8, Fig. 5.9, and Fig. 5.10 show the results for HumanEvaII(S4) dataset for walking, jogging, and balancing parts respectively; Fig. 5.11 shows the errors of the annealed tracker (red crosses) and the GPAPF tracker (blue circles): (top) HumanEval (S1, walking1, frames 6-590); (middle) HumanEvalII(S2, frames 1-1202) (marked by blue stars); (bottom) HumanEvaII (S4, frames 2-1258) (marked by red circles). The average errors for these sequences are presented in Table 5.5.1. The errors were calculated using the online evaluation system. We can see that the HAPF tracker is very robust and performs much better than GPAPF and APF trackers.

As we mentioned above, HAPF allows dealing with transitions between motions in a much more natural fashion. This is one of the reasons why we see the improvement in tracking results for the datasets involving several different motion types, such as HumanEvaII. Figure 5.12 shows the sample frames from HumanEvaII S1 dataset,
Figure 5.5: The errors of the annealed tracker (green crosses), the GPAPF tracker (blue circles) and HAPF tracker (red stars) for a walking sequence captured at 30 fps.

Figure 5.6: HAPF tracking results. Sample frames from the ThrowCatch1 sequence (throwing part) from the HumanEvaI (S2) dataset.
Figure 5.7: HAPF tracking results. Sample frames from the ThrowCatch1 sequence (throwing part) from the HumanEvaI (S2) dataset.

Figure 5.8: HAPF tracking results. Sample frames from the combo2 sequence (walking example) from the HumanEvaII (S4) dataset.
Figure 5.9: HAPF tracking results. Sample frames from the combo2 sequence (jogging example) from the HumanEvaII (S4) dataset.

Figure 5.10: HAPF tracking results. Sample frames from the combo2 sequence (balancing example) from the HumanEvaII (S4) dataset.
Figure 5.11: The errors of the annealed tracker (green crosses), the GPAPF tracker (blue circles), and the HAPF tracker (red stars): (top) HumanEvaI (S1, walking1, frames 6-590); (middle) HumanEvaII (S2, frames 1-1202); (bottom) HumanEvaII (S4, frames 2-1258).

<table>
<thead>
<tr>
<th>Dataset Sequence</th>
<th>HumanEvaI</th>
<th>HumanEvaII</th>
<th>HumanEvaII</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF</td>
<td>95.4</td>
<td>163.8</td>
<td>172.1</td>
</tr>
<tr>
<td>GPAPF</td>
<td>86.3</td>
<td>86.6</td>
<td>89.0</td>
</tr>
<tr>
<td>HAPF</td>
<td>75.4</td>
<td>75.2</td>
<td>81.8</td>
</tr>
</tbody>
</table>

Table 5.1: The average errors for HumanEvaI(S1, walking1), HumanEvaII(S2) and HumanEvaII(S4) sequences produced by APF, GPAPF and HAPF. The error measures the average distance in millimeters between the joint locations.
which involves three different actions. The subject starts with walking, then continues with jogging and ends with balancing. As it is shown on Figure 5.11 there is no distinguishable peaks on the error graph during the transition, which implies that the tracker is capable of maintaining stable results during the motion type change.

5.5.2 Irregular Motions

The last experiment that we have performed was using the irregular motions dataset, that was created in our lab. This dataset that contains abnormal versions of the regular actions, used for training. For example, walking with raised arm, punching with raised leg and similar. These types of irregular actions were not used during the learning stage of the algorithm. As expected, GPAPF failed to track such actions, as a single latent space does not comprise such poses, and therefore we couldn’t generate suitable particle set. However, HAPF was capable of producing quite good results on these sequences as well. Fig. 5.13, Fig. 5.14, and Fig. 5.15 show the sample frames from 3 different irregular motions: (1) punching with raised leg, (2) waving with both arms and with raised leg, and (3) kicking and raising up the knee with raised arm. As these actions were not used for the learning of the latent spaces, these experiments proof that HAPF is capable of robust tracking new, previously unseen, actions.
5.5.3 Two Persons

We also performed experiments involving two subjects. We created a dataset with two humans performing different actions, such as *handshaking, kicking, hugging, and passing an object*. We used four cameras with a 30 fps sample rate to capture these interactions. In addition, we used the MoCap system to obtain the ground truth location of the body joints of both people in the scene. This data was used for the HGPLVM training and for error calculation. The hierarchy that was used is one showed on Figure 5.3. The error was calculated by comparing the tracker’s output with the ground truth, using the average (per joint) distance in millimeters between the 3D joint locations of both people.

Figure 5.16 demonstrates some sample frames from the hand shaking action, Figure 5.17 demonstrates some sample frames from the kicking action, and Figure 5.18
Figure 5.14: HAPF tracking results for irregular action "waving with both arms and with raised leg".

Figure 5.15: HAPF tracking results for irregular action "kicking and raising up the knee with raised arm".
Figure 5.16: Tracking results of HAPF tracker. Sample frames from the *handshaking* sequence.

Figure 5.17: Tracking results of HAPF tracker. Sample frames from the *kicking* sequence.
Figure 5.18: Tracking results of HAPF tracker. Sample frames from the hugging sequence.

demonstrates sample frames from the hugging action. We compared the results produced by the GPAPF and APF trackers for several motion types involving two humans. As before we used 500 particles per layer for the APF algorithm. The GPAPF algorithm was tested in two different ways. The first (denoted by GPAPF1) was to use a separate latent space for each subject (two latent spaces in all) with five annealing layers and 100 particles for each subject; the second way (denoted by GPAPF2) was to apply a combined latent space describing both subjects (their poses and relative position) with five annealing layers and 200 particles for each. In both cases we used GPDM for learning. Finally, for HAPF we used a hierarchy as shown on Figure 5.3 with 200 particles in each layer.

Table 5.5.3 presents the average error for these algorithms for three different actions: handshaking, kicking, and hugging. Each row represents a different tracking algorithm: the top row shows the results of APF, the second row shows the results of GPAPF with two independent latent spaces, and the third row shows the results
Figure 5.19: The error graphs for *handshaking* (top), *kicking* (middle), and *hugging* (bottom) actions. The errors produced by APF tracker are marked by green crosses, the errors of GPAPF1 tracker are marked by blue circles, the errors of GPAPF2 tracker are marked by magenta squares, and the errors of the HAPF tracker are marked by red stars.
of GPAPF with a joint latent space. The columns represent the motion types. Figure 5.19 shows the error graphs for these actions. Not only is the average error of HAPF significantly lower than that of APF, but the error graphs of GPAPF also have fewer peaks, which emphasizes the relative robustness of the algorithm.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Handshaking</th>
<th>Kicking</th>
<th>Hugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF</td>
<td>142.1</td>
<td>147.3</td>
<td>200.3</td>
</tr>
<tr>
<td>GPAPF1</td>
<td>102.6</td>
<td>115.9</td>
<td>163.6</td>
</tr>
<tr>
<td>GPAPF2</td>
<td>102.4</td>
<td>120.2</td>
<td>152.2</td>
</tr>
<tr>
<td>HAPF</td>
<td>82.1</td>
<td>90.4</td>
<td>102.6</td>
</tr>
</tbody>
</table>

Table 5.2: The average errors of four algorithms: the top row shows the results of APF, the second row shows the result of GPAPF with two independent latent spaces (GPAPF1), the third row shows the result of GPAPF with a joint latent space (GPAPF2), and, finally, the last row shows the results of HAPF tracking algorithm. The columns represent the motion types: handshaking, kicking, and hugging.

5.6 Conclusions

In this paper we have introduced an approach that uses HGPLVM in order to reduce dimensionality and thus improve the ability of the annealed particle filter to track an object even in a high dimensional space. We show that our tracking algorithm is capable of performing simultaneous articulated human body tracking for two subjects. We also show that using HGPLVM makes it possible to recover from temporal target loss. Furthermore, we show that the tracker is capable of robust tracking new, previously unseen motions.

Currently the tracking algorithm cannot adopt to a change of the number of the subjects in the scene. The main reason for this is that the latent space, learned by HGPLVM has a solid model, which requires a prior knowledge of the number of the actors. A possible future research is making an algorithm capable of switching between the models that are used for the tracking.
Chapter 6

Action Classification Using HAPF

6.1 Hierarchical Action Classification

Humans in motion can perform different actions. Our aim is to classify these actions into predefined action types. The classification of actions is based on the sequences of poses detected by the tracker during the performed motion. Suppose there are $K$ different motion types that we attempt to classify. Similar to the GPAPF classification method, each type $k$ is represented by a model $M_k$. For every latent space in every hierarchy layer the model $M_k$ is a sequence of the $l_k + 1$ coordinates in this latent space. We denote a sequence that corresponds to the model $M_k$ on the $l$-th latent space in the hierarchy level $h$ as $M_{h,l,k} = \{\theta_{M_k}^{(h,l)}[0], ..., \theta_{M_k}^{(h,l)}[m_k]\}$. For a frame sequence $Y = \{y_0, ..., y_m\}$ the HAPF tracker generates a sequence of latent coordinates for each latent space. Such a sequence of the coordinates on the $l$-th latent space in the hierarchy level $h$ is denoted as $\Upsilon_{h,l} = \{\theta_{\Upsilon_{(h,l)}}[0], ..., \theta_{\Upsilon_{(h,l)}}[m]\}$.

Using 4.1.3, we can compare the model and the sequence for each latent space. The cumulative distance can be calculated by averaging over the distances for each latent space:

$$d(Y, M_k) = \sum_{h=1}^{H} \sum_{l=1}^{L_h} F(\Upsilon_{h,l}, M_{h,l,k})$$

(6.1.1)

where $H$ is the depth of hierarchy tree and $L_h$ is the number of the latent spaces in the layer $h$ of the hierarchy. The model with the smallest distance is chosen to represent the type of the action.

111
6.2 Hierarchical Action Weighted Classification

The approach, introduced in Section 6.1 produces stable results for the action similar to ones used in the training set. However, many actions produced by people, may differ a lot because of the way each person performs them or because the person can perform multiple actions simultaneously. For example, one can walk with raised arms or perform a kick and a punch simultaneously. In order to allow detection and classification of such actions we need to allow the classification algorithm to make the decision based only on as small group of body parts as possible. For example, it would be very helpful, if we could detect walking only based on the motions of the legs and clapping only based on the movements of the hands. This would allow us to categorize same motion (walking with clapping) to two different actions types (class 1 is walking and class 2 is clapping). However, for many actions such a selection of important body parts is problematic and hard to perform.

Suppose the model for each action contains not only a sequence of the expected latent locations, but also the weight for each latent space. The higher the weight of the latent space the higher is the dependency of the concrete action on this latent space. For example, the latent space, controlling the hands movements is expected to have a low weight, while the latent space controlling left leg should have a high weight. We denote the weight that corresponds to the model $M_k$ on the $l$-th latent space in the hierarchy level $h$ as $w_{h,l,k}$. We normalize the weights, such that:

$$\sum_{h=1}^{H} \sum_{l=1}^{L_h} w_{h,l,k} = 1 \quad (6.2.1)$$

Using these weights we can calculate a distance between model $M_k$, and the sequence produced by the tracker as:

$$d(Y, M_k) = \sum_{h=1}^{H} \sum_{l=1}^{L_h} w_{h,l,k} F(\Upsilon_{h,l}, M_{h,l,k}) \quad (6.2.2)$$

Again, the model with the smallest distance is chosen to represent the type of the
action.

6.3 Results

In this section we provide the results of the classification algorithm and the discussion on the factors that can improve the accuracy rate of the classification. As in the previous section we will provide the results for the classification of the actions performed by a single actor, as well as the interactions between two actors.

Single Person

The classification algorithm was tested on two different datasets. The first set contained 3 different activities: (1) lifting an object, kicking with (2) the left and (3) the right leg. For each activity 5 different sequences were captured. We used one sequence for each motion type in order to construct the action templates. In the second experiment we used five different activities: (1) hand waving, (2) lifting an object, (3) kicking, (4) sitting down, and (5) punching. For each activity five different sequences were captured. A cross-validation procedure was applied: for each motion type one sequence was used in order to construct the latent space and define the model of the motion type and the rest were used for evaluation.

Figure 6.3 shows the trajectories produced by the trackers for the sitting down action projected on different latent spaces from different hierarchy levels. The green line represents the correct model (a model of sitting down action), the red lines represent the incorrect models (models of waving, punching, kicking and picking an object actions), and the colored lines represent the trajectories produced by the tracker. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker. Different types of actions involving hand gestures were difficult to classify because they all look alike. Actions involving sitting down or object lifting were difficult to classify because the high rate of self-occlusions caused it to incorrectly estimate the actual poses.
Figure 6.1: Tracking trajectories of sitting down action projected to different latent spaces: (a) hierarchy level 1, latent space 1; (b) hierarchy level 2, latent space 3; (c) hierarchy level 3, latent space 5. The green line represents the correct model (a model of sitting down action), the red lines represent the incorrect models (models of waving, punching, kicking and picking an object actions), and the colored lines represent the trajectories produced by the tracker. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker.

For the first set we were able to achieve perfect classification results. This is due to clear differences between the models of the different motions in the latent space. In the second experiment the models are less distinguishable, which makes the classification task harder. Table 6.1 shows the results of the classification for the actions from the second dataset. The lower classification rates of actions involving the hand gestures can be explained by the native similarity of the actions. The poor classification rates of sitting down and object lifting actions are due to the high self occlusions, which caused the tracker to produce less accurate results.

Two Persons

The third experiment was to classify the interaction between two subjects. As mentioned above, we created a dataset with two humans performing different actions, such as handshaking, kicking, hugging, pushing, and passing an object, similar to the dataset used by Park et al. [111]. Classifying interactions is a very difficult task because the image of one person often occludes the other, making it hard to use image-based methods. Moreover, some types of motion closely resemble others. As
Table 6.1: The accuracies of the classification, using the hierarchical classification, for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th></th>
<th>Hand waving</th>
<th>Object lifting</th>
<th>Kicking</th>
<th>Sitting down</th>
<th>Punching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand waving</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Object lifting</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Kicking</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sitting down</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Punching</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
</tbody>
</table>

in the previous experiments, cross-validation was applied. We used the GPDM based GPAPF tracker to produce the trajectories in the latent space and tried to perform classification of these interactions. The classification results are presented in Table 6.2.

The results are significantly better than the ones presented by Park et al. [111] and the one based on the GPAPF tracker [126]. There are two reasons for the low success rate for some actions. We can see that the Hugging action was often assigned a wrong motion class, while no other motion was incorrectly classified as Hugging. This is because the tracker was unable to robustly track this action. When two people actually hug one another, the tracker cannot estimate the body part locations correctly. Consequently, the classification was not successful. Punching and pushing were also misclassified due to the limitation of the latent space produced by GPDM. The action templates (trajectories) are very similar, which makes classification difficult.

**Learning Weights**

In order to learn the weights for the hierarchical weighted classification we have created an additional dataset. In this dataset we have different actors performing multiple action simultaneously. For example, walking and raising arms. For each such combined motion we have added the tags, describing the actions on the video. This allowed us to create a dataset of the irregular performances of each motion type.
Table 6.2: Classification accuracy for interactions between two people (using GPDM latent space) for 6 different activities: Hand shaking, Kicking, Hugging, Punching, Pushing, and Object passing. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th></th>
<th>Hand shaking</th>
<th>Kicking</th>
<th>Hugging</th>
<th>Punching</th>
<th>Pushing</th>
<th>Object passing</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand shaking</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>90%</td>
</tr>
<tr>
<td>Kicking</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Hugging</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>75%</td>
</tr>
<tr>
<td>Punching</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>3</td>
<td>1</td>
<td>80%</td>
</tr>
<tr>
<td>Pushing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>17</td>
<td>0</td>
<td>85%</td>
</tr>
<tr>
<td>Object passing</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>90%</td>
</tr>
</tbody>
</table>

The dataset was created using MoCap system, which provided us with a correct 3D location of each joint. These joint locations were mapped to the latent spaces, forming a piece-wise-linear curves in each latent space. For each such curve the Fréchet distance [6] to the model for each tag was calculated. Using the example above (raising arms), a distance to model of walking and to model of raising arms was calculated.

The curves on the latent spaces, describing essential body parts for this motion type (as legs movements for walking) should be very close to the model, while for the latent space that control body parts, that are not important for the concrete motion type (arms for the walking sequence) may contain curves that have a relatively large distance to the model of this type of motion. Fig. 6.2 shows 2 the latent and tracking trajectories of punching action projected to these latent spaces: (a)hierarchy level 3, latent space 1 (representing the left leg); (b) hierarchy level 3, latent space 2 (representing right arm). The green line represents the correct model (a model of punching action), the red lines represent the incorrect models (models of waving, sitting down, kicking and picking an object actions), and the colored lines represent the trajectories produced by the tracker during irregular performance of the punching action. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker. We can clearly see that all the trajectories...
Figure 6.2: The latent and tracking trajectories of punching action projected to these latent spaces: (a) hierarchy level 3, latent space 1 (representing the left leg); (b) hierarchy level 3, latent space 2 (representing right arm). The green line represents the correct model (a model of punching action), the red lines represent the incorrect models (models of waving, sitting down, kicking and picking an object actions), and the colored lines represent the trajectories produced by the tracker during irregular performance of the punching action. The crosses on the colored lines represent the actual latent locations, that were estimated by the HAPF tracker.

Involving the arm movement are very close to the model, suggesting that the entire action is highly dependent on the motion of this body part. The trajectories on the latent space involving the left leg greatly vary, suggesting that the punching action cannot be defined by the leg motion.

Based on this observation, the weights for the leaves of the hierarchy were calculated using the following formula:

\[
\forall l \quad w_{H,l,k} = (\frac{1}{N^k} \sum_{i=1}^{N^k} d(M_{H,l,k}, S_{i,k}(h,l)))^{-1}
\]  

(6.3.1)

Here \(S_{i,k}^{i,k}\) denote the i-th sample of action of type \(k\), \(S_{i,k}^{i,k}(h,l)\) is the sequence of the latent coordinates that corresponds to \(S_{i,k}^{i,k}\) on the l-th latent space in the hierarchy layer \(h\). Finally, \(N^k\) denotes the number of samples of the action of the type \(k\).

For the inner nodes the weights are calculated by averaging over the weights of
Table 6.3: The accuracies of the classification, using the weighted classification, for different irregular activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

<table>
<thead>
<tr>
<th></th>
<th>Hand waving</th>
<th>Object lifting</th>
<th>Kicking</th>
<th>Sitting down</th>
<th>Punching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand waving</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Object lifting</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Kicking</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sitting down</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Punching</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
</tbody>
</table>

where \( N^{h,l} \) denotes the number of child nodes and \( w_{h+1,l^i,k} \) denotes the weight of the \( l^i \)-th child of the \( l \)-th latent space in the hierarchy layer \( h \).

6.3.1 Weighted Classification

We have also performed several experiments attempting to classify the irregular actions from the dataset described in section 6.3. We tried to classify the output of the tracker during the irregular actions. The model that were used are the ones learned for the regular actions (same models that were used in the experiments, described in section 6.3). Table 6.3 shows the result of the classification. We can notice that the results are very close to the results that were achieved during the classification of the regular actions. This is a very important property, implying that the robustness of the classification algorithm does not depend on the minor or significant change in the manner the action is performed. Moreover, it shows that the classification does not depend on the person, who performs the action.
6.4 Conclusions

In this chapter we have presented 2 algorithms for human motion classification using a hierarchy of low dimensional latent spaces. The algorithms uses the trajectories generated in the latent space by the HAPF tracker. These trajectories are classified using Fréchet distance. We demonstrate the algorithm’s robustness in classifying different actions performed by different subjects.

We show that our method can classify interactions between people, which is a much harder task due to the high rate of occlusion. We introduce body parts weights, that allow the classification algorithm to detect irregular actions, that combine several motion types, and classify them correctly.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

Tracking humans, understanding their actions and interpreting them are crucial to a great variety of applications. Analysis of human interactions is a complicated and challenging task for several reasons, such as the large number of body parts, differences in clothing style and illumination, ambiguity caused by body articulation and difficult technical issues, as tracking in occluded environment, image segmentation and feature extraction, which increase the number of aspects that have to be considered in this kind of analysis. Despite the high dimensionality of the problem, many poses can be presented in a low dimensional space by dimensionality reduction. The human actions can be described as curves in this space.

In this thesis we proposed a novel framework for articulated human body pose tracking and action recognition. In this framework we apply different algorithms for a nonlinear dimensionality reduction and perform the tracking and classification in a low dimensional latent space.

The first tracking algorithms that we propose - GPAPF [120, 125, 124, 121] (see Chapter 3) - use the Gaussian Process Latent Variable model (GPLVM)[79] and the Gaussian Process Dynamical Model (GPDM)[167] to perform tracking. These methods generate mappings from a low dimensional latent space to the full data space.
based on learning from previously observed poses from different motion types. Human body model consists of two independent parts: one contains information about 3D location and orientation of the body and the second one describes the pose. We learn latent space that describes poses only. The tracking algorithm consists of two stages. Firstly, the particles are generated in the latent space and are transformed into the data space by using learned a priori mapping function. Secondly, we add rotation and translation parameters to obtain valid poses. The likelihood function calculated in order to evaluate how well a pose matches the visual data. The resulting tracker estimates the locations in the latent space that represents poses with the highest likelihood.

As the latent space is learned from sequences of poses from different motion types, each action can be characterized by a curve in this latent space. The classification of an action can be done based on the comparison of the sequences of latent coordinates, produced by the tracker, and the ones that represent poses sequences of different motion types. In our algorithm [126, 127, 128] we use a modified Frèchet distance in order to compare the sequences of the latent coordinates. This approach allows introducing a different action from the ones we have used for learning by exploiting the curve that represents it.

The second tracking algorithm, presented in this thesis is called Hierarchical Annealed Particle Filter (HAPF) [122, 123] (see Chapter 5). For this algorithm a human body is represented as a hierarchy of body parts. Using the Hierarchical Gaussian Process Latent Variable model (HGPLVM) we construct a latent space for each node of the hierarchy. The improved annealing approach is used for the propagation between different body models and sequential frames. This method is shown to produce a better tracking results. In this thesis we show the ability of this algorithm to robustly track two persons simultaneously. Furthermore, we present the results showing the ability of our tracking algorithm to generalize to the new, previously unseen actions.

We also presented two algorithms for human motion classification using a hierarchy of low dimensional latent spaces [130, 131, 129]. The algorithms uses the trajectories
generated in the latent spaces by the HAPF tracker. These trajectories are classified using Fréchet distance. We demonstrate the algorithm’s robustness in classifying different actions performed by different subjects. We show that the method can classify interactions between people, which is a much harder task due to the high rate of occlusion. We introduce body parts weights, that allow the classification algorithm to detect irregular actions, that combine several motion types, and classify them correctly.

7.2 Future Work

In this section we will discuss the possible future research direction along with the application that can be created in the future, based on the presented methods.

7.2.1 Learning

We have shown an ability to construct a latent space that depicts multiple distinct actions. However, there are several issues that needed to be addressed in the future research. We have found actions that produced a latent space that contains many gaps, which is less suitable for the tracking. Moreover, combining several actions in a single latent space is not a straightforward task. In some cases the resulting space was not suitable for tracking or classifications of the actions. This suggests that the kernel function should be modified to fit our demands. For example we want to obtain distinct curves for each motion that was used for the learning and minimize the gaps between sequential poses for any action. Moreover, we want to try to achieve a latent space, in which similar poses from distinct actions will be close to each other. This will allow us to perform tracking of a person that performs different actions.
7.2.2 Generalization: Tracking New Actions Using the Existing Latent Space

The latent space, that was used for tracking in our experiments for the GPLVM and GPDM based trackers, was learned for a specific action. Combining several types of motion into an unified manifold allowed us to track new poses. However, this ability of the tracker was very limited. Therefore, in this thesis we suggested using hierarchy of the latent space and showed that this approach is capable of tracking irregular actions, which were not used for the leaning. However, the drawback of the suggested approach is that the propagation in time (between different frames) is done only from the lowest hierarchy layer on the current frame to the highest hierarchy layer of the next frame (from leaves to root). The problem of this is that if the root latent space is not capable of comprising the specific pose (which is likely when we are dealing with conceptually new poses) it is not helpful in the sense of the ability to propagate the pose of each body part. For example, if we are tracking a walking sequence with raised left arm then the initial configuration for each frame will be some walking pose with both arms down. The reason for this is that the root latent space can not represent the needed pose reliably. Therefore, we will need to correct the location of the arm in the lower layers. In our algorithm that is done by increasing the number of generated particles in the lower layers.

The alternative would be to propagate between the corresponding latent spaces. However, this would create another problem. If the propagation is done solely between the latent spaces in the low hierarchy layers, such as latent space of the left hand, then we loose all the information that describes the correlation between different body parts (between left hand and right hand for example). This will prevent us from robustly track the body parts and the ability to recover after a temporal target loose.

According to our understanding, in order to use the information of the location of the all body parts and, in the same time, to propagate the information of each body part independently, one need to combine the methods described above. In
other words, in order to generate the initial particle set one should use both the information from the particles in the higher layer of the current frame together with the information from the particles in the same layer from the previous latent space. However, achieving such a mixture is not straightforward and requires an extensive research.

### 7.2.3 Complex Action Detection and Analysis

A very complicated task that has not been addressed in this thesis is detection and recognition of actions from the a video, containing multiple different actions. For example, given a sportsman performing a series of karate elements we need to be able to detect these elements and then classify the whole series. Same goes for a dancing couple. We would like to be able to detect different motion primitives. Therefore, we need to obtain an ability to segment the video and then to perform the classification. This is a very challenging task that requires an extended research and development work to be done. However, being able to perform this is a very important step towards making this algorithm effective for a real life applications, like the ones that are presented in the Section 7.3.

### 7.3 Application

The algorithms, presented in this thesis, have a broad practical use. Many security applications can use it to track human body poses and determine the actions that are performed. However, besides the security issues there is a great variety of other applications that can strongly benefit from using the body tracker and pose estimation algorithm.

**Casino Gesture Detection and Analysis**: It is common knowledge that casinos invest a huge amount of money in order to fight the casino cheaters. According to US laws every game on a casino table has to be video recorded. The tricks that are performed by the players are very complex and hard to detect even for
a human eye. However, a body tracker can generate some logs that describe the events on the table. For example, one can develop a system that will be able to detect occurrence of gestures like "no more bets" and "game end". Such an application has a great practical use.

**Sport Events Detector:** Many sports disciplines, like skating, gymnastics etc., require certain figures and elements to be completed during the performance. We do not believe that using the current state of the art algorithms, it is possible to perform an evaluation of the quality of the elements. We doubt that a reliable verification of the small details, such as checking whether the athlete’s legs were straight while performing a back jump, can be performed. Even harder task is to have the artistic scoring tool. However, it is possible to create a system that will be capable of detecting the basic elements, to verify that all needed components were performed and replay specific ones for the jury, if they need to recheck the performance. Besides the scoring issues, such applications can be used by the TV companies. In many cases they are interested in showing a reply of some complex action or fault. By using our system it will be possible to choose a needed element from the whole performance.
Bibliography


