MOTION MODELS FOR ROBUST 3D HUMAN BODY TRACKING

THÈSE N^{o} 3541 (2006)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédia

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Raquel URTASUN SOTIL

Electrical engineer, Public University of Navarra, Spain
de nationalité Espagnole

Composition du jury:

Prof. P. Fua, directeur de thèse
Prof. David J. Fleet, rapporteur
Prof. Neil Lawrence, rapporteur
Dr. Ronan Boulic, rapporteur
Prof. Roger D. Hersch, président

Lausanne, EPFL
2006

Abstract

In this work, we propose new ways to learn pose and motion priors models and show that they can be used to increase the performance of 3D body tracking algorithms, resulting in very realistic motions under very challenging conditions.

We first explored an approach to 3D people tracking that combines learned motion models and deterministic optimization. The tracking problem is formulated as the minimization of a differentiable criterion whose differential structure is rich enough for optimization to be accomplished via hill-climbing. This avoids the computational expense of Monte Carlo methods, while yielding very good results under challenging conditions. To demonstrate the generality of the approach we show that we can learn and track cyclic motions such as walking and running, as well as acyclic motions such as a golf swing. We also show results from both monocular and multi-camera tracking. Finally, we provide results with a motion model learned from multiple activities, and show how these models can be used for recognition and motion generation. The major limitation of these linear motion models is that they required many noiseless, segmented and time warped examples to create a complete database with good generalization properties.

We therefore investigated more complex non-linear statistical techniques. We advocate the use of Scaled Gaussian Process Latent Variable Models (SGPLVM) to learn prior models of 3D human pose. The SGPLVM simultaneously optimizes a low-dimensional embedding of the high-dimensional pose data and a density function that both gives higher probability to points close to training data and provides a nonlinear probabilistic mapping from the low-dimensional latent space to the full-dimensional pose space. The SGPLVM is a natural choice when only small amounts of training data are available. We demonstrate our approach with two distinct motions, golfing and walking. We show that the SGPLVM sufficiently constrains the problem such that tracking can be accomplished with straightforward deterministic optimization. However, in the presence of very noisy or missing data, for example due to occlusions, the simplistic second order Markov model we use is not realistic enough to sufficiently constrain the algorithm. Moreover, when learning models that contain stylistic diversity, from different people or from the same person performing an activity multiple times, the SGPLVM results in models whose latent trajectories are not smooth, and are therefore not suited for hill climbing tracking.

Finally, we present a more powerfull approach based on the Gaussian Process Dynamical Models (GPDMs) that combines the strengths of the two previous ones. We advocate the use of GPDMs for learning human pose and motion priors. A GPDM provides a low-dimensional embedding of human motion data, with a density function that gives higher probability to poses and motions close to the training data. With Bayesian model averaging a GPDM can be learned from relatively small amounts of data, and it generalizes gracefully to motions outside the training set. Here we modify the GPDM to permit learning from motions with significant stylistic variation. The resulting priors are effective for tracking a range of human walking styles, despite weak and noisy image measurements and significant occlusions.

Index words: Pose models, Motion models, Human Body Tracking

Version Abrégée

Dans cette thèse, nous présentons de nouvelles méthodes d'apprentissage de modèles a priori de poses et de mouvements, et montrons qu'ils peuvent être utilisés dans le but d'améliorer les performances des algorithmes de suivi 3D de corps humain.

Dans un premier temps, nous avons exploré une approche de suivi 3D de personnes qui combine le modèle de mouvement appris à une optimisation déterministe. Le problème de suivi est posé sous la forme d'une minimisation d'un critère différenciable dont la structure différencielle est suffisante pour optimiser selon une stratégie à hypothèse unique. Ceci permet d'éviter l'emploi de coûteuses méthodes de Monte Carlo tout en réalisant de très bons résulats dans des conditions difficiles. Afin de démontrer la généralité de notre méthode, nous montrons que nous pouvons aussi bien apprendre et suivre des mouvements cycliques, comme la marche ou la course, que des mouvements acycliques, comme un swing de golf. De plus, nous présentons des résultats de suivi obtenus soit en monoculaire, soit à l'aide de plusieurs caméras. Finalement, nous présentons des résultats provenant d'un modèle de mouvement appris à partir d'activités mutliples et montrons que ces modèles peuvent être utilisés pour la reconnaissance et la synthèse de mouvements. La principale limitation de ces modèles de mouvements est qu'ils nécessitent un grand nombre d'exemples non-bruités, segmentés et alignés dans le temps afin de créer une base de données complète permettant une généralisation suffisante.

Nous avons donc étudié des techniques statistiques non-linéaires plus complexes. Nous préconisons l'utilisation de *Scaled Gaussian Process Latent Variable Models* (SGPLVMs) pour l'apprentissage de modèles a priori de poses humaines 3D. Le SGPLVM optimise simultanément un plongement en basse dimension des données de poses en haute dimension, ainsi qu'une fonction de densité qui donne une probabilité plus élevée aux points proches des données d'apprentissage et fournit une correspondence probabiliste non-linéaire entre l'espace latent en basse dimension et l'espace des poses en dimension complète. Le SGPLVM est un choix approprié lorsqu'un nombre limité d'exemples d'apprentissage est disponible. Nous démontrons l'efficacité de notre approche à l'aide de deux mouvements distincts, le golf et la marche. Nous montrons que le SGPLVM contraint suffisamment le problème pour que le suivi puisse être obtenu à l'aide d'une optimisation déterministe directe. Malgré tout, en présence de données très bruitées ou manquantes, par exemple lors d'occultations, le modèle simpliste de Markov du deuxième ordre que nous utilisons n'est pas assez réaliste pour contraindre suffisamment l'algorithme. De plus, lors de l'apprentissage de modèles présentant des styles différents d'un mouvement, provenant de différentes personnes ou de la même personne effectuant plusieurs fois la même activité, le SGPLVM produit des modèles dont les trajectoires dans l'espace latent ne sont pas lisses, et donc peu adaptées à du suivi à hypothèse unique.

Finalement, nous présentons une approche plus performante basée sur les *Gaussian Process Dynamical Models* (GPDMs) qui combinent les avantages des deux méthodes précédentes. Nous préconisons l'utilisation des GPDMs pour l'apprentissage de modèles a priori de poses et de mouvements humains. Un GPDM fournit un plongement en basse dimension de données de mouvements humains, ainsi qu'une fonction de densité qui donne une probabilité plus élevée aux poses et aux mouvements proches des données d'apprentissage. En se basant sur le moyennage d'un modèle Bayesien, un GPDM peut être appris à l'aide d'une quantité relativement faible de données et généralise agréablement les mouvements n'appartenant pas à l'ensemble des données d'apprentissage. Dans cette thèse, nous modifions le GPDM afin d'autoriser l'apprentissage de mouvements présentant des variations de style significatives. Les modèles a priori obtenus sont efficaces pour le suivi de marches effectuées avec des styles différents, malgré des données images faibles et bruitées et des occultations significatives.

Mot clefs: Modèles de pose, Modèles de movement, Suivi de corps humain

To my Mother

Acknowledgments

I have been very lucky to work with two great researchers during my phD, Pascal Fua and David J. Fleet. Each of them has teached me a different view of research, "deterministic" vs "probabilistic", showing that there is not a single solution to a problem, but different approaches to it. I have learned a great deal from working with both of them. This work would never have happened without their help.

Pascal's work piqued my interest, convincing me to do a phD in computer vision, even though my idea at that time was to do computer graphics or machine learning. Many thanks for giving me the opportunity to developped my skills during these years, for teaching me all about computer vision, and for spending countless hours with me discussing about this research.

David's conciousness help me improve my research qualities. He changed my research vision by sharing with me his probabilistic view. Thanks for countless hours spent during my trips to Toronto or on endless phone convesations, for giving me back the interest for machine learning, and specially all the brain storming with GPs. Many thanks for the great time spent in Toronto, and for motivating me during the difficult time that is ending a phD.

I would also like to thank the members of my thesis committee for the comprehensive review of my thesis and the constructive discussion during the thesis defense. Thanks to Aaron Hertzmann, Neil Lawrence and Jack Wang for usefull discussions and their work on GPs.

I'm grateful to all the past and present members of CVLab for the great working atmosphere during the past years. In particular, I thank Mathieu, as my officemate, for usefull discussion and endless help, for supporting my caothic side of the office, and specially for his friendship. Special thanks goes to Vincent for all his help and his practical vision of computer vision, and Lorna for all the great time and fun spent "rotating the quaternions". She teached me that one can do research and have fun at the same time. Special thanks goes also to Josiane, for her fast and efficient response to all the administrative problems I had. Many thanks to my fellows at the CVLab: Ali, Andrea, Emilio, Engin, Francois, Jerome, Julien, Miodrag, Mustafa, Pascal and Slobodan for the time spent in the lab.

Special thanks to Etienne, for his friendship, and his help in my personal development and my choices in life. I would like also to thank the past and present members of LIG/VRLab: Amaury, Anderson, Anthony, Christelle, Benoit, Pascal, Rachel, etc.

This adventure started 12 years ago, and since them I have had the support of my keko that I want to specially thank. Many thanks also to txapo, pelutxi, cieguito and nini.

Many thanks go to Marcos and Josepas, for the time spent together and for allways re-

mind me that there is life outside research. Thanks also to Alex, Celine, Historias, Monster for great moments.

I would like also to thank Isabelle Bloch for enforcing my interested in research, and for the great time spent in Paris. Thanks also for all my friends there.

To my basque friends, specially Inigo, for making me feel as the prodigal son. Even being more that one thousand kilometers away, I have the comfort of knowing that they are always close to me.

Special thanks to Petite, "c’est toi", for being you and making me the happiest I have ever been, always taking car of me. Many thanks to Rosita and Fudlito.

Last but not least, I thank my family for their continuous support since the day I started this long trip now twelve years ago. Among all this is dedicated to my Mother.

Contents

1 Glossary of Notation and Acronyms 31

2 Introduction 35
- 2.1 Applications 35
- 2.2 Prior models 36
- 2.3 Contribution of the thesis 37
  - 2.3.1 Pose Models 37
  - 2.3.2 Motion Models 37
  - 2.3.3 Pose Dynamical Models 38
- 2.4 Thesis outline 38

3 State of the art 41
- 3.1 Motion Capture Techniques 41
  - 3.1.1 Mechanical Motion Capture 41
  - 3.1.2 Electro-Magnetic Motion Capture 42
  - 3.1.3 Acoustic Motion Capture 43
  - 3.1.4 Optical Motion Capture 43
  - 3.1.5 Inferring 3D joint orientation and position 44
  - 3.1.6 Video based Motion Capture 45
- 3.2 Generic models: ensure smoothness and joint limits 45
- 3.3 Activity-specific pose models 48
  - 3.3.1 Linear models 48
  - 3.3.2 Mixture of Linear Models 50
  - 3.3.3 Non-parametric models 50
  - 3.3.4 Geometrically-motivated manifold learning 51
  - 3.3.5 Non-linear latent variable models 53
- 3.4 Dynamical models 55
  - 3.4.1 Simple models 55
  - 3.4.2 Non-linear dynamical models 55
  - 3.4.3 Modeling motions 56
  - 3.4.4 Non-linear dynamical latent variable models 56

4 **Motion Models** **59**
- 4.1 Linear Motion Models . . . 59
  - 4.1.1 PCA Motion Model . . . 62
  - 4.1.2 Cyclic motions . . . 62
  - 4.1.3 Golf Swing . . . 65
  - 4.1.4 Motion Clustering . . . 65
  - 4.1.5 Model Convexity . . . 67
- 4.2 Tracking Framework . . . 69
  - 4.2.1 Objective Function . . . 69
  - 4.2.2 Computational Requirements . . . 73
- 4.3 Monocular Tracking . . . 73
  - 4.3.1 Projection Constraints . . . 73
  - 4.3.2 Foreground and Background . . . 74
  - 4.3.3 Point Correspondences (Optical Flow) . . . 74
  - 4.3.4 Experimental Results . . . 75
    - 4.3.4.1 Walking . . . 75
    - 4.3.4.2 Golf Swing . . . 76
- 4.4 Multi-view Tracking . . . 80
  - 4.4.1 Objective Function . . . 80
  - 4.4.2 Experimental Results . . . 81
  - 4.4.3 Recognition . . . 84
- 4.5 Motion Generation . . . 86
  - 4.5.1 Related Work on Motion Generation . . . 87
    - 4.5.1.1 Motion Editing . . . 88
    - 4.5.1.2 Principal Component Analysis for motion synthesis . . . 89
  - 4.5.2 Models for Motion Synthesis and Analysis . . . 89
    - 4.5.2.1 Walking and Running . . . 89
    - 4.5.2.2 Jumping . . . 91
  - 4.5.3 Motion Generation . . . 91
  - 4.5.4 Validation and Results . . . 93
    - 4.5.4.1 Animation Results . . . 93
    - 4.5.4.2 Statistical Validation . . . 95
    - 4.5.4.3 Stylistic extrapolation . . . 97
    - 4.5.4.4 From Video to Animation . . . 98
- 4.6 Conclusion and Future Work . . . 100

5 **Gaussian Process Latent Variable Models for 3D people Tracking** **103**
- 5.1 Gaussian Process Models . . . 105
  - 5.1.1 Gaussian Process Latent Variable Models . . . 105
  - 5.1.2 Human Body parametrization . . . 108
  - 5.1.3 Sparsification and Overfitting . . . 108
  - 5.1.4 Non-smooth latent trajectories . . . 110

5.1.5 Reconstruction Error vs Tracking 115
5.1.6 Learning Specific Motions for Tracking 115
5.2 Monocular Tracking 116
5.2.1 Image Likelihood: 116
5.2.2 Prediction Distribution 117
5.2.3 Initialization and Optimization 118
5.3 Results 119
5.3.1 Walking Motion 119
5.3.2 Golf Swing 119
5.3.3 Failure Modes 122
5.4 Summary and Conclusions 122

6 Gaussian Process Dynamical models 127
6.1 Gaussian Process Dynamical Model 128
6.1.1 Learning 130
6.1.2 Balanced GPDM: 130
6.1.3 Model Results 131
6.1.4 Prior over New Motions 132
6.2 Tracking 135
6.3 Tracking Results 139
6.4 Conclusions and Future Work 141

7 Comparative Results 143
7.1 Learning GP 143
7.1.1 Is parameterization an issue? 143
7.1.2 Generalization of the different GPs 149
7.1.3 Generalization vs amount of training data 154
7.1.4 Dimensionality of the input space 154
7.1.5 Dimensionality of the latent space 162
7.2 Tracking 162
7.2.1 Synthetic data 162
7.2.1.1 GP models 163
7.2.1.2 Motion models 176
7.2.2 Caricatural walking 180
7.2.2.1 GP models 180
7.2.2.2 Motion models 180
7.3 Conclusion 183

8 Conclusions and Future Work 185
8.1 Contributions 185
8.2 Future Work 186

List of Figures

3.1 **Mechanical commercial motion capture systems.** (a) Gypsy system. (b) Physilog system. 42

3.2 **Electro-magnetical motion capture systems.** (a) Liberty mocap system from Polhemus^{TM} [112]. (b) Cabled Flock of Birds system from Ascension^{TM} [7]. (c) Wireless electro-magnetical mocap system Motion Star from Ascension^{TM}. (d) The motion capture for Lara Craft movie was performed by Motion Star. 42

3.3 **Optical motion capture systems.** (a) Capture using a Vicon^{TM} system [155]. (b) Capture performed for the film IRobot using a Motion Analysis^{TM} [100] system. (c) Animal motion capture using Motion Analysis^{TM}. (d) Screen-shots from The Polar Express film that used motion capture obtained from a Vicon^{TM} system. 43

3.4 **Complex primitives for 3D Human Body Tracking.** (a) Superquadrics [133], (b-c) Implicit surfaces for hand [34] and upper body [111] tracking. (d) Texture mesh [135]. 46

3.5 **Modeling dynamic textured scenes.** Realistic texture-mapped human synthesized from new motions, from Starck and Hilton [135]. This method requires a well-engineered capture environment. 46

3.6 Tracking using hierarchical joint limits. (a) Implicit surface representation of the limits for the parent joint. (b) Hierarchical representation. The child joint limits are a function of the rotation of the parent joint. (c) Tracking result projected into the image. (d) 3D tracking result from a view different than the ones used for tracking. 48

4.1 **Monocular Tracking a walking motion** of 43 frames. **First two rows**: The skeleton of the recovered 3D model is projected onto the images. **Bottom two rows**: Volumetric primitives of the recovered 3D model projected into a similar view. 60

4.2 **Monocular Tracking a full swing** in a 45 frame sequence. **First two rows**: The skeleton of the recovered 3–D model is projected into a representative subset of images. **Middle two rows**: Volumetric primitives of the recovered 3–D model projected into the same views. **Bottom two rows**: Volumetric primitives of the 3–D model as seen from above. 61

4.3 **Motion models**. First two PCA components for (a) 4 different captures of 4 subjects walking at speeds varying from 3 to 7km/h, (b) the same subjects at speeds ranging from 6 to 12km/h, (c) the multi-activity database composed of the walking and running motions together. The data corresponding to different subjects is shown in different styles. The solid lines separating clusters have been drawn manually for visualization purposes. (d) Percentage of the database that can be generated with a given number of eigenvectors for the walking (dashed red), running(solid green) and the multi-activity databases(dotted blue). 63

4.4 **Walking motion eigenvectors.** The first one appears in red, green the second, blue the third, cian the fourth and magenta for the last one. Eigenvectors of the flexion-extension in the sagittal plane of the upper leg on the top and flexion-extension in the sagittal plane of the lower leg on the right. 64

4.5 **Individual contribution of the eigenvectors.** The top row shows equispaced poses of the mean walk. The next 5 rows illustrate the influence of the first 5 eigen-motions. The second row shows a linear combination of the mean walk and the first eigen-motion, $\Theta_0 + 0.7\Theta_1$. Similarly, the third row depicts $\Theta_0 + 0.7\Theta_2$ to show the influence of the second eigen-motion, and so on for the remaining 3 rows. 66

4.6 **Key postures** for the golf swing motion capture database that are used to align the training data: Beginning of upswing, end of upswing, ball hit, and end of downswing. The body model is represented as volumetric primitives attached to an articulated skeleton. 67

4.7 **Poor clustering in a pose subspace.** The solid lines that delimited clusters have been manually done for visualization purposes. (a) Projection of training poses onto the first two eigen-directions of the pose subspace. (b) Projection of training poses onto the third and fourth eigen-directions of the pose subspace. While in the motion motion there is strong inter-subject separation, with the pose model in this figure, there is no inter-subject separation. 67

4.8 **Sampling the single activity databases.** In each plot we show the most probable sample that is at the origin, and a sample with very low probability (far from the origin) for the (a) walking, (b) running, and (c) golfing databases. Their respective motions are shown in Fig. 4.9. The dashed curves are one standard deviation ellipses for the underlying Gaussian density model for the data. 68

4.9 **Sampling** the first five components of each single activity database produce physically possible motions. The odd rows show the highest probability sample that for each single-motion database, which is the at the origin $x_i = 0, \forall i$. The even rows show some low probability samples very far from the training motions to demonstrate that even those samples produce realistic motions. The coefficients for these motion are shown in Fig. 4.8 (a,b,c) respectively. **First two rows (a):** Walking, **Middle rows (b):** Running, **Last two rows (c):** Golf swing samples. 70
4.10 **Sampling the multi-activity subspace.** The 4 samples that generate the motions of Fig. 4.11 are depicted. 71
4.11 **Sampling** the first two components of a multi-activity database compose of walking and running motions can produce physically impossible motions. The coefficients of the motions depicted in this figure are shown in Fig. 4.10. **Top row:** Physically impossible motion. The input motion space compose of walking and running is not convex. **Middle row:** Physically possible motion close to a walking. **Bottom row:** Motion close to a running. As the convexity of the input space is assumed when doing PCA, and it may not be the case, the resulting motion as a combination of principal directions can be physically impossible. 72
4.12 **2D Tracking using the WSL tracker. Top row:** Tracking the chest, knees, head, ankles and visible arm. The tracked upper body joints are shown in red, with the head and tracked lower joints points shown in yellow. **Bottom row**: Regions used for tracking the ankles, knees, and head are shown. . . 74
4.13 **Poor quality foreground binary mask. First row:** Extracted from the walking sequence of Fig. 4.1 and **Second row:** from the golf swing of Fig. 4.17. 75
4.14 **Automatic Initialization of the normalized time** parameter μ_t for the walking sequence of Fig. 4.1. (a) Width of the detected silhouette. (b) Spline interpolation for the detected key-postures. 76
4.15 **Detected hand trajectories** for the *full* swing in Fig. 4.2 and the *approach* swing in Fig. 4.18. The left and right hand positions (pixel units) are represented in black and red respectively. 77
4.16 **Assigning normalized times** to the frames of Fig. 4.2. (a) We use the automatically detected club positions to identify the key postures of Fig. 4.6. (b) The corresponding normalized times are denoted by red dots. Spline interpolation is then used to initialize the μ_t parameter for all other frames in the sequence. 78
4.17 **Monocular tracking a full swing** of 68 frames. The skeleton of the recovered 3–D model is projected onto the images. 78
4.18 **Monocular Tracking a short swing** during which the club goes much less high than in a driving swing. The skeleton of the recovered 3–D model is projected onto the images. 79

4.19 **Tracking using only joint constraints vs using the complete objective function**. (a) Original image. (b) 2–D appearance based tracking result. (c) 2–D projection of the tracking results using only joint constraints. The problem is under-constrained and a multiple set of solutions are possible. (d) 3–D tracking results using only joint constraints. (e) and (f) The set of solutions is reduced using correspondences. 79

4.20 **Tracking a running motion**. The legs are now correctly positioned in the whole sequence. 81

4.21 **Input stereo data** for the running sequence of Fig. 4.20. Side views of the 3–D points computed by the Digiclops ^{tm} system. Note that they are very noisy and lack depth because of the low quality of the video sequence. . . . 81

4.22 **Multi-view walking tracking**. Using low resolution stereo data to track the two women whose motions were not used to learn the motion model. The recovered skeleton poses are overlaid in white. 82

4.23 **Tracking a walking motion** from a subject whose motion was not recorded in the database. The legs are correctly positioned. 83

4.24 Comparing recovered rotation angles using visual tracking (solid curve), and by integrating gyroscopic data (smooth curve) for the walk of Fig. 4.23. **Left column**: Right hip and knee sagittal rotations. **Right Column**: Same thing for the left leg. Note that both curves are very close in all plots, even though the left leg is severely occluded. 83

4.25 **Tracking the transition between walking and running.** In the first four frames the subject is running. The transition occurs in the following three frames and the sequence ends with running. The whole sequence is shown. 84

4.26 **Recognition of the walking person from stereo data.** Different subjects appear in different colors and symbols. (a) Clusters formed by the first two PCA coefficients in the database used for tracking, composed of 4 subjects. Black circles represent tracking results for database subjects. The black triangle represent tracking result for a new subject. Note that when the database is conformed of this five subjects it does not cluster anymore in 2–D. (b) Fist two components of a bigger database than the one used for tracking, composed of 9 subjects. Black circles represent tracking results for database subjects and black triangles for a subject whose motions was not recorded when building the database. Note that the database clusters in 4–D. (c) Third and Fourth coefficients of the 9 subjects database. 85

4.27 **The motion database.** (a) Percentage of the database that can be generated with a given number of eigenvalues for the running database. (b) Clustering behavior of the first 3 x_i coefficients of Eq. 4.4 for the 140 motion vectors measured for 5 subjects running at speeds ranging from 6 to 12 km/h. They form relatively compact clusters in 3–D space that can be used for recognition purposes. c) Percentage of the database that can be generated with a given number of eigenvalues for the jumping database. d) Clustering behavior of the first 3 components for 48 motions vectors measured for 4 subjects jumping distances ranging from 40 to 120 cm. They cluster in 3–D. 90

4.28 **Cubic Spline interpolations** of the first two components as a function of the speed for the running database. The synthesized motion depicted in Figure 4.29 is generated from a single optical motion at 6 km/h and is shown in solid black. The original captured motion is depicted in dashed black while the database subjects are shown in different dotted colors. 94

4.29 **Running at speeds increasing from 6 to 12 km/h. Top row:** Superposition of the synthesized motion generated from a *single* optical motion capture at 6 km/h, in yellow, to an animation observed in the actual motion capture data at *all* the relevant speeds, in blue. Note that the two synthesized characters are superposed almost perfectly. **Bottom row:** Superposition of the animations corresponding to *two* different women. Note the big differences compared to the results in the top row. 95

4.30 **Walking at speeds ranging from 3 to 7 km/h. Top row:** Superposition of the motions generated from a *single* example, in yellow, to those observed in the optical motion capture, in blue. They matched very well except from small discrepancies of the arm motion, that can be ascribed to the fact that people do not perform a motion twice exactly in the same fashion. **Bottom row:** Superposition of the animation corresponding to two different subjects. Note the big differences compared to the results in the top row. Note that the motion captured has not been corrected. This is the reason why the left leg is penetrating into the floor. 96

4.31 **Motion generation from a jump of length 80 cm. Top row:** 40 cm jump. **Bottom row:** 120 cm jump. We superpose motions generated from a *single* example (80 cm), in yellow, to those observed by the optical motion capture, in blue. They match very well except for small discrepancies in the arms. . 97

4.32 Superposition of the 120 cm jumping motion generated from a *single* example of length 80 cm, in yellow, to those observed by the optical motion capture, in blue. **Top row:** Frames 1-40. **Bottom row:** Frames 41-80. . . . 98

4.33 **Interpolation Errors**, Intra-variability and Inter-variability values. **Top row:** Running. **Second row:** Walking. **Third row:** Jumping. **Bottom row:** Graphic depiction of the above tables. Note that the interpolation error is consistently larger than the intra-variability but much smaller than the inter-variability in the three databases. 99

4.34 **Generation of stylized motion. Top row:** Original sneaking walking at 4.5 km/h. **Bottom row:** Walking with a sneaking style at 7km/h generated using only one example of such style at 4.5 km/h. Note that the database used is composed only of normal walking styles. 100

4.35 **Walking at speeds from 3 to 7 km/h.** Superposition of the video of Figure 4.22, in blue, on a whole set of optical motion capture examples, in yellow. The leg motion matches almost perfectly, which results in the lower body of the two figures being almost perfectly superposed. Small discrepancies in the arms are due to the intra-variability of the motion, the low resolution of the video, and the fact that the optical motions were performed using a treadmill. Note that the motion capture data has not been cleaned-up, which is why the left leg is penetrating the floor. 101

4.36 Side view of the original and synthesized motion from 3 to 7 km/h for the subject of the video sequence of Figure 4.23. The subject is compared to its closest neighbor in the database according to the Mahalanobis distance. The original motion capture data is represented in blue, and the synthesized one in yellow. They are quite different, which goes to show that our approach can generate a wide range of realistic styles. 102

5.1 Tracking of a 62-frame *short* golf swing. **Top two rows**: The skeleton of the recovered 3D model is projected into a representative subset of images. **Middle two rows**: Volumetric primitives of the recovered 3D model projected into the same views. **Bottom two rows**: Volumetric primitives of the 3D model as seen from above. 104

5.2 **Graphical model**: for the (S)GPLVM. 105

5.3 **Reducing the complexity by subsampling:** without sparsification. The grayscale plot represents $-\frac{D}{2}\ln\sigma^2(\mathbf{x}) - \frac{1}{2}\|\mathbf{x}\|^2$. The blue crosses are the optimized $\mathbf{x}_i$ positions associated with the training poses. (a) SGPLVM using the 139 samples, sampling at 120 Hzs (b) SGPLVM sampling at 60 Hzs. (b) SGPLVM sampling at 40 Hzs. 109

5.4 **Smoothing the prior by decreasing the number of points in the active set:** Learned SGPLVMs with (a) 80, (b) 35, (c) 15 points in the active set. . 110

5.5 **SGPLVMs learned starting from different initial conditions** for the golfing sequence sampled at 30Hzs. (Left) Initial latent coordinates. From top to bottom: Circle, PCA, Isomap, Laplacian Eigenmaps, and LLE. (Right) Learned SGPLVM with active sets and $m = 19$. 111

5.6 **SGPLVM likelihood:** $L(\mathbf{y}_i, \mathbf{x})$ of Eq. 5.11 for a given $\mathbf{y}_i \in \mathbf{Y}$ as a function of x, for the walking sequence, composed of $N = 139$ poses sampled at 120Hz. (a) $L(\mathbf{y}_1, \mathbf{x})$, (b) $L(\mathbf{y}_{N/4}, \mathbf{x})$, (c) $L(\mathbf{y}_{N/2}, \mathbf{x})$, (d) $L(\mathbf{y}_{3N/4}, \mathbf{x})$. The minimun of each plot is depicted as a black dot. The $\mathbf{x}_i$ that are part of the active set are depicted as blue crosses, and the rest of the data as red crosses. White colors depict the most probably positions ($L(\mathbf{y}, \mathbf{x})$ small), and the dark ones the less probable ones (big $L(\mathbf{y}, \mathbf{x})$). Note that there is a very likely flat valley far from the training data, which may result in discontinuities. 113

5.7 **Jumps in the latent space:** can be produce when the initialization contains loops. **Top row:** Init conditions. **Botom row:** latent coordinates learned using a GPLVM without active sets, and the golfing sequence sampled at 60 Hzs. 114

5.8 **GP models:** where the $\mathbf{X}$ are given by PCA, LLE and Laplacian Eigenmaps using the same input data as for the models of Fig. 5.7. 114

5.9 **2D SGPLVM latent spaces used for tracking**. The training data is compose of, **(left)**: 139 frames sampled at 120Hz of a walking cycle performed on a treadmill and retaining 22, **(right)**: 35 frames sampled at 30Hzs of a golf swing and retaining 19 active set points. 115

5.10 **2D Tracking using the WSL tracker. Top row:** Tracking the chest, knees, head, ankles and visible arm. The tracked upper body joints are shown in red, with the head and tracked lower joints points shown in yellow. **Bottom row**: Regions used for tracking the ankles, knees, and head are shown. . . 117

5.11 **Detected club and hand trajectories**: for the *full* swing of Fig. 5.13 and the *short* swing of Fig. 5.1. Left and right hand positions (pixel units) are represented in black and red respectively. Note that the full swing has a much longer trajectory than the other. 120

5.12 **Tracking a walking motion** of 32 frames. **First two rows**: The skeleton of the recovered 3D model is projected onto the images. **Middle two rows**: Volumetric primitives of the recovered 3D model projected into a similar view. **Bottom two rows**: Volumetric primitives of the 3D model as seen from the front view. 120

5.13 **Tracking of a full golf swing** in a 50 frames sequence. **First two rows**: The skeleton of the recovered 3D model is projected into a representative subset of images. **Last two rows**: Volumetric primitives of the 3D model as seen from viewpoint similar to the camera used. 121

5.14 **PCA vs SGPLVM.** Comparison between the PCA-based tracker (chapter 4) in the **first row** and the SGPLVM-based tracker in the **second row**. Note the gain in accuracy due to the generalization ability of the SGPLVMs. . . . 122

5.15 **Estimated latent positions:** when tracking the sequence of Fig. 5.12. Note that the latent positions obtained are far away from the training ones. 123

5.16 **SGPLVM likelihood:** (a) of Eq. 5.11 for a $\mathbf{y}$ that is far from the training data, as a function of the latent position $\mathbf{x}$. It is compose of three terms, (b) $\frac{\|\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))\|^2}{2\sigma^2(\mathbf{x})}$,(c) $\frac{D}{2}\ln\sigma^2(\mathbf{x})$, and (d)$\frac{1}{2}\|\mathbf{x}\|^2$. Lighter colors represent smaller values of $L(\mathbf{x},\mathbf{y})$. The prior in (a) is clearly dominated by (b). . . 124
5.17 **Increasing the amount of training data.** SGPLVMs learned from a database containing 3 walking cycles, each one performed by a different subject. The latent trajectories are non smooth for one of the individuals, resulting in a model not suitable for tracking. 125

6.1 **Graphical model**: for the GPDM and Balanced GPDM. 128
6.2 **Golf Swing:** (a) GPLVM, (b) GPDM and (c) balanced GPDM learned from 9 different golf swings performed by the same subject. (d) Volumetric visualization of reconstruction variance; warmer colors (i.e., red) depict lower variance. 131
6.3 **Walking GPLVM and GPDM:** Learned from 1 gait cycle from each of 6 subjects. Circles and arrows denote latent positions and temporal sequence. (a,b) Side/top views of the 3D latent space for the GPLVM. (c,d) Side/top views of the 3D latent space for the GPDM. (e) Volumetric visualization of the reconstruction variance. Warmer colors denote lower variance. (f) Green trajectories are fair samples from the dynamics model. 133
6.4 **Speed Variation:** 2D models learned for 4 different subjects. Each one walking at 9 speeds ranging from 3 to 7 km/h. Red points are latent positions of training poses. Intensity is proportional to $-2\ln\sigma_{\mathbf{y}|\mathbf{x},\mathbf{X},\mathbf{Y},\bar{\beta}}$, so brighter regions have smaller pose reconstruction variance. **Top row:** healthy subjects. **Bottom row:** subjects with a knee and a hip pathology respectively, that walks asymmetrically. 134
6.5 **WSL Tracks:** The 2D tracked regions for the different tracked sequences (in yellow) are noisy and sometimes missing. 136
6.6 Tracking 63 frames of a walking, with noisy and missing data. The skeleton of the recovered 3D model is projected onto the images. The WSL tracks are shown in red. Note the accuracy of the result. 138
6.7 Tracking 56 frames of a walking motion with an almost total occlusion (just the head is visible) in a very clutter and moving background. Note that the prediction when the occlusion happens is a realistic motion. 139
6.8 **Tracked Latent Positions:** Side and top views of the 3D latent space show the latent trajectories for the tracking results of Figs. 6.6, 6.7, 6.9, and 6.10 are shown in red, blue, black, and green. The learned model latent positions are cyan. 140
6.9 Tracking 72 frames of a walking motion with a total occlusion. During the occlusion the tracker is switch off and the mean prediction is used. Note the accuracy of the tracking before and after the occlusion and the plausible motion during it. 141

6.10 Tracking 37 frames of an exaggerated gait. Note that the results are very accurate even though the style is very different from any of the training motions. The last two rows depict two different views of the 3D inferred poses of the second row. 142

7.1 **GPLVM and GPDM for different parametrizations** seen from the same viewpoint. They were learned from a database compose of 9 walking cycles of 9 different persons (6 male and 3 female) walking at 4 km/h in a treadmill. The latent positions are initialized using PCA (first column). Due to the high dimensionality of the data, the learned GPLVM (second column) and GPDM (last column) models have non smooth latent trajectories. Because the input space is very high dimensional (80D), the reconstruction term of GPDM is overfitting, the dynamical one has almost no effect. This results in similar models to the GPLVMs. 144

7.2 **Balanced GPDM for different parameterizations** learned from the 9 people database, seen from two viewpoints. The latent coordinates were initialized with the ones depicted by the left column of Fig. 7.1. Note the clustering corresponding to the positional parametrization of the top row. The exponential map and the quaternion models are very similar. The Euler angle is the simplest parametrization, but also the one with the largest artifacts. The cos-sin is the sparsest one since it has double the degrees of freedom of the exponential map or Euler-angle parameterizations. 145

7.3 **Error histograms for the Balanced GPDM, GPDM and GPLVM** of Figs. 7.1 and 7.2. The errors, defined as $||\mathbf{y} - \mu_Y(\mathbf{x}))||$, are depicted in blue for the training data and in green for the test data. The test sequences are walkings performed by the same 9 subjects as before, but at a different speed (5km/h) than the training ones (4km/h). Note that the B-GPDM generalized better; The error histograms for the test data are narrower than the ones of GPDM and GPLVM. 146

7.4 **Mean Errors** for the 9 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponential map, Euler angle and cos-sin parameterizations. For each parametrization, three error bars are depicted, one for each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** Training data. **Right:** Test sequences are walkings performed by the same 9 subjects, but at a different speed (5km/h) than the training ones (4km/h). Note that the GPDM and GPLVM are overfitting (bigger errors for testing), and the training errors of the B-GPDM are much bigger since the dynamics is minimized assuming a narrower prior than the GPDM. The mean errors for training and testing are very small. 147

7.5 **GPLVM and GPDM for different parametrizations** seen from the same viewpoint. PCA was used for initialization (left). As for the 9 people walking database of Fig. 7.1, due to the high dimensionality of the data, the learned GPLVM (center) and GPDM (right) models have non smooth latent trajectories, resulting in models less suitable for tracking than the B-GPDM of Fig. 7.6 150

7.6 **Balanced GPDM for different parameterizations**, learned from the 6 people database, view from two different viewpoints. The PCA coordinates depicted by Fig. 7.5 (left) were used as initialization. As for the 9 people database, the positional parametrization is the most clustered, the exponential map model is similar to the quaternion one and the cos-sin yields the most sparse model. Note that the artifacts of the Euler angle parametrization are less visible here, the cycles are well represented as ellipses within a common cylindrical structure. The content is represented as an ellipse an the style (subject identity) as the length axis of the cylinder. These models are more suitable for tracking since poses performed at the same phase of the motion are aligned. 151

7.7 **Error histograms** for the Balanced GPDM, GPDM and GPLVM of Figs. 7.5 and 7.6. The errors are depicted in blue for the training data, in green for test data performed by the same subject at different speeds ranging from 3 to 7 km/h, and in red for test data performed by 3 different subjects (whose motions are not part of the database) at different speed ranging from 3 to 7 km/h. Note that the B-GPDM generalized better, the error histograms for the test data are narrower than the ones of GPDM and GPLVM. 152

7.8 **Mean Errors** for the 6 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponential map, Euler angle and cos-sin parameterizations. For each of parametrization, three error bars are depicted, one of each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** training data, **Center:** walking cycles performed by the same subjects but with different speeds ranging from 3 to 7km/h. **Right:** walking cycles of 3 subjects whose motions are not part of the database, walking with speeds ranging 3 to 7km/h. Note that as for the 9 people database of Fig. 7.8 the GPDM and GPLVM are overfitting, and the B-GPDM has the worst training errors. The mean errors for training and testing are very small. 153

7.9 **GPLVM and GPDM for different parametrizations**, learned from a database composed of 2 subjects (1 male, 1 female) walking on a treadmill at 4km/h. Because of the small amount of training data, the GPLVM (center) and (GPDM) models have smooth latent trajectories. 155

7.10 **B-GPDM for different parameterizations for** the 2 subjects database, view from two different viewpoints. Note that the learned models vary a lot depending on the type of parametrization used. The positional one yields once more the most clustered model, and the exponential map model is quite similar to the quaternion one. Because the PCA initialization (Fig. 7.9) has a small loop, the B-GPDM pull apart the two walkings. This model could be useful for recognition, since the poses from different subjects are clearly separated. 156

7.11 **Error histograms** for the Balanced GPDM, GPDM and GPLVM of Figs. 7.9 and 7.10. The errors are depicted in blue for the training data, in green for test data performed by the same subject at different speeds ranging from 3 to 7km/h, and in red for test data performed by 3 different subjects (whose motions are not part of the database) at 4km/h. Due to the small amount of training data, the errors of the B-GPDM, GPDM and GPLVM are similar. . 157

7.12 **Mean Errors** for the 2 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponetial map, Euler angle and cos-sin parameterizations. For each parametrization, three error bars are depicted, one of each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** training data, **Center:** walking cycles performed by the same subjects but with different speeds ranging from 3 to 7km/h. **Right:** walking cycles of 3 subjects whose motions are not part of the database, walking at a speed of 4km/h. Note that all the models are overfitting and have similar reconstruction errors for the test sequence. 158

7.13 **GP Models for a 20D input space**. The training data is composed of 6 subjects walking at 4km/h on a treadmill. Note that the models clustered, and the poses that correspond to the same phase of the motion for the 6 subjects are aligned. 159

7.14 **2D Balanced GPDM and GPDM for different parametrization**. When the initialization contains loops the B-GPDM and GPDM result in non smooth latent trajectories. The jumps in the latent space are depicted in red. To avoid these discontinuities, one should change the initial conditions, raise the dynamics likelihood prior to a much stronger power than D/d, include back-constraints, model suitable priors for $\mathbf{X}$, or increase the dimensionality of the latent space. The 3D models are depicted in Figs. 7.9 and 7.10. 160

7.15 **2D Balanced GPDM and GPDM for different parameterizations**. When the initialization contains loops the B-GPDM and GPDM result in non smooth latent trajectories. The jumps in the latent space are depicted in red. None of the models result in smooth latent trajectories. The dimensionality used is not high enough and one should learn 3D models (Fig. 7.6). 161

7.16 **Mean errors when tracking a training sequence with GPs**. Each plot is composed of 3 groups, representing Balanced GPDM, GPDM and GPLVM results. For each GP type, 8 bars are shown in 4 different colors. Each color corresponds to a different model: 80D (9, 6, and 2 subjects) and 20D (6 subjects) respectively. For each model, two bars are depicted in the same color, the first one represents the error when using all the 2D projection as input, the second one uses only a subset, the same that was used for real sequences. Three types of errors are depicted: The 2D projections errors (pixels), the error in the 3D location of the joints (mm), and the error in the estimated Euler angles (radians). They are not always correlated, as shown in the first row for the 20D model (blue). 164

7.17 **Estimated latent trajectories for the 9 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Four trajectories are depicted for each plot. Minimizing Eq. 6.26 whole set (red) and subset (green), when MP and complete set (black) and MP and subset (blue). The latent trajectories are not well recovered for the Balanced GPDM, and non smooth for the GPLVM. . . . 165

7.18 **Estimated latent trajectories for the 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories estimated when tracking with the Balanced GPDM and non assuming perfect reconstruction are not well recovered, resulting in big errors, as depicted by Fig. 7.16. 166

7.19 **Estimated latent trajectories for the 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories are well recovered, resulting in small errors, as depicted by Fig. 7.16. 167

7.20 **Estimated latent trajectories for the 20D 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories are well recovered, resulting in relatively small errors, as depicted by Fig. 7.16. . . 168

7.21 **Estimated latent trajectories when reducing σ_e. Top row** represents the results when using B-GPDM and the 9 people database, with a value of $\sigma_e = 1$ for the training sequence of Fig. 7.18. **Bottom row** represents the latent trajectories when using B-GPDM and the 6 people database, with a value of $\sigma_e = 1$ for the test sequence. The corresponding $\sigma_e = 10$ tracked values are shown in Fig. 7.23. 170

7.22 **Mean errors when tracking with GPs a test sequence** performed by a subjects whose motion is not part of the 6 and 9 people database, performed at 3km/h, while the training motions are performed at 4km/h. Each plot is composed of 3 groups, representing Balanced GPDM, GPDM and GPLVM results. For each GP type, 8 bars are shown in 4 different colors. Each color corresponds to a different model: 80D (9, 6, and 2 subjects) and 20D (6 subjects) respectively. For each model, two bars are depicted in the same color, the first one represents the error when using all the 2D projection as input, the second one uses only a subset, the same that was used for real sequences. Three types of errors are depicted: The 2D projections errors (pixels), the error in the 3D location of the joints (mm), and the error in the estimated Euler angles (radians). 171

7.23 **Estimated latent trajectories for the 9 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the Balanced GPDM are the only ones that are smooth, resulting in large tracking errors for the GPDM and GPLVM when assuming perfect reconstruction (Fig. 7.22), since the pose is directly computed from the $\mathbf{x}$ estimated as the mean reconstruction prediction. 172

7.24 **Estimated latent trajectories for the 6 people database** when tracking the test sequence. The training data is shown in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the GPLVM latent trajectories are non smooth. 173

7.25 **Estimated latent trajectories for the 2 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the GPLVM are not smooth. 174

7.26 **Estimated latent trajectories for the 20D 6 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the Balanced GPDM have diverged, but that the tracking errors (Fig. 7.22) are not big (in terms of 2D projection, and 3D position). The tracking is doing the correct thing, but the system does not have the freedom to evolve, resulting in large tracking errors. 175

7.27 **Mean errors when tracking with the PCA motion models.** Three types of errors (2D projection, 3D location, Euler angles) are depicted. Each plot is split in two groups, the left one represents the training sequence errors and the right one the test sequence ones. For each group 4 error bars of 2 different colors are depicted, each color represents a different window size ($\tau + 1 = 3$ on red and $\tau + 1 = 5$ on green). For each color two bars show the errors first for the complete set of joints and then for the subset of joints, with similar results. The errors for the sequence performed by a subject whose motion was not recorded when building the database are much bigger than the ones for the training sequence. 176

7.28 **PCA motion models** tracking results when tracking the training sequence. The tracking errors are depicted in the left side of each plot of Fig. 7.27. **Top row:** Latent coordinates recovered when tracking a training sequence. Note that the trajectories are less smooth than the ones obtained with the GP, and that they are in the neighborhood of the training data of the same subject, so that the tracker recognizes the subject. **Bottom row:** Estimated phase of the motion as a function of the frame index. Note that the phase is smoothly monotonically increasing. 177

7.29 **PCA motion models** tracking results when tracking the test sequence that was performed at 3km/h by a subject whose motions were not used for learning. The tracking errors are depicted in the right side of each plot of Fig. 7.27. **Top row:** Latent coordinates recovered when tracking a training sequence. Note that the trajectories are less smooth than the ones obtained with the GP, and that they do not cluster in any of the subjects. The tracker does not recognize the subject. **Bottom row:** Estimated phase of the motion as a function of the frame index. Note that the phase is less smooth that for the training sequence depicted in Fig. 7.28. 178

7.30 **Error comparison (mm).** Table representing the mean, minimal and maximal errors in the 3D joint estimation for the different 80D models tested in the training and test sequences. The second row represent the mean, minimal and maximal errors in the same order as the table. Each plot is divided in two, the training and testing results. The errors are expressed in mm. For the GPs models the errors are a mean, (minimal, maximal) of the 9, 6 and 2 people database with all the joints and with a subset. For the PCA, the results are obtained with $\tau = 3$ and $\tau = 5$, for all the joints and a subset. Note that the B-GPDM is the one with the best results when $\sigma_e = 1$. It generalizes much better to the test sequence. 179

7.31 Tracking 40 frames of an exaggerated gait with the different GP and PCA models. Note that the 20D B-GPDM is the one that performed the best. The PCA results in impossible positions. The GPLVM and GPDM trackers are lost suddenly at the end of the sequence because of divergence. The 9 subject database too, since the latent trajectories are not aligned by phase of the motion, and 80D are too much to estimate for a sequence that is so far from the training ones. The 3D representation is depicted in Fig. 7.32. . 181

7.32 Tracking 40 frames of an exaggerated gait with the different GP and PCA models. The 3D visualization is depicted by Fig. 7.32. 182

1 Glossary of Notation and Acronyms

We include here the notation used in this work. In those few cases where a symbol has more than one meaning, the context (or a specific statement) resolves the ambiguity. We use bold letters for vectors and capital letters for matrices.

$\mathbf{y}$	Input variable, either pose or motion
$\mathbf{x}$	Latent variable. Low dimensional representation of $\mathbf{y}$
x_i	I-th dimension of variable $\mathbf{x}$
$\mathbf{y}_i$	Input training point
$\mathbf{x}_i$	Latent coordinate associate with training point $\mathbf{y}_i$
$\mathbf{Y}$	Training set $\mathbf{Y} \equiv [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T$
$\mathbf{X}$	Latent coordinate set $\mathbf{X} \equiv [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T$
$\|\mathbf{X}\|$	Cardinality of set $\mathbf{X}$
N	Number of training points, $\|\mathbf{X}\| = \|\mathbf{Y}\| = N$
D	Observation dimension, $\mathbf{y}_i \in \mathbf{R}^D$
d	Latent dimension, $\mathbf{x}_i \in \mathbf{R}^d$
$\mathcal{M}$	Manifold
$p(\mathbf{x})$	Probability of $\mathbf{x}$
$p(\mathbf{y}\|\mathbf{x})$	Conditional probability of $\mathbf{y}$ given $\mathbf{x}$
$p(\mathbf{x}, \mathbf{y})$	Joint probability of variables $\mathbf{x}$ and $\mathbf{y}$
$\mathcal{N}(\mathbf{x}\|\mu, \Sigma)$	$\mathbf{x}$ is a random variable normally distribed, with mean μ, and covariance Σ.
$\mathbf{I}$	Identity matrix
$\mathbf{x}_{1:t}$	Set of $\{\mathbf{x}_1, \cdots, \mathbf{x}_t\}$
$\mathbf{n}$	Noise random variable
g	Mapping from the latent space to the observation space, $g : \mathbf{x} \rightarrow \mathbf{y}$
g^{-1}	Inverse mapping (from the observation space to the latent space) $g^{-1} : \mathbf{y} \rightarrow \mathbf{x}$
∇	Laplacian
δ	Displacement infinitesimally small
f	Mapping that models the dynamics
B	Parameters of the reconstruction mapping
A	Parameters of the dynamics mapping
$\varphi(\mathbf{x})$	Kernel that models the mapping from the latent space to the observation space
$\chi(\mathbf{x})$	Kernel that models the dynamics
Ψ	Motion vector

$\mathbf{z}_t$	Global translation
$\mathbf{o}_t$	Global rotation
$\mathbf{g}$	Global motion, $\mathbf{g}_t = (\mathbf{z}_t, \mathbf{o}_t)$
σ	Variance
μ_t	Normalize time or phase of the motion
Q	Total variance of the training data capture by the subspace
λ	Eigenvalue
Θ_0	Mean motion
Θ_i	I-th Eigenvector, when each example is compose of a motion
ϵ_i	Residual error of the i-th constraint
n_{obs}	Number of observations
h_i	Primitive field function
h^b	Complete field function for body part b
B	Number of body parts
SK	Skin level set
$\mathbf{K_Y}$	Kernel reconstruction matrix of a GP
$\mathbf{K_X}$	Kernel dynamics matrix of a GPDM
$k(\mathbf{x}, \mathbf{x}')$	Kernel function
M	GP model
ϕ	State variable
t	time
$\mathbf{I}_t$	Image at time t
T	Number of frames in the video sequence
$\tau + 1$	Size of the temporal window

We use the following abreviations.

$Mocap$	motion capture
KF	Kalmann filtering
EKF	Extended Kalmann filtering
UKF	Unscented Kalmann filtering
AR	Auto-regressive model
LDS	Linear dynamical system
PCA	Principal Component Analysis
$PPCA$	Probabilistic Principal Component Analysis
FA	Factor Analysis
ICA	Independent Component Analysis
CCA	Canonical Correlation Analysis
MAF	Maximum Autocorrelation Factors
GMM	Gaussian Mixture models
EM	Expectation Maximization algorithm
GMR	Gaussian Mixture Regression
$NLDR$	Nonlinear dimensionality reduction
LLE	Locally Linear Embedding
MDS	Multidimensional Scaling
MFA	Mixture of Factor Analyzers
$LWPR$	Locally Weighted Projection Regression
NN	Neural Networks
RBF	Radial Basis Functions
RVM	Relevance Vector Machines
$NLVM$	Nonlinear latent variable model
GTM	Generative Topographic Mapping
GP	Gaussian Process
$GPLVM$	Gaussian Process Latent Variable Model
$SGPLVM$	Scale Gaussian Process Latent Variable Model
$GPDM$	Gaussian Proces Dynamical Model
$B-GPDM$	Balanced Gaussian Proces Dynamical Model
MAP	Maximum a posteriory estimate

2 Introduction

The 3D estimation of human pose from video is often poorly constrained, owing to reflection ambiguities, occlusions, cluttered backgrounds, non-rigidity of tissue and clothing, image blur, complex and rapid motions and poor image resolution. This can be addressed by using several synchronized cameras, engineering the environment to facilitate the extraction of features such as silhouettes, or introducing prior models to constrain the tracking and disambiguate difficult situations.

In this work, we investigate the latter approach with a view to deriving meaningful 3D information from single video sequences acquired without having to modify the environment. Given the ubiquity of cameras ranging from cheap webcams to sophisticated movie cameras, we believe that this approach has the potential to make video-based motion capture a truly practical approach in a wide range of applications.

In the remainder of this chapter, we first describe some of these applications and outline what we mean by *prior models* in this context. We then summarize our contributions and provide an outline of this thesis.

2.1 Applications

Human Motion capture can be defined as digitally recording the movements of humans, or creating a 3D representation of a live performance [93]. It is of broad interest for a very wide range of applications, such as clinical studies, movies, computer games, sports, or surveillance. Video-based motion capture is the most attractive approach since it is both cheap and non invasive. It does not require any special hardware, since ordinary cameras can be used to provide input data.

Gait analysis, orthopedics and neurology studies are representative examples of medical applications. Nowadays almost all hospitals have a motion capture system, usually an optical or electro magnetic one. Gait analysis is useful to measure the degree of change (range of motion, shaking) in conditions such as arthritis, Parkinson or strokes. It is also used to evaluate gain in performance after operations, evaluate different prothesis, and so on. The use of gait signature for recognition, monitoring of elder people or people suffering diseases like dementia, the detection of atypical motions to prevent unsafety situations, are examples of typical Surveillance applications.

Athletic coaching provides another application domain, which involves capturing and analyzing the motion of athletes. Comparison with other motions helps prevent and detect mistakes that may degrade performance. Golfers, tennis players and skiers are representative of the athletes that can benefit from such techniques.

However, entertainment is by far the biggest application field. Special effects for films such as the Lord of the Rings or Matrix, and particularly 3D computer animated films such as Nemo, The Incredibles, Shreck, Final Fantasy, or the Polar Express, animate their characters by using motion capture data. The artists adapt the capture motions to the particular shape and characteristics of the characters, for example to caricature the motions. Computer games also represent a big portion of the motion capture market. Automatic comments from video or automatic video indexing are some of the goals of the TV market.

2.2 Prior models

Unfortunately, because of the high-dimensional parameterization of human models, learning or manually designing prior models is difficult.

The simplest prior models are those that enforce smoothness across time, typically by introducing priors derived using first or second order Markov models. Because human motion can involve abrupt accelerations and orientation changes, the resulting models are neither particularly realistic nor effective at constraining a video-based reconstruction algorithm.

These models can be augmented by introducing joint limits that prevent physiologically impossible postures. While this usually helps, it does not remove the ambiguities inherent to monocular human motion tracking. They usually involve several anatomically possible poses whose projections produce roughly the same 2D outlines.

In this work, we investigate the use of statistical learning techniques to automate this task and learn effective prior models from motion-capture data. Although this is in principle more restrictive, it is in most of the cases necessary to resolve ambiguities. A pose can be defined as a combination of joint angles or spatial joint locations. Similarly, a motion can be described as a combination of poses, and the human spatial behavior as combination of motions. Moreover, nothing prevents us from having different models working in parallel, each one modeling a restricted set of activities. For example, this could be achieved by choosing the one that performs better. We therefore distinguish three kinds of models.

- **Pose models.** The priors are defined over human body configurations. We will show that they can be effectively learned using non-linear statistical techniques using relatively small amounts of training data. They are used in combination with dynamical models, usually simple first and second order Markov models, which model a prior over consecutive poses.

- **Motion models.** The priors are defined over a sequence of poses across several temporal frames. Surprisingly, these models can be learned using simpler linear techniques than the ones necessary to model poses, but at the cost of requiring larger amounts of training data.

- **Pose dynamical models.** The priors are defined both over human body configurations, and over consecutive poses. We will show that they can be learned, within a common framework in a single step, using non-linear statistical techniques from small amounts of training data.

Pose models can be learned from a database of poses without temporal information, and from small amounts of data. Motion models contain a dynamical model and are easier to learn, but they require relatively large amounts of noiseless segmented and time warped training data. Pose dynamical models combine the advantages of both methods. They contain a dynamical model. They can model different activities at the same time and be learned from small amounts of training data. Finally they do not require noiseless segmented and time warped training data.

2.3 Contribution of the thesis

In this work, we explore new ways to learn pose, motion and pose dynamical models, and to impose them as priors for 3D body tracking from ordinary monocular video. We show that if we use motion models, then simple linear techniques can be used to achieve good tracking results and perform activity and subject recognition, and stylistic motion generation.

However, they require large amount of noiseless segmented and time warped training data. We have therefore investigated more complex non-linear statistical techniques. We showed that Gaussian Process can be used to learn pose models from much smaller amounts of training data than competitive techniques, while generalizing well to motions that are very different from the training ones. This yields very robust in the presence of occlusions and noisy data, resulting in very realistic motions.

2.3.1 Pose Models

We exploit the recently developed Scaled Gaussian Process Latent Variable Model (SGPLVM) [54, 78] to learn a low-dimensional embedding of high-dimensional human pose data. The SGPLVM provides a continuous, kernel-based density function over positions in a low-dimensional latent space and positions in the full pose space that is generally non-Gaussian and multimodal. Importantly, it provides a natural preference for poses close to the training data, smoothly falling off with distance. The model also provides a simple, nonlinear, probabilistic mapping from the latent space to the full pose space. Its variance reflects the uncertainty of the mapping.

We show that the model can be learned from much smaller amounts of training data than competing techniques such as [41, 130]. Learning can be achieved using as little as one exemplar of each motion and involves very few manual parameter tuning , while resulting in very realistic motions.

2.3.2 Motion Models

While complex non-linear methods are required to learn pose models, we demonstrate that one can use simple algorithms such as PCA to learn effective motion models. This may seem surprising as one would expect the learning of a lower dimensional space (pose) to be less complex than the learning of a higher dimensional one (motion), which contains the smaller one. But learning clean segmented and time warped motion databases is equivalent to learning pose models with the phase of the motion as hidden state. This means that, using motion models a pose is obtained by combining poses performed at similar phase times, resulting in more convex spaces.

With such motion models we can formulate and solve the tracking problem in terms of continuous objective functions whose differential structure is rich enough to take advantage of standard optimization methods, resulting in much faster methods.

Moreover, we show that these subspace motion models are much more discriminative than pose models and that one can perform motion-based recognition of individuals and activities.

We present a new method for motion extrapolation, where these motion models are used to infer new motions of a subject that is observed once performing a given activity, while respecting his/her particular style.

2.3.3 Pose Dynamical Models

Simple first order and second order Markov models are not good enough to track while dealing with very few or noisy image features with outliers, for example in the presence of severe occlusions. Moreover, smooth SGPLVMs that contain stylistic diversity cannot be learned.

Here, we propose a new form of Gaussian Process Dynamical Models (GPDM) [159], called Balanced GPDM, to model pose and complex dynamics with stylistic diversity. The GPDM is a latent variable model with a nonlinear probabilistic mapping from latent positions to human poses, and a nonlinear dynamical mapping on the latent space. It provides a continuous density function over poses and motions that is generally non-Gaussian and multimodal. Given training sequences, one simultaneously learns the latent embedding, the latent dynamics, and the pose reconstruction mapping. We show that these models can be learned from smaller amount of training data than competitive techniques, while producing realistic motions and generalizing well for motions very different from the training ones.

2.4 Thesis outline

Chapter 3 discuss related approaches to 3D people tracking. Chapter 4 deals with our linear motion models, showing their properties and limitations and their application to tracking, activity and subject recognition and motion generation. The motion models are shown first since the pose models and pose dynamical models we explore are related. Chapter 5 handles GPLVMs to model pose priors, showing their properties, problems and their application to tracking. Chapter 6 the novel pose dynamical models, showing how to modify the GPDM for learning multiple sequences, and how to use these models for tracking. We will show how these models solve some of the problems of the GPLVM. The different issues when learning Gaussian Process human pose models for tracking are discussed in Chapter 7. This chapter also includes comparative tracking results for the different pose and motion models for synthetic and real data. Finally Chapter 8 gives our conclusions and presents possible future extensions.

3 State of the art

Our goal is to perform motion capture from a single camera. This is the cheapest, non-invasive technique that can be performed in an uncontrolled environment, resulting in a truly practical approach in a wide range of applications. The most important ones are the film industry, computer games, surveillance and medical applications. Unfortunately, it is also by far the most difficult approach, and remains essentially unsolved. In this chapter we present an overview of different motion capture techniques. First, we will discuss briefly existing commercial systems, followed by a more detailed survey on video-based motion capture approaches.

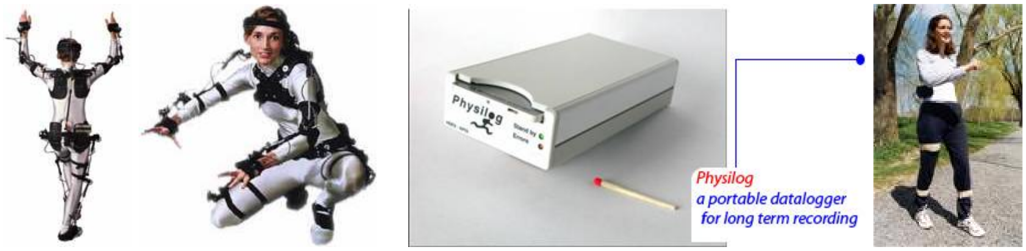
3.1 Motion Capture Techniques

None of the existing commercial motion capture systems use video as input, instead they rely on mechanical, electro-magnetic, acoustic or optical features to perform the motion estimation. These systems provide the 3D position and/or orientation of a set of markers. Some of them incorporate facilities to estimate the 3D position and orientation of the joints from those set of markers. In this section we review the different motion capture techniques and we briefly discuss different approaches to estimate joint locations and orientations from the tracked markers.

3.1.1 Mechanical Motion Capture

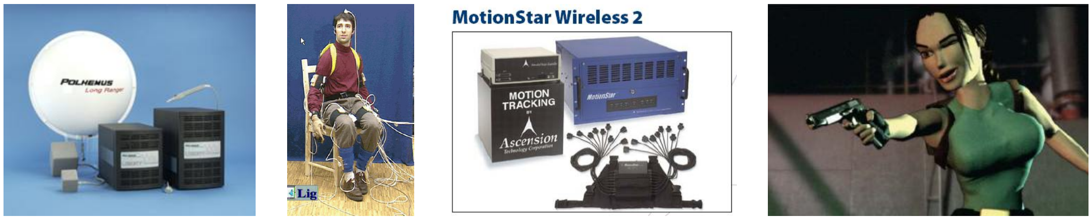
There are two main systems to perform mechanical motion capture: exo-skeletons, and ambulatory systems. An *exo-skeleton* is an electro-mechanical system that consists of a suit with small gyroscopes attached to the actor's limbs or other body parts. They detect the exact motions of all the body parts to which they are attached. The angular changes are collected on the suit and transmitted to a computer in real-time. An example of an exo-skeleton, is the Gypsy system of Meta Motion^{tm} [94] (Fig. 3.1 (a)). In the case of *ambulatory systems*, such as the Physilog [110] (Fig. 3.1 (b)), accelerometers and gyroscopes are placed on the subject. A small digital portable recorder digitizes, filters, amplifies and saves the signals that are transmitted at the end of the day to a computer. They are used in medical, orthopedict and gait analysis applications.

Unlike magnetic systems, which have important problems with metal in the environment, and optical systems that need a lot of dedicated controlled space, the mechanical systems are less restrictive. However they are always used in controlled environments. Although they are less expensive than other techniques, such as optical motion capture, they remain much



(a) (b)

Figure 3.1: **Mechanical commercial motion capture systems.** (a) Gypsy system. (b) Physilog system.



(a) (b) (c) (d)

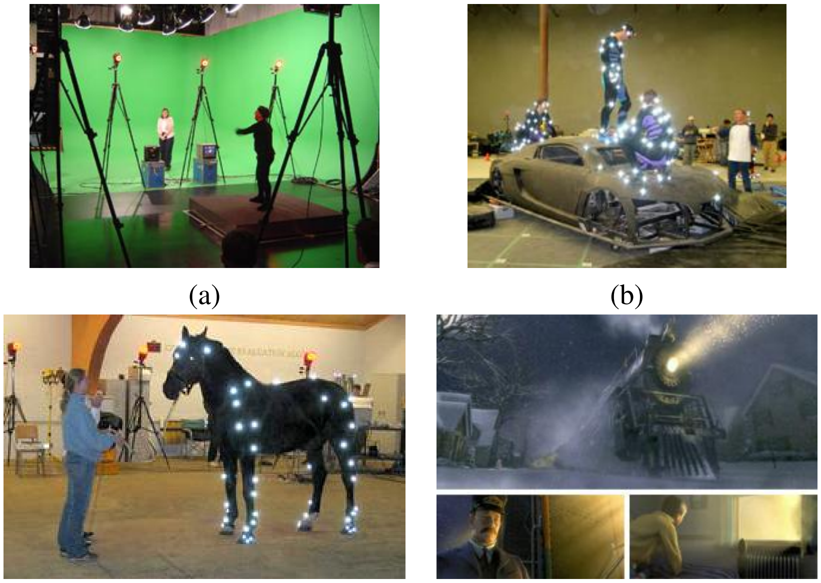
Figure 3.2: **Electro-magnetical motion capture systems.** (a) Liberty mocap system from Polhemus^{TM} [112]. (b) Cabled Flock of Birds system from Ascension^{TM} [7]. (c) Wireless electro-magnetical mocap system Motion Star from Ascension^{TM}. (d) The motion capture for Lara Craft movie was performed by Motion Star.

more expensive than video-based approaches. Moreover they are cumbersome, invasive and they suffer from drift.

3.1.2 Electro-Magnetic Motion Capture

Magnetic motion capture systems measure the low-frequency magnetic field generated by a transmitter source from sensors placed on the body. The sensors and source transmit to an electronic control unit that correlates their reported locations within the field. The electronic control units transmit this information to a computer that estimates 3D positions and rotations.

The data is usually noisy; the markers tend to move during capture sessions, requiring readjustment and recalibration. Moreover, when capturing multiple people at the same time, sensors from different actors interfere with each other, providing distorted results. Their use is very restrictive and cumbersome, since even in their wireless or cabled configurations, the system experiments interference through the presence of any magnetic field or metallic objects. However it is relatively a low cost solution. Fig. 3.2 depicts two commercial wireless magnetical motion capture systems, Liberty^{TM} of Polhemus [112], and the Motion Star from Ascension.



(a) (b)

(c) (d)

Figure 3.3: **Optical motion capture systems.** (a) Capture using a Vicon^{TM} system [155]. (b) Capture performed for the film IRobot using a Motion Analysis^{TM} [100] system. (c) Animal motion capture using Motion Analysis^{TM}. (d) Screen-shots from The Polar Express film that used motion capture obtained from a Vicon^{TM} system.

3.1.3 Acoustic Motion Capture

Acoustic motion capture systems use high-frequency sound waves in order to determine the position of the transmitters. They require the presence of at least three audio receivers in the capture volume. Each transmitter sequentially outputs a short sound, and each receiver measures the time it takes for the sound to travel to them and generates positional data by triangulation.

The major advantages are that they do not have occlusion problems, like the mechanical and electro-magnetical systems, and their low cost. However they have some disadvantages, the cabling restricts the performance of the actor; the area of capture is limited by the speed of sound and the number of transmitters. They are not suitable for high-speed movements. Moreover they suffer from possible interferences from other sound sources.

3.1.4 Optical Motion Capture

There exist two main technologies used in optical motion capture: Reflective and Pulsed-LED (light emitting diodes). Small, reflective balls, or pulsed LEDs, called markers are attached to the performer at various positions, as depicted by Fig. 3.3 (b,c). These markers are tracked by proprietary cameras. Reflective systems use IR pass filters placed over the camera lens and Infrared (IR) LEDs mounted around the camera lens. Optical motion capture systems based on Pulsed-LEDs measure the Infrared light emitted by the markers rather than light reflected from them. To help identifying the markers, some systems incorporate markers that can emit different frequencies.

These techniques are popular because they require no cabling, but they are very sensitive to marker occlusions, resulting in off-line manual interactions. There must be at least three cameras in order for the computer to be able to correctly determine the three-dimensional position of each marker. Moreover their cost is higher than other techniques. Fig. 3.3 depicts different optical motion capture systems. However nowadays, they remain the most used technique for motion capture.

3.1.5 Inferring 3D joint orientation and position

Inferring the 3D joint orientation and position is a problem encountered with all the commercial systems described above, since they provide the 3D position and/or orientation of a set of markers, and not the joint ones. Some of the previously described systems provide specific software to perform this task, as the recently developed Vicon IQ. But most of the animation studios that use these systems rely on other commercial softwares (e.g., MotionBuilder).

There has been much effort dedicated to help automate this process. Herda et al. [55] proposed an approach to marker tracking and joint estimation that uses an anatomical human model to increase the robustness and reduce the manual interaction required when occlusions happen or marker identification fails. O'Brien et al. [105] devised an algorithm to estimate skeletons from magnetic motion capture data. Because magnetical systems produce both position and orientation of the markers, they are able to solve the joint location estimation problem with a linear system. Other methods have been proposed by the computer graphics [128], computer vision [117], or robotics [25, 71] communities.

Kirk et al. [72] presented an approach to estimate the skeletal structure, by automatically clustering markers into segment groups, and to estimate the positions of the locating joints, without knowing a priori the structure of the skeleton.

These methods use global priors to ensure smoothness. These priors are of the form of simplistic first or second order Markov models. Given $\mathbf{y}_{1:t}$, a set of poses at times $(1, \cdots, t)$, the first and second order Markov assumptions are

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = p(\mathbf{y}_t|\mathbf{y}_{t-1}), \tag{3.1}$$

and

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = p(\mathbf{y}_t|\mathbf{y}_{t-1}, \mathbf{y}_{t-2}), \tag{3.2}$$

respectively.

3.1.6 Video based Motion Capture

Modeling and tracking the 3D human body from video is of great interest, as attested by recent surveys [96, 97], yet existing approaches remain brittle. The causes of the various problems include joint reflection ambiguities, occlusion, cluttered backgrounds, non-rigidity of tissue and clothing, complex and rapid motions, and poor image resolution. This implies that there is not a general method that works in all possible situations. One can consider three main possibilities to simplify the pose estimation problem:

- Add markers to be tracked from the video sequences, or use special clothes designed to simplify the task of identifying body parts (e.g. each body part is of a different color). By adding markers one simplifies the problem of tracking by the one of tracking and identifying the individual markers, and the one of estimating the 3D joint location and orientation from the marker position. This can be solved with the methods discussed in Section 3.1.5.
- Use multiple cameras and/or simplify the feature extraction process. For example, using backgrounds easy to segment, and knowing the camera internal and external parameters.
- Add an a priori knowledge of what we are going to observe to constrain the search space.

The work presented in this thesis falls into the last category, where one relies on activity-specific models that strongly constrain the 3D tracking and help resolving the potential ambiguities, but at the cost of having to infer the class of motions, and learn the models.

We now turn into discuss the different markerless motion capture approaches, that can be classified in terms of the prior information they use. The most general ones are the ones that use generic models that try to ensure smoothness between consecutive frames and try to respect the joint limits. The second category is more restrictive, and build prior models of the activities to be observed, and can be categorized in pose and motion prior models.

3.2 Generic models: ensure smoothness and joint limits

People tracking is comparatively simpler if multiple calibrated cameras can be used simultaneously. Techniques such as space carving [17, 26], 3D voxel extraction from silhouettes [95], fitting to silhouette and stereo data [32, 39, 53], and skeleton-based techniques [19, 28] have been used with some success. If camera motion and background scenes are controlled, ensuring that body silhouettes are easy to extract, these techniques can be very effective. Nevertheless, in natural scenes, with monocular video, cluttered backgrounds with significant depth variation, and many moving objects, the problem remains very challenging.

Typically approaches to tracking use simplistic primitives, such as ellipses, or cylinders to represent the human body. To represent more accurately the human body more complicated

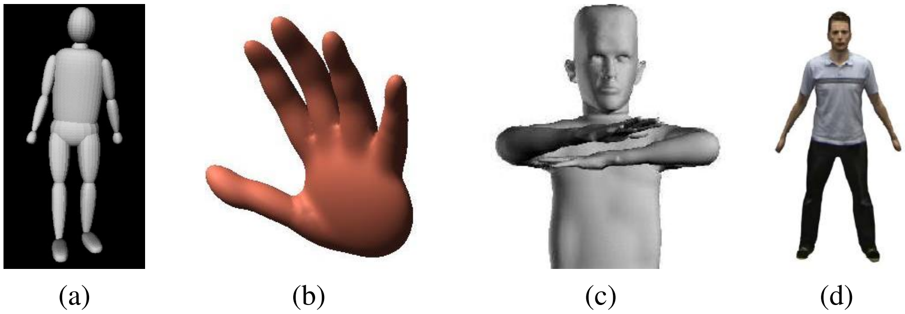


Figure 3.4: **Complex primitives for 3D Human Body Tracking.** (a) Superquadrics [133], (b-c) Implicit surfaces for hand [34] and upper body [111] tracking. (d) Texture mesh [135].

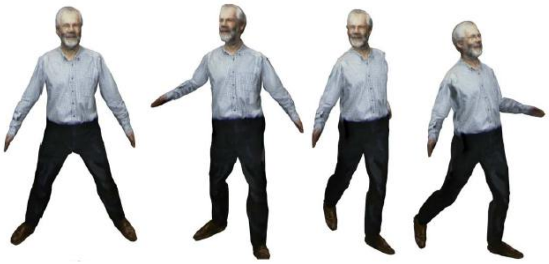


Figure 3.5: **Modeling dynamic textured scenes.** Realistic texture-mapped human synthesized from new motions, from Starck and Hilton [135]. This method requires a well-engineered capture environment.

shape models have been investigated, such as superquadrics [133], implicit surfaces [56, 111, 34], or meshes [23, 45, 135, 136], as depicted by Fig. 3.4.

An important emerging field is to model dynamic textured scenes, where instead of estimating the 3D joint positions and orientations, one tries to recover the 3D shape and texture of the articulated object [23, 135, 136]. By using enough cameras and backgrounds easy to extract, these methods result in impressive photo-realistic 3D models, as depicted by Fig. 3.5, that can be used for modeling characters in games or films.

Tracking involves pose inference at one time instant given state information (e.g., pose) from previous time instants. Tracking often fails as errors accumulate through time, producing poor predictions and hence divergence. This can be usually mitigated by introducing sophisticated statistical techniques for a more effective search [27, 30, 33, 133].

Given $\mathbf{y}_t$ the state to estimate (i.e., pose), $p(\mathbf{y}_t|\mathbf{I}_{1:t-1})$ represents all information about the state at time t that is deducible from the entire data-stream up to that time, $\mathbf{I}_{1:t}$. Using Bayes rule and assuming a first order Markov model, the tracking problem can be written as

$$p(\mathbf{y}_t|\mathbf{I}_{1:t}) = k_t p(\mathbf{I}_t|\mathbf{y}_t) p(\mathbf{y}_t|\mathbf{I}_{1:t-1}), \tag{3.3}$$

where $\mathbf{I}_t$ is an image at time t, and

$$p(\mathbf{y}_t|\mathbf{I}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{y}_t|\mathbf{y}_{t-1})p(\mathbf{y}_{t-1}|\mathbf{I}_{1:t-1})d\mathbf{x}_{t-1}, \tag{3.4}$$

and k_t is a normalization constant that does not depend on $\mathbf{y}_t$.

These two equations are called the **filtering equations**, and are key for most tracking approaches. Eq. 3.3 is called the propagation rule and should be interpreted as the Bayes rule for inferring posterior state density from data for the time-varying case [63]. $p(\mathbf{y}_t|\mathbf{I}_{1:t-1})$ is a prediction taken from the posterior $p(\mathbf{y}_{t-1}|\mathbf{I}_{1:t-1})$ from the previous time-step, where the dynamical model, $p(\mathbf{y}_t|\mathbf{y}_{t-1})$, is taken into account. In general because the observation density is non-Gaussian, $p(\mathbf{y}_t|\mathbf{I}_{1:t})$ is also non-Gaussian. The problem is how to apply a *nonlinear filter* to evaluate the state density over time, without being computationally too expensive.

The **Kalman filtering** [44] assumes that the distributions are Gaussian. Extended Kalman filtering (EKF) approximates the nonlinear dynamics and observation mappings with locally linear mappings around the current state. For motions with sudden acceleration, the local linearization could produce highly inaccurate estimation. The unscented Kalman filter (UKF) [69] is an alternative method that approximates the arbitrary state distribution with a Gaussian, but makes no assumption for the nonlinear mapping.

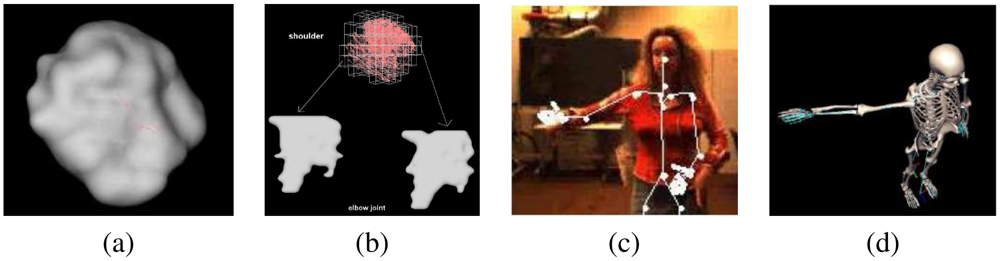
Monte Carlo sampling methods [27, 30, 33, 87, 133], such as the well-known Condensation algorithm [63], try to represent the filtering distribution $p(\mathbf{y}_t|\mathbf{I}_{1:t-1})$ with a number of weighted discrete samples. At each time the samples are propagated through the dynamics $p(\mathbf{y}_t|\mathbf{y}_{t-1})$, and re-weighted using the observation likelihood $p(\mathbf{I}_t|\mathbf{y}_t)$, to generate the estimate of the distribution of $p(\mathbf{y}_t|\mathbf{I}_{1:t})$.

The methods presented above use global dynamical priors modeling $p(\mathbf{y}_t|\mathbf{y}_{t-1})$ to ensure smoothness, in the form of simplistic first and second order Markov models. They are usually combined with simplistic min-max Euler angle joint limits [27, 132, 133]. The study of the joint limits has been a field of significant research in the Robotics and Biomechanical communities, that have developed more complex approaches, such as Joint Sinus Cones [43, 92], Spherical Polygons [73, 8], Bezier patches [146] or Single or Hierarchical Implicit Surface joint limits representations [58, 56].

In [58] we introduced sophisticated joint limits as constraints for video based motion capture. The joint limits were modeled from optical motion capture data as a closed, continuous implicit surface approximation for the quaternion orientation-space boundary. Its interior represents the complete space of valid orientations. In [57] we model hierarchical joint limits by representing the space of valid configurations for a child joint as function of the position of its parent joint, as depicted by Fig. 3.6.

Although imposing joint limits constrains much the tracking it does not solve all the problems. For example, ambiguities in monocular tracking still remain a problem when the different probable poses do not violate the joint limits. Another way to mitigate errors is to use strong pose and motion prior models, that we now turn into discuss.

The most common approach to learning motion or pose models has been to use optical



(a) (b) (c) (d)

Figure 3.6: Tracking using hierarchical joint limits. (a) Implicit surface representation of the limits for the parent joint. (b) Hierarchical representation. The child joint limits are a function of the rotation of the parent joint. (c) Tracking result projected into the image. (d) 3D tracking result from a view different than the ones used for tracking.

motion capture data from one or more people performing a specific activity. In the reminder of the chapter we discuss the different approaches to model pose and motion.

3.3 Activity-specific pose models

Recent approaches to people tracking can be viewed in terms of those that *detect* and those that *track*. Detection involves pose recognition from individual frames. It has become increasingly popular in recent research [1, 41, 99, 137], since it has the potential to make the algorithms more robust, but requires large sets of training poses to be effective.

Tracking involves pose inference at one time instant given state information (e.g., pose) from previous time instants. Tracking often fails as errors accumulate through time, producing poor predictions and hence divergence. This can be usually mitigated by introducing sophisticated statistical techniques [63, 27, 30, 33, 133], or by using strong prior motion models [3, 106, 126, 152, 149]. The optimum approach for motion estimation uses detection techniques for initialization and search, while tracking to smooth the results and gain in accuracy.

3.3.1 Linear models

One way to cope with high-dimensional data is to learn **low-dimensional models**. The simplest case involves a linear subspace projection. **Principal Component Analysis (PCA)** is the most well known linear dimensionality reducing technique. It finds a basis for the projected subspace maximizing the variance of the projected data. The basic form of PCA does not provide a probabilistic model of the data. **Probabilistic PCA (PPCA)** [120, 145] is a latent variable model which assumes that the observed data $\mathbf{y}$ is generated by a lower-dimensional data $\mathbf{x}$

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{b} + \mathbf{n}_y. \tag{3.5}$$

It has an underlying Gaussian model, where $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, \mathbf{I})$, and $p(\mathbf{n}_y) = \mathcal{N}(\mathbf{n}_y|0, \sigma^2\mathbf{I})$. PPCA is a special case of a more general class of latent variable models known as **Factor Analyzers** that model $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, \mathbf{P})$, and $p(\mathbf{n}_y) = \mathcal{N}(\mathbf{n}_y|0, \mathbf{Q})$. These linear models were used in [13, 38].

Independent component Analysis (ICA) [62] is a linear model (e.g. Eq. 3.5 holds) that projects the dataset onto a basis that best represents the statistical distribution of the data. ICA searches directions by minimizing the statistical dependence between components. This is done by maximizing the non-gaussianity of the independent components. ICA has proven very effective to solve the Blind deconvolution problem, defined as the separation of the sources of a given signal, especially in the case of sound sources. It has never been very effective when modeling poses compared to other linear techniques. PCA and ICA were used by Calinon et al. [21] to model robot gestures for imitation.

Canonical Correlation Analysis (CCA) [16] is also a linear model, but the basis are chosen minimizing the correlation between them, while having the maximum spatial or temporal correlation within each component. A similar approach is called **Maximum Autocorrelation Factors (MAF)** [139].

Linear models are tractable and produce interesting results, but they lack the ability to capture the nonlinearities of human pose. Moreover, the underlying probabilistic model is far too simple; it assumes that the highest probable sample is the mean pose. For example, when learning pose models for walking or running, the mean pose is far from the training data, and the probabilistic model is not realistic.

To handle multimodal densities one can use density estimation techniques, such as Gaussian Mixture Models or Parzen Window (e.g. [60]). See [11] for an overview of different density estimation techniques.

The **Gaussian Mixture Models (GMM)** [11] model the probability of the latent space as a mixture of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_{\mathbf{k}}, \Sigma_k), \tag{3.6}$$

where π_k is the prior probability on the Gaussian component k. Typically the GMM is estimated using an *Expectation-Maximization (EM)* algorithm, that interactively estimates π_k in the E-step, and the mean and variance of each component $\mu_{\mathbf{k}}, \Sigma_k$ in the M-step. The EM algorithm for estimating the parameters of a mixture of Gaussians is sensitive to the initialization of the parameters and in some cases can result in an inaccurate prior model.

Calinon et al. [22] use GMM to model the statistical representation of a robot task. Howe et al [60] use a mixture model density estimation to learn a distribution of short sequences of poses. With such high dimensional data, density estimation will have problems of under- and over-fitting unless one has large amounts of training data. In particular, the number of parameters quickly becomes untenable, and it can be extremely difficult to choose a reasonable number of (Gaussian) component densities.

Parzen Window density estimation [11] is a non-parametric scheme that directly uses the samples $\mathbf{x}_i$ drawn from an unknown distribution to model its density. The general form of the density is then

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} R(\mathbf{x} - \mathbf{x}_i), \tag{3.7}$$

where R is a smoothing function. The choice of R is important and conditions the quality of the estimate. The most popular one is the Gaussian distribution.

3.3.2 Mixture of Linear Models

Mixture of experts [65] such as Mixture of Principal Component Analysis [70], Mixture of Factor Analysis [46, 122] or Locally Weighted Projection regression [157] are other ways to cope with the complexity of the pose space. They try to characterize the complex global structure of the manifold by collections of simpler models, each of which describes a locally linear neighborhood.

Howe et al. [60] use a **Mixture of Factor Analyzers (MFA)** learning its density model with a GMM. They used it as a prior for monocular 3D tracking. MFA has been used for detecting heads in images [165]. Li et al. [83] use a Coordinated MFA [122] to create a low dimensional representation of the walking, and use it as a prior for monocular 3D tracking.

Locally Weighted Projection Regression (LWPR) [157, 156] is a supervised algorithm to approximate the nonlinear functions by means of piecewise linear models. The region of validity b_k of each linear model is computed from a Gaussian function

$$b_k = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x} - \mathbf{c}_k)\right), \tag{3.8}$$

where $\mathbf{c}_k$ is the center of the k-th linear model. Given an input vector $\mathbf{x}$, each linear model calculates a prediction $\mathbf{y}_k$. The total output of the networks is the weighted mean of all linear models

$$\mathbf{y} = \frac{\sum_{k=1}^{K} b_k \mathbf{y}_k}{\sum_{k=1}^{K} b_k}. \tag{3.9}$$

It can be learned in an incremental way. LWPR was used by [40] to learn the inverse kinematics problem for a robot.

Linear methods are efficient and easy to implement, but they are not suitable (too simplistic) for many problems. To cope with the complexity of modeling human poses, more sophisticated machine learning techniques are needed.

3.3.3 Non-parametric models

Non-parametric models match the image data to a large database of examples. They can handle complex motions, but they require very large amounts of training data [80, 127, 24, 123]. Further, they do not produce a density function. Mori and Malik [98] estimate the centers of the joints in the image by using shape context image matching against a set of training images with prelabeled centers. They reconstruct the 3D by using [140], which reconstructs the joint locations from point correspondences on single images. Shakhnarovich et al. [123] use locality sensitive hashing to efficiently match local models from the input data to large exemplar sets.

Richer parameterizations of human pose and motion can be found through **non-linear dimensionality reduction (NLDR)** [41, 116, 130, 160]. Following [158] the NLDR methods can be classified in two main categories, Geometrically-motivated manifold learning and Nonlinear latent variable models.

3.3.4 Geometrically-motivated manifold learning

Geometrically-motivated manifold learning recovers the embedding of the data without providing in general a density model or a mapping back to the output space $\mathbf{y}$. They can be classified in **global** and **local**, depending if they tend to preserve the global or local geometry of the data.

Locally Linear Embedding (LLE) makes the assumption that all smooth manifolds are locally linear if they are sufficiently sampled. This implies that given the samples $\mathbf{y}_i$, there exists a set of constant weights $d_{ij} \in \mathbb{R}$ such that

$$\mathbf{y}_i \approx \sum_j d_{ij}\mathbf{y}_j \tag{3.10}$$

where $\mathbf{y}_j$ are the neighbored samples of $\mathbf{y}_i$. The d_{ij} can be obtained in a closed form by minimizing

$$\epsilon(\mathbf{D}) = \sum_i \|\mathbf{y}_i - \sum_j d_{ij}\mathbf{y}_j\|^2 \tag{3.11}$$

where $\mathbf{D} = \{d_{ij}\}$, $d_{ij} = 0$ for all j, such that $\mathbf{y}_j$ is not in the neighborhood of $\mathbf{y}_i$, and $\sum_j d_{ij} = 1$. The latent coordinates are computed by assuming that the same linearity occurs in the latent space, i.e. $\mathbf{x}_i \approx \sum_j d_{ij}\mathbf{x}_j$, and is obtained by minimizing

$$\epsilon(\mathbf{x}) = \sum_i \|\mathbf{x}_i - \sum_j d_{ij}\mathbf{x}_j\|^2, \tag{3.12}$$

where the d_{ij} are the ones obtained by minimizing Eq. 3.11. This can also be done in a closed form.

Another similar technique is the **Laplacian Eigenmaps** [10]. Given a manifold $\mathcal{M}$, such that $\forall_i, \mathbf{y}_i \in \mathcal{M}$, a mapping g^{-1} from the manifold to the latent space can be computed by minimizing

$$\int_{\mathcal{M}} \|\nabla g^{-1}(\mathbf{y})\|^2. \tag{3.13}$$

This produces an embedding that best preserves the average locality. This is true because points near each other in $\mathcal{M}$ will be mapped near each other in $\mathbf{x}$ since for any derivable function f, the following inequality holds for $||\delta\mathbf{y}||$ small enough

$$\|f(\mathbf{y} + \delta\mathbf{y}) - f(\mathbf{y})\| \leq \|\nabla f(\mathbf{y})\| \|\delta\mathbf{y}\|. \tag{3.14}$$

In particular this is true for $f = g^{-1}$.

In **Multidimensional Scaling (MDS)**, given a measure of dissimilarity between pairs of training data $(\mathbf{y}_i, \mathbf{y}_j)$, a low-dimensional manifold is recovered while trying to preserve this dissimilarity in latent space. Using different dissimilarity measures, different structures can be recovered. **Isomap** [141] and its variants **C-Isomap**, **L-Isomap** [31], and **ST-Isomap** [66], are global techniques that extend MDS. In the Isomap algorithm a geodesic distance is employed, measuring the separation on a path in the manifold, and not the distance in the observation space. The ST-Isomap incorporates temporal information by penalizing, in the distance function, samples that are far in time.

However, while LLE [121], Isomap [141], and spectral embedding techniques [10] provide a low-dimensional representation of the data, they do not produce a density model in the embedding space, nor do they provide straightforward mappings between the embedding space and the full pose space. Of course, one can first learn the embedding, and then learn a density model and the inverse mapping. The density model can be learned with any of the density estimation techniques described above, such as GMM or Parzen windows. Wang et al. [160] use Isomap to learn the embedding. Then they assume a MFA and an approximate linear model based on K-nearest neighbors to learn a latent density and a mapping to the full state space. This mapping will generally be discontinuous and is therefore inappropriate for continuous optimization.

Non-linear mappings, such as Radial Basis Functions, Neural Networks or Relevance Vector Machines, are more suitable to cope with the complexity of the human pose space. A nonlinear mapping is usually defined as a generic parametric function g such that

$$\mathbf{y} = g(x, \mathbf{B}) + \mathbf{n}_y. \tag{3.15}$$

Radial Basis Function (RBF) regression assumes that the mapping can be written as

$$\mathbf{y} = \sum_i \mathbf{b}_i \varphi(\mathbf{x}, \mathbf{x}_i) + \mathbf{n}_y, \tag{3.16}$$

where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots]$. If the kernel φ is a radial basis kernel, such as $\varphi(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$, then Eq. 3.16 minimizes any cost function of the form

$$\sum_i V(g(x_i, \mathbf{B})) + \|g(x_i)\|^2. \tag{3.17}$$

Relevance vector machines (RVM) [143, 144] are a sparse Bayesian approach to regression and classification, where

$$\mathbf{y}_t = \sum_k \mathbf{b}_k \varphi_k(\mathbf{x}_t) + \mathbf{n}_y = \mathbf{BF} + \mathbf{n}_y. \tag{3.18}$$

The parameters of the mapping $\mathbf{b}_k$ are learned by minimizing

$$\|\mathbf{BF} - \mathbf{Y}\|^2 + \lambda \sum_k \log \|\mathbf{b}_k\|. \tag{3.19}$$

The RVM automatically selects the most relevant basis function to describe the problem, resulting in a sparse solution that can be very efficient, but requires a large training set.

Neural Networks (NN) [11] approximate the function g from basis functions that are adaptive (i.e. they depend on other parameters). A two layers feed-forward neural network can be written as

$$g(\mathbf{x}, \mathbf{B}) = \sum_{h=1}^{H} b_h^{(2)} \varphi \left(\sum_{i=1}^{I} b_{hi}^{(1)} \chi(x_i) + b_{ho}^{(1)} \right) + b_0^{(2)}, \tag{3.20}$$

where φ and χ can be any type of function. Typically linear, sigmoid and hyperbolic tangent functions are used.

Although we just mention three different techniques for nonlinear regression, there exist a wide broad range of approaches, such as **Self Organizing Maps (SOM)**. Their study is out of the scope of this work. Here, we described briefly three techniques that were used by relevant tracking papers.

Sminchisescu and Jepson [130] use spectral embedding for 3D pose data. Given the embedding, they learn a GMM as a density model for the training data in the embedding space, and then a mapping from the embedding to the pose space using RBF regression. To learn a prior model for walking they used several thousand training poses. Moreover, LLE and Isomap assumed that the observed data is obtained from a densely sampled manifold.

Alternatively, instead of learning a *generative model* that models $p(\mathbf{y}|\mathbf{x})$, one can learn a *discriminative model* (i.e., models $p(\mathbf{x}|\mathbf{y})$), and model the pose as a function of the images features [4, 41, 131]

Rosales and Sclaroff [118] used a set of NN to learn, after clustering, the mapping between visual features and body configurations. They use it as a *discriminative* model to perform 2D body tracking from Hu moments [61]. These moments are invariant to translation, scaling and rotation on the image plane features that are extracted from silhouettes. They use the image likelihood to decide the best estimate between the different mappings (one per cluster).

Elgammal and Lee [41] proposed a *discriminative model* that uses LLE to learn a low dimensional manifold from silhouettes. They learned both mappings, from latent space to silhouette and pose spaces, using RBFs. 3D pose is then inferred from observed silhouettes by computing the inverse mapping to the latent space and using the RBF mapping to the pose space.

3.3.5 Non-linear latent variable models

A very useful set of models are **Non-linear latent variable models (NLVM)**. They are latent variable models that are capable to model data generated from a nonlinear manifold. Examples of NLVM are density network models [89], Generative Topographic Mapping [12], and Gaussian Process Latent Variable Models [78].

Density network models [89] are an extension of supervised Bayesian Neural Networks to the unsupervised problem of density estimation. The mapping is modeled as a generic parametric function approximation (Eq. 3.15). $\mathbf{B}$ is obtained by performing gradient based minimization of the negative log posterior. Assuming independence of the observation data

$$p(\mathbf{Y}|\mathbf{B}) = \prod_i p(\mathbf{y}_i|\mathbf{B}), \tag{3.21}$$

where $p(\mathbf{y}_i|\mathbf{B})$ is normally evaluated by Monte Carlo sampling from

$$p(\mathbf{y}_i|\mathbf{B}) = \int p(\mathbf{y}_i|\mathbf{x}_i\mathbf{B})p(\mathbf{x}_i)dx_i. \tag{3.22}$$

Inference is obtained using Bayes rule and evaluating the posterior distribution $p(\mathbf{x}_i|\mathbf{y}_i\mathbf{B})$.

Assuming a regularly sampled discrete prior on $\mathbf{x}_i$, and approximating $g(\mathbf{x}, \mathbf{B})$ by an RBF network, the density network model can be solved without Monte Carlo simulation, and is called **Generative Topographic Mapping (GTM)** [12].

In the **Gaussian Process Latent Variable Model (GPLVM)** [78, 76], g takes the following form

$$g(\mathbf{x}, \mathbf{B}) = \sum_j \mathbf{b}_j \varphi_j(\mathbf{x}). \tag{3.23}$$

Since g is linear with respect to $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots]$, it can be marginalized out. This is not the case in general for $\mathbf{x}$. If the kernel is linear $\varphi(\mathbf{x}) = \mathbf{x}$, then g is also linear with $\mathbf{x}$, and can be marginalized out, resulting in PPCA. Lawrence [76] has shown that in the linear case, PPCA formulation can be obtained by marginalizing out either $\mathbf{B}$ or $\mathbf{x}$. For any function that results in a non-negative definite kernel matrix for all possible $\mathbf{X}$, the marginalization results in

$$p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right), \tag{3.24}$$

where $\mathbf{K}$ is a kernel matrix, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and $\bar{\beta}$ are the parameters of the kernel function used to compute the kernel matrix.

Grochow et al. [54] introduced the **Scaled Gaussian Process Latent Variable Model (SGPLVM)** to take into account the differences in variance of the different dimensions of $\mathbf{y}$. They learn a SGPLVM of human pose for interactive computer animation. More recently, Tian et al. [142] used a GPLVM as a *discriminative model* to constrain the estimation of 2D upper-body poses from 2D silhouettes. In Chapter 5 we advocate the use of SGPLVM to learn a generative model with a continuous mapping between the latent space and the full pose space, even for very small training sets, using it as a prior for monocular 3D body tracking [151].

While powerful, the (S)GPLVMs are static models; they have no intrinsic dynamics and do not always produce smooth latent paths given smooth time-series data. Simple dynamical models added to the SGPLVM (e.g., [142, 151]) are weak, often failing because of occlusions or anomalous jumps in the latent space.

3.4 Dynamical models

In this section we review the dynamical models that have been used for tracking.

3.4.1 Simple models

The dynamical models used in many tracking algorithms are weak. Most models are linear with Gaussian process noise, including simple first- and second-order **Markov models** [27, 67] (see Eq. 3.1 and 3.2), and learned linear **auto-regressive (AR) models** [104]. A Linear AR model of order k defines the pose at a given time t as a function of the previous poses

$$\mathbf{y}_t = \sum_{i=1}^{k} a_i \mathbf{y}_{t-i} + \mathbf{b} + \mathbf{n}_t, \tag{3.25}$$

where a_i are scalar weights, $\mathbf{b}$ is a vector representing usually the mean pose, and $\mathbf{n}_t$ is an independent (for each sample) noise, normally assumed Gaussian. [13, 38] learn an AR model in a linear subspace. These two types of models are often suitable for low-dimensional problems, and admit closed-form analysis, but they apply to a restricted class of systems. For high-dimensional data, the number of parameters that must be manually specified or learned for AR models is untenable. When used for people tracking they usually include large amounts of process noise, and thereby provide very weak temporal predictions.

3.4.2 Non-linear dynamical models

Switching LDS and **Hybrid Dynamics** provide much richer classes of temporal behaviors [64, 104, 108]. Nevertheless, they are computationally challenging to learn, and require large amounts of training data, especially as the dimension of the state space grows.

Agarwal and Triggs [4] used RVM to learn a discriminative model for monocular 3D tracking. They extend the RVM by including a second order prediction term to model the dynamics. Given $\mathbf{z}_t$, the image features at time t, the discriminative model is defined as follows

$$\mathbf{y}_t = \mathbf{A}\hat{\mathbf{y}}_t + \sum_k \mathbf{b}_k \varphi_k(\hat{\mathbf{y}}_t, \mathbf{z}_t) + \mathbf{n}_y \tag{3.26}$$

$$\hat{\mathbf{y}}_t = (\mathbf{I} + \mathbf{C})(2\mathbf{y}_{t-1} - \mathbf{y}_{t-2}) + \mathbf{D}\mathbf{y}_{t-1}, \tag{3.27}$$

where $\varphi_k(\mathbf{y}, \mathbf{z})$ is a product of two independent kernels, one modeling the similarity in pose and the other in image features.

Rahimi et al. [116] proposed a discriminative model that learns an embedding through a nonlinear RBF regression with a linear AR dynamical model to encourage smoothness in the latent space. This is a natural way to produce well-behaved latent mappings for time series data.

Troje [147] considered a related class of subspace models in which, after applying PCA, the basis functions exhibited are sinusoidal (Fourier series). A pose $\mathbf{y}_t$, is then expressed as

$$\mathbf{y}_t = \mathbf{e}_0 + \mathbf{e}_1 sin(wt) + \mathbf{e}_2 sin(wt + \psi_2) + \mathbf{e}_3 sin(wt + \psi_3) + \mathbf{e}_4 sin(wt + \psi_4), \quad (3.28)$$

where $\mathbf{e}_i$ are the eigenvectors computed from applying PCA to individual poses, and ψ_i are scalars representing the shifted phases of each sinusoidal. He finds that a subspace modeled with only three harmonics is sufficient for reliable gender classification from optical motion capture data. Calinon et al. [22] proposed a relatively similar approach, but they used **Gaussian Mixture Regression (GMR)** [138] instead of Fourier series to model poses of a robot.

3.4.3 Modeling motions

An alternative to modeling the space of plausible human poses is to directly model the space of *motions*. For example, linear subspace models have been used to model human motion, from which realistic computer animations have been produced [5, 14, 18, 154]. Subspace models learned from multiple people performing the same activity have been used successfully for 3D people tracking [106, 126, 164]. For the restricted class of cyclic motions, Ormoneit et al. [106] developed an automated procedure for the alignment of training data as a precursor to PCA.

Linear subspace models also provide natural statistical priors that generalize naturally to motions not in the training set. This makes them amenable for Bayesian formulations of 3D people tracking. Indeed, most existing trackers use some form of multiple-hypothesis algorithm, such as a particle filter, within a probabilistic framework. In Chapter 4, we exploit the fact that this class of models permits a formulation of the tracking problem as one of minimizing differentiable objective functions, which might be accomplished much more efficiently with simple deterministic optimization. We consider both cyclic motions and acyclic motions, and we demonstrate models learned from multiple activities, for which one can both track and recognize the class of motion. Moreover, these models can also be used for motion extrapolation. We will show how to use them to infer new motions of a subject that is observed once performing a given activity, while respecting his/her particular style.

But these models required a relatively big amount of training data compared to Gaussian Process. Using the same philosophy as [130] one can learn the (S)GPLVM, and then learn a complex dynamical model for the latent space, such as the AR models, Switching LDS [64, 104, 108], Hybrid Dynamics, or RVM [4] described above. The learning of the low dimensional space and of the dynamical model is not done at the same time; not being optimal. This is not the case when learning non-linear dynamical latent variable models.

3.4.4 Non-linear dynamical latent variable models

An alternative to modeling complex dynamics is to use the **Gaussian Process Dynamical Model (GPDM)** proposed by Wang et al. [159]. The GPDM comprises a mapping from a latent space to the data space, and a dynamical model in the latent space. These mappings are typically non-linear. The first-order GPDM assumes

$$\begin{aligned}
\mathbf{x}_t &= f(\mathbf{x}_{t-1}, \mathbf{A}) + \mathbf{n}_x = \sum_j \mathbf{a}_j \varphi_j(\mathbf{x}_{t-1}) + \mathbf{n}_{x,t} \qquad (3.29) \\
\mathbf{y}_t &= g(\mathbf{x}_t, \mathbf{B}) + \mathbf{n}_y = \sum_j \mathbf{b}_j \chi_j(\mathbf{x}_t) + \mathbf{n}_{y,t}. \qquad (3.30)
\end{aligned}$$

By marginalizing out the mapping parameters, $\mathbf{A}$ and $\mathbf{B}$, this results in two Gaussian processes, one modeling the reconstruction and one modeling the dynamics [159].

While powerful, the GPDM still suffers from jumps in the latent space when the dimensionality of the observations is high with respect to the latent dimension. In this work we introduce a new set of models, called **Balanced-GPDMs** that assume narrow priors for the dynamics to avoid the reconstruction overfitting present in the GPDM when learning high-dimensional spaces [150]. In Chapter 6 we use the Balanced-GPDM to facilitate tracking in the presence of very noisy data and significant occlusions. This is similar in spirit to [116], however we employ non-linear dynamics and obtain a simple generative mapping from the latent space to the pose space, all within a consistent probabilistic framework [150].

4 Motion Models

Prior models of pose and motion play a central role in 3D monocular people tracking, mitigating problems caused by ambiguities, occlusions, and image measurement noise. While powerful models of 3D human *pose* are emerging, there has been comparatively little work on *motion* models. Most state-of-the-art approaches rely on simple Markov models that do not capture the complexities of human dynamics. Accordingly, this often results in a more challenging optimization (or inference) problem, for which Monte Carlo techniques (e.g., particle filters) are often used to cope with ambiguities and local minima [27, 33, 63, 126, 133]. Most such methods suffer computationally as the number of degrees of freedom in the model increases.

In this chapter, we use activity-specific motion models to help overcome this problem. We show that, while complex non-linear methods are required to learn pose models, one can use simple algorithms such as PCA to learn effective motion models, both for cyclic motions such as walking and running, and acyclic motions such as a golf swing (e.g., see Figs. 4.1 and 4.2). This may not be intuitive, one expects the learning of a lower dimensional space (pose) to be less complex than the learning of a higher dimensional one (motion), whose input vectors are concatenations of the lower dimensional ones. But learning noiseless segmented and time warped motion models is equivalent to learn pose models with the phase of the motion as hidden state. This means that, under the motion models a pose is obtained by combining poses performed at similar phase times, resulting in more convex spaces.

With such motion models we formulate and solve the tracking problem in terms of continuous objective functions whose differential structure is rich enough to take advantage of standard optimization methods. This is significant because the computational requirements of these methods are typically less than those of Monte Carlo methods. This is demonstrated here with two tracking formulations, one for monocular people tracking, and one for multiview people tracking. Moreover, with these subspace motion models we also show that one can perform motion-based recognition of individuals and activities. Finally we present a new method for motion extrapolation, where these motion models are used to infer new motions of a subject that is observed once performing a given activity, while respecting his/her particular style.

4.1 Linear Motion Models

This chapter extends the use of linear subspace methods for 3D people tracking. In this section we describe the protocol we use to learn cyclic and acyclic motions, after which we

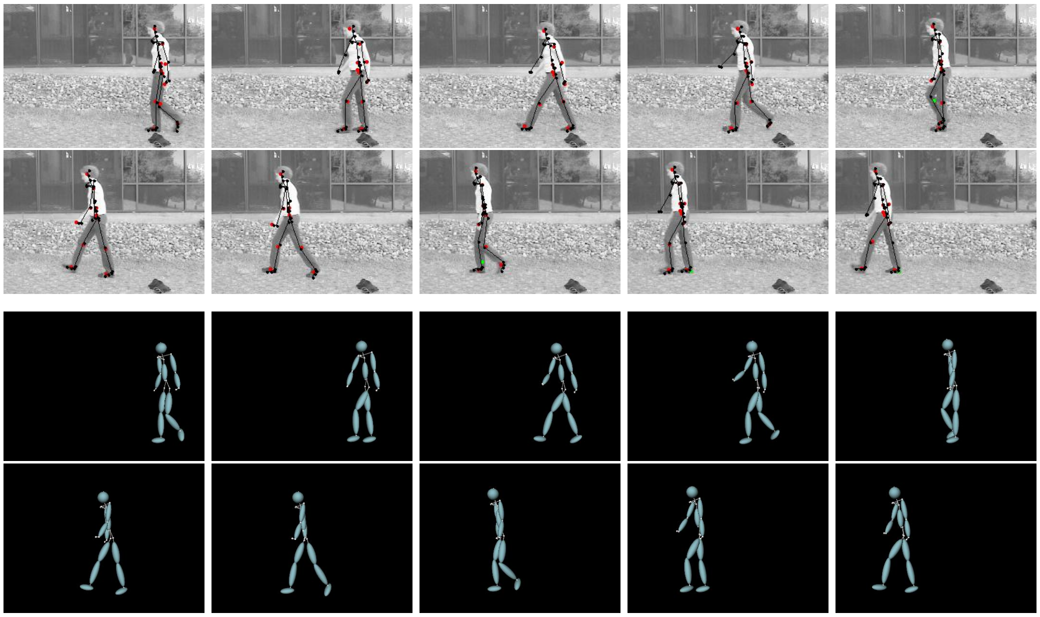


Figure 4.1: **Monocular Tracking a walking motion** of 43 frames. **First two rows**: The skeleton of the recovered 3D model is projected onto the images. **Bottom two rows**: Volumetric primitives of the recovered 3D model projected into a similar view.

discuss the important properties of the models. We show how they tend to cluster similar motions, and that the linear embedding tends to produce convex models. These properties are important for the generalization to motions outside of the training set, to facilitate tracking with continuous optimization, and for motion-based recognition.

We represent the human body as the set of volumetric primitives attached to an articulated 3–D skeleton, like those depicted in Figs. 4.1 and 4.2. A *pose* is given by the position and orientation of its root node, defined at the sacroiliac, and a set of joint angles. More formally, let D denote the number of joint angles in the skeletal model. A pose at time t is then given by a vector of joint angles, denoted $\psi_t \in \mathcal{R}^P$, along with the global position and orientation of the root, denoted $\mathbf{g}_t \in \mathcal{R}^6$.

A *motion* is a temporal sequence of poses, comprising a sequence of joint angle vectors and a sequence of global positions and orientations:

$$\mathbf{y} = [\psi_1^T, \cdots, \psi_M^T]^T \ \in \ \mathcal{R}^{P\,M} \quad , \tag{4.1}$$

$$\mathbf{G} = [\mathbf{g}_1^T, \cdots, \mathbf{g}_M^T]^T \ \in \ \mathcal{R}^{6\,M} \quad , \tag{4.2}$$

where M is the number of time steps at which the underlying continuous motion is sampled. The dimensionality of my observations is $D = P * M$, much higher than in pose space models. We choose the sampling rate to be sufficiently high so that we can interpolate the

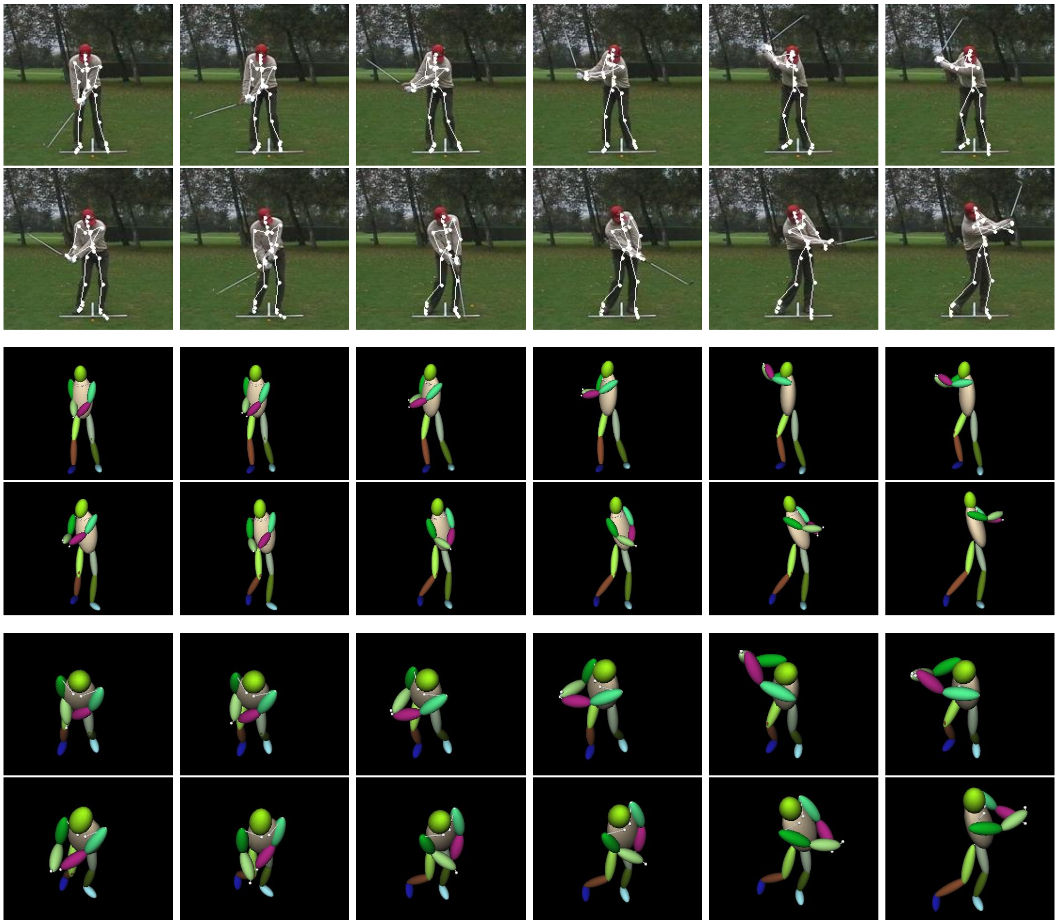


Figure 4.2: **Monocular Tracking a full swing** in a 45 frame sequence. **First two rows**: The skeleton of the recovered 3–D model is projected into a representative subset of images. **Middle two rows**: Volumetric primitives of the recovered 3–D model projected into the same views. **Bottom two rows**: Volumetric primitives of the 3–D model as seen from above.

continuous pose signal. Finally, instead of using the sample times, $n \in [1, M]$, we define the normalized time, or *phase* of the motion, as

$$\mu_n \; = \; \frac{n-1}{M-1} \; , \; \mu_n \in [0, 1]. \tag{4.3}$$

Here, μ_n increases linearly from 0 at the beginning of the motion to 1 at the end.

4.1.1 PCA Motion Model

Linear subspace motion models are learned from optical motion capture data, given one or more people performing the same activity several times. Because different people perform the same activity with some variability in speed, we first dynamically time-warp and re-sample each training sample. This produces training motions with the same number of samples, and with similar poses aligned. To this end, we first manually identify a small number of key postures specific to each motion type. We then linearly time warp the motions so that the key postures are temporally aligned. The resulting motions are then re-sampled at regular time intervals using quaternion spherical interpolation [125] to produce the training poses $\{\psi_j\}_{j=1}^{M}$.

Given a training set of N such motions, denoted, $\{\mathbf{y}_j\}_{j=1}^{N}$, we use Principal Component Analysis to find a low-dimensional basis with which we can effectively model the motion. In particular, the model approximates motions in the training set with a linear combination of the mean motion Θ_0 and a set of *eigen-motions* $\{\Theta_i\}_{i=1}^{d}$:

$$\mathbf{y} \; \approx \; \Theta_0 + \sum_{i=1}^{d} x_i \Theta_i = \Theta_0 + \mathbf{B}\mathbf{x} \;\; , \tag{4.4}$$

where $\mathbf{B} = [\Theta_1, \cdots, \Theta_d]$. This is equivalent to Eq. 3.5 with $\mathbf{b} = \Theta_0$. The scalar coefficients, x_i, characterize the motion, and $d \leq D$ controls the fraction of the total variance of the training data that is captured by the subspace, denoted by $Q(d)$:

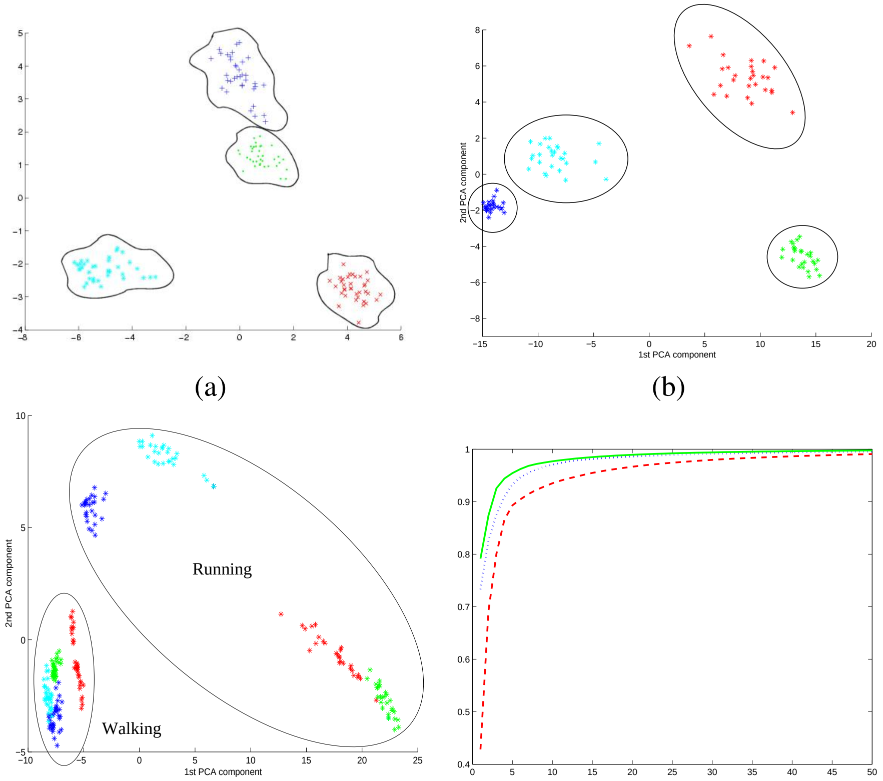
$$Q(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i}, \tag{4.5}$$

where λ_i are the eigenvalues of the data covariance matrix, ordered such that $\lambda_i \geq \lambda_{i+1}$. In what follows we typically choose m so that $Q(d) > 0.9$.

4.1.2 Cyclic motions

We first consider models for walking and running. We used a Vicon^{tm} optical motion capture system to capture the motions of two men and two women on a treadmill:

- walking at 9 speeds ranging from 3 to 7 km/h, by increments of 0.5 km/h, for a total of 144 motions;



(a) (b)

(c) (d)

Figure 4.3: **Motion models**. First two PCA components for (a) 4 different captures of 4 subjects walking at speeds varying from 3 to 7km/h, (b) the same subjects at speeds ranging from 6 to 12km/h, (c) the multi-activity database composed of the walking and running motions together. The data corresponding to different subjects is shown in different styles. The solid lines separating clusters have been drawn manually for visualization purposes. (d) Percentage of the database that can be generated with a given number of eigenvectors for the walking (dashed red), running(solid green) and the multi-activity databases(dotted blue).

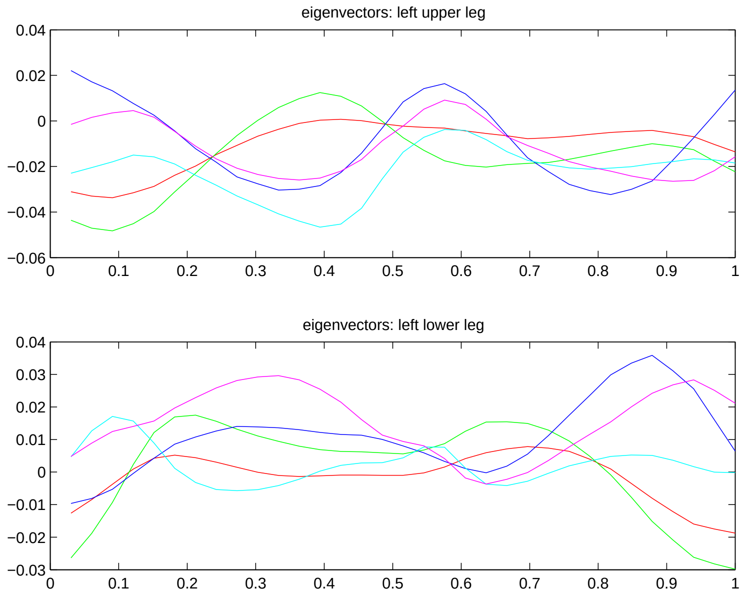


Figure 4.4: **Walking motion eigenvectors.** The first one appears in red, green the second, blue the third, cian the fourth and magenta for the last one. Eigenvectors of the flexion-extension in the sagittal plane of the upper leg on the top and flexion-extension in the sagittal plane of the lower leg on the right.

- running at 7 speeds ranging from 6 to 12 km/h, by increments of 1.0 km/h, for a total of 112 motions.

The body model had $P = 84$ degrees of freedom. While one might also wish to include global translational or orientational velocities in the training data, these were not available with the treadmill data, so we restricted ourselves to temporal models of the joint angles. The start and end of each gait cycle were manually identified. The data were thereby broken into individual cycles, and normalized so that each gait cycle was represented with $M = 33$ pose samples. The dimension of the observation space $\mathbf{y}$ is then $D = 33 * 84 = 2772$. Four cycles of walking and running at each speed were used to capture the natural variability of motion from one gait cycle to the next for each person.

In what follows we learn a motion model for walking and one for running, as well as multi-activity model for the combined walking and running data. In Fig. 4.3(d) we display $Q(d)$ in (4.5) as a function of the number of eigen-motions for the walking, running and the combined datasets. We find that in all three cases five eigen-motions out of a possible 144 for walking, 112 for running and 256 for the multi-activity data, capture more than 90% of the total variance. In the experiments below we show that these motion models are sufficient to generalize to styles that were not captured in the training data, while eliminating the noise present in the less significant principal directions.

The first five walking eigenvectors, Θ_i, for the upper and lower leg rotations in the sagittal plane are depicted by Fig. 4.4 as a function of the gait phase μ_t. One can see that they are smooth and therefore easily interpolated and differentiated (numerically). Fig. 4.5 illustrates the individual contributions of the first five eigen-motions. The first row shows the mean motion alone. In each subsequent row we show a linear combination of the mean motion and the i^{th} eigen-motion, for $i = 1...5$. Each row therefore illustrates the influence of a different eigen-motion. While one cannot expect the individual eigen-motions to have any particular semantic meaning, their behavior provides some intuitions about the nature of the underlying model.

4.1.3 Golf Swing

We use the same approach to learn the swing of a golf club. Toward this end, we used the $N = 9$ golf swings of the CMU database [29]. The corresponding body model has $P = 72$ degrees of freedom. We identified the 4 key postures depicted in Fig. 4.6, and linearly time-warped the swings so that the same key postures are temporally aligned. We then sampled the warped motions to obtain motions vectors with $M = 200$ poses. The sampling rate here is higher than the one used for walking and running since a golf swing contains fast speeds and large accelerations. The dimensionality of the observation space is then $D = 200 * 72 = 14400$. Given the small number of available training motions we only used $d = 4$ coefficients, capturing more than 90% of the total variance.

4.1.4 Motion Clustering

Troje [147] claims that with effective motion models one can perform interesting motion-based recognition. He showed that one can classify gender and other individual attributes including emotional states. In this context it is of interest to note that the subspace motion models learned here exhibit good inter-subject and inter-activity separation, suggesting that these models may be useful for recognition. For example, Fig. 4.3(a) shows the walking training motions, at all speeds, projected onto the first two eigen-motions of the walking model. Similarly, Fig. 4.3(b) shows the running motions, at all speeds, projected onto the first two eigen-motions of the running model. The closed curves in these figures were drawn manually to help illustrate the large inter-subject separation. One can see that the intra-subject variation in both models is much smaller than the inter-subject variation.

The motion model learned from the combination of walking and running training data shows large inter-activity separation. Fig. 4.3(c) shows the projection of the training data onto the first two eigen-motions of the combined walking and running model. One can see that the two activities are easily separated in this subspace. The walking components appear on the left of the plot and form a relatively dense set. By contrast, running components are sparser because inter-subject variation is larger, indicating that more examples are required for a complete model.

While the motion models exhibit this inter-subject and inter-activity variation, we would not expect pose models to exhibit similar structure. As an example to demonstrate this, we

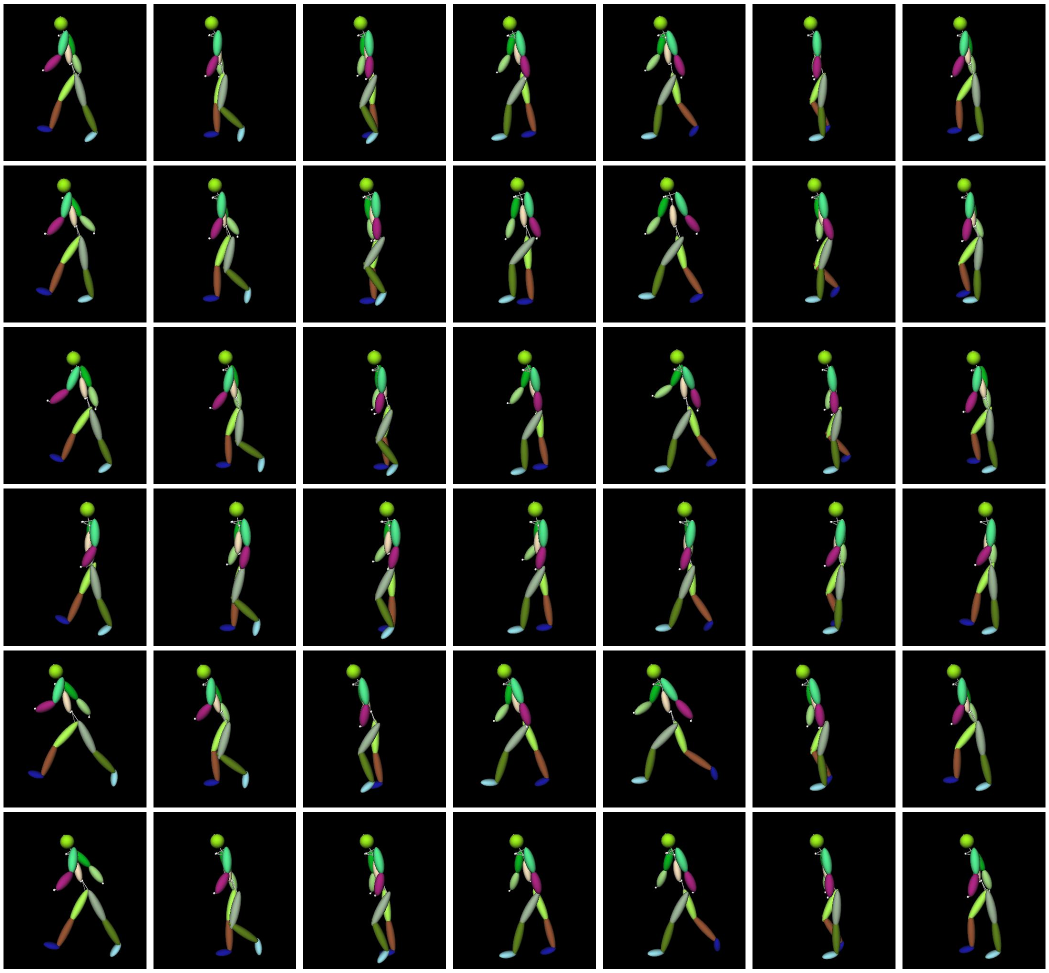


Figure 4.5: **Individual contribution of the eigenvectors.** The top row shows equispaced poses of the mean walk. The next 5 rows illustrate the influence of the first 5 eigen-motions. The second row shows a linear combination of the mean walk and the first eigen-motion, $\Theta_0 + 0.7\Theta_1$. Similarly, the third row depicts $\Theta_0 + 0.7\Theta_2$ to show the influence of the second eigen-motion, and so on for the remaining 3 rows.

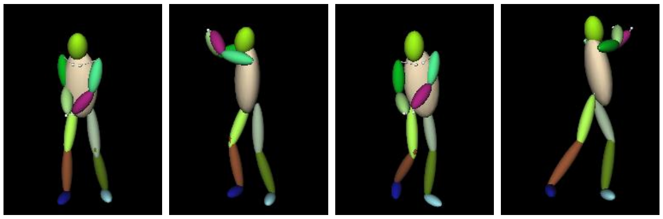
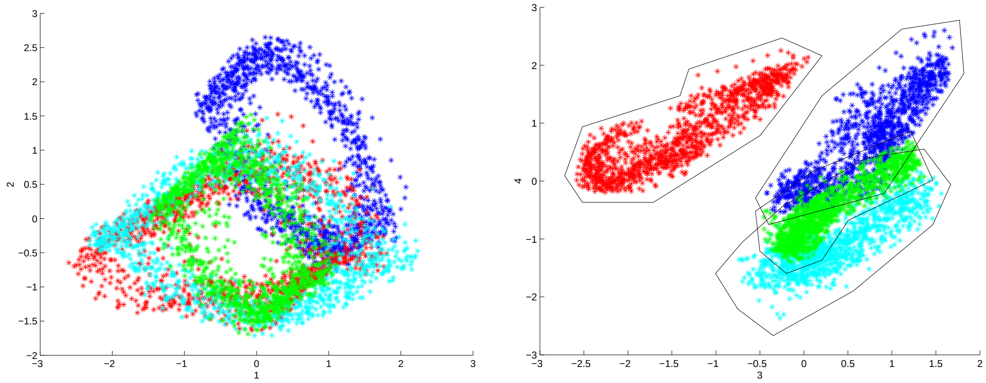


Figure 4.6: **Key postures** for the golf swing motion capture database that are used to align the training data: Beginning of upswing, end of upswing, ball hit, and end of downswing. The body model is represented as volumetric primitives attached to an articulated skeleton.

also learned a pose model from the walking poses in the walking dataset. Fig. 4.7 shows poses from the walking data projected onto the first four eigen-directions of the subspace model learned from poses in the walking motions. It is clear that there is no inter-subject separation in the pose model.

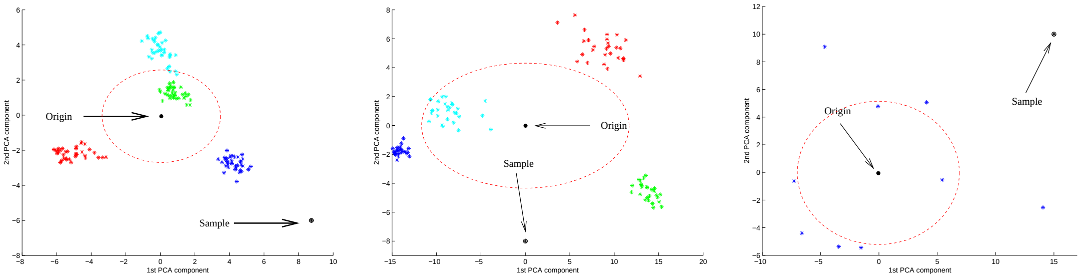


(a) (b)

Figure 4.7: **Poor clustering in a pose subspace.** The solid lines that delimited clusters have been manually done for visualization purposes. (a) Projection of training poses onto the first two eigen-directions of the pose subspace. (b) Projection of training poses onto the third and fourth eigen-directions of the pose subspace. While in the motion motion there is strong inter-subject separation, with the pose model in this figure, there is no inter-subject separation.

4.1.5 Model Convexity

PCA provides a subspace model within which motions are expressed as (arbitrary) linear combinations of the eigen-motions (4.4). With probabilistic PCA [145] one further con-



(a) (b) (c)

Figure 4.8: **Sampling the single activity databases.** In each plot we show the most probable sample that is at the origin, and a sample with very low probability (far from the origin) for the (a) walking, (b) running, and (c) golfing databases. Their respective motions are shown in Fig. 4.9. The dashed curves are one standard deviation ellipses for the underlying Gaussian density model for the data.

strains the model with a multivariate Gaussian density. A key property of such linear models is the convexity of the motions, i.e., that linear combinations of motions (or eigen-motions) are legitimate motions.

While convexity is clearly violated with pose data (cf., Fig. 4.7(a), we find that with the subspace motion models convexity is satisfied to a much greater extent. In other words, we find that random samples from the subspaces, according to a subspace Gaussian model for walking, running and the golf swing, all produce plausible motions. Fig. 4.9 depicts two motions from each of (a) the walking model, (b) the running model, and (c) the golfing model. The first row in each case depicts the mean motion for each model, corresponding to the origin of the respective subspaces. As shown in Fig. 4.8 the origin is relatively far from any particular training motion, yet these motions look quite plausible. The second motion in each case corresponds to a point drawn at random that is far from the origin and any training motion (as shown in Fig. 4.8). These points, typical of other random samples from the underlying Gaussian density, also depict plausible motions. Accordingly, the models appear to generalize naturally to points relatively far from the training data.

The multi-activity model learned from the combined running and walking data does not satisfy convexity to the same extent. Fig. 4.10 shows the subspace spanned by the first two eigen-motions of the combined model. In addition to the training data, the figure shows the locations of four points that lie roughly between the projections of the walking and running data. The four rows of Fig. 4.11 depict the corresponding motions (for which the remaining subspace coefficients, $x_j = 0$, $3 \leq j \leq d$, were set to zero). While three of the motions are plausible mixtures of running and walking, the top row of Fig. 4.11 clearly shows an implausible motion. Here we find that points close to the training data generate plausible motions, but far from the training data the motions become less plausible.

Nevertheless there are regions of the subspace between walking and running data points that do correspond to plausible models. These regions facilitate transitions between walking and running that are essential if we wish to be able to track subjects through such transitions (as shown below).

4.2 Tracking Framework

In this section we show how the motion models of Section 4.1 can be used for 3D people tracking. Our goal is to show that with activity-specific motion models one can often formulate and solve the tracking problem straightforwardly with deterministic optimization. Here, tracking is expressed as a nonlinear least-squares optimization, and then solved using Levenberg-Marquardt [114].

The tracking is performed with a sliding temporal window. At each time instant t we find the optimal target parameters for f frames within a temporal window from time t to time $t+\tau$. Within this window, the relevant target parameters include the subspace coefficients, $\{x_i\}_{i=1}^d$, where x_i is the i-th dimension of the latent variable $\mathbf{x}$, the global position and orientation of the body at each frame $\{\mathbf{g}_j\}$ and the phases of the motion at each frame $\{\mu_j\}$, for $t \leq j \leq t+\tau$:

$$\phi_t \;=\; [\mathbf{x}, \mu_{t:t+\tau}, \mathbf{g}_{t:t+\tau}] \;\; . \tag{4.6}$$

While the global pose and phase of the motion vary throughout the temporal window, the subspace coefficients are fixed over the window.

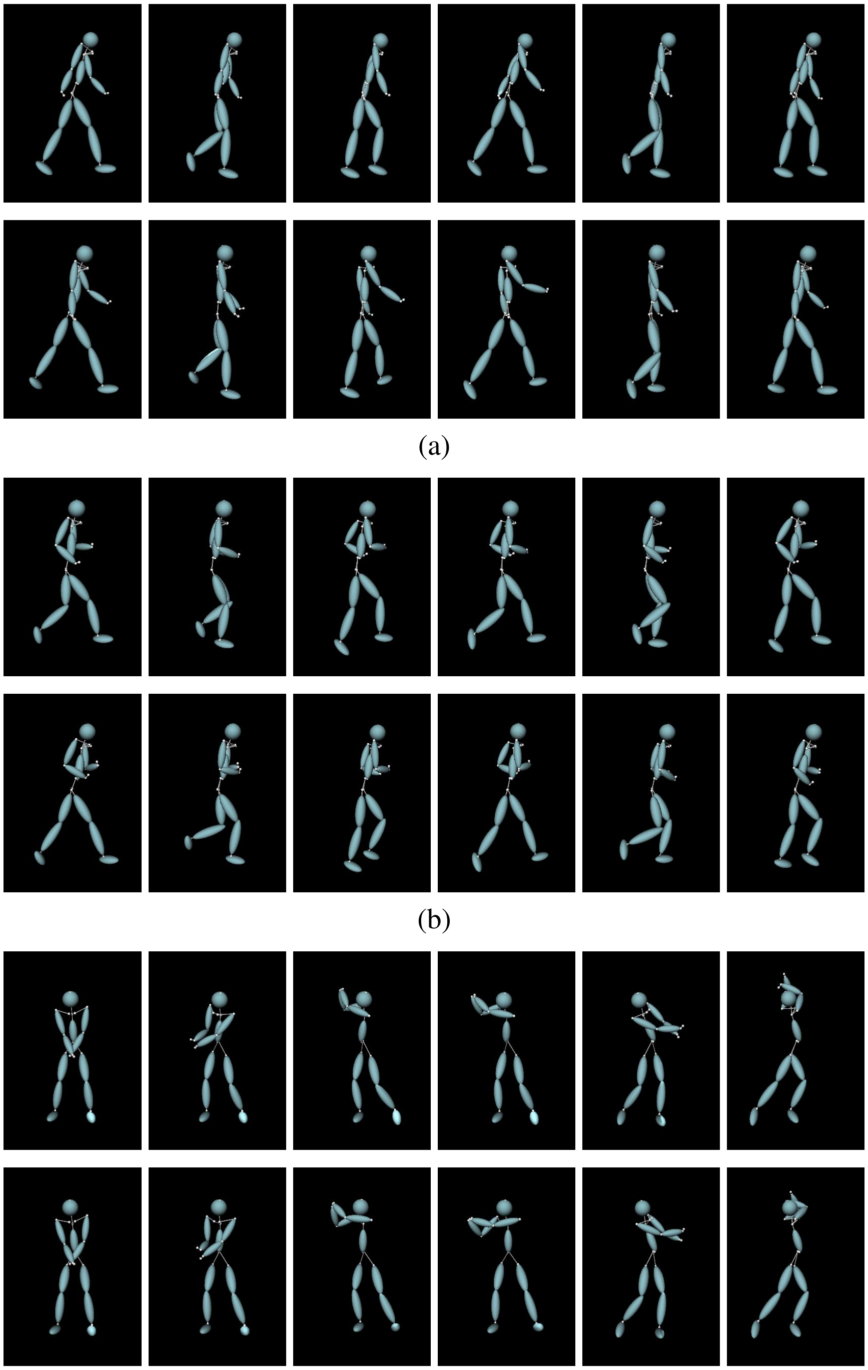
After minimizing an objective function over the unknown parameters $\mathbf{phi}_t$, we retain the pose estimate at time t given by the estimated latent variable $\hat{\mathbf{x}}$, along with the global parameters and phase at time t, i.e., $\hat{\mathbf{g}}_t$ and $\hat{\mu}_t$. Because the temporal estimation windows overlap from one time instant to the next, the estimated target parameters tend to vary smoothly over time. In particular, with such a sliding window the estimate of the pose at time t is effectively influenced by both past and future data. It is influenced by past data because we assume smoothness between parameters at time t and estimates already found at previous time instants $t-1$ and $t-2$. It is influenced by future data as data constraints on the motion are obtained from image frames at times $t+1$ through $t+\tau$.

4.2.1 Objective Function

We use the image data to constrain the target parameters with a collection of n_{obs} constraint equations of the form

$$O(\mathbf{x}_i, \phi) \;=\; \epsilon_i \;\; , \quad 1 \leq i \leq n_{\mathrm{obs}} \;\; , \tag{4.7}$$

where the $\mathbf{p}_i$ are 2D image features, O is a differentiable function whose value is zero for the correct value of ϕ and noise-free data, and ϵ_i denotes the residual error in the i^{th} constraint. Our objective is to minimize the sum of the squared constraint errors. Because some measurements may be noisier than others, and our observations may come from different image properties that might not be commensurate with one another, we weight each constraint of type $type$ with a constant, γ^{type}. In effect, this is equivalent to a model in which



(a)

(b)

(c)

Figure 4.9: **Sampling** the first five components of each single activity database produce physically possible motions. The odd rows show the highest probability sample that for each single-motion database, which is the at the origin $x_i = 0, \forall i$. The even rows show some low probability samples very far from the training motions to demonstrate that even those samples produce realistic motions. The coefficients for these motion are shown in Fig. 4.8 (a,b,c) respectively. **First two rows (a):** Walking, **Middle rows (b):** Running, **Last two rows (c):** Golf swing samples.

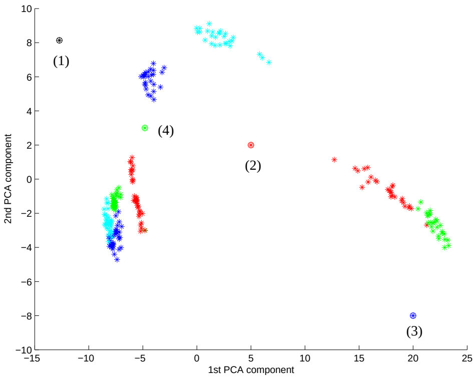


Figure 4.10: **Sampling the multi-activity subspace.** The 4 samples that generate the motions of Fig. 4.11 are depicted.

the constraint residuals are IID Gaussian with isotropic covariance, and the weights are just inverse variances.

Finally, since image data are often noisy, and sometimes under-constrain the target parameters, we further assume regularization terms that encourage smoothness in the global model. We also assume that the phase of the motion varies smoothly. The resulting total energy to be minimized at time t, F_t, can therefore be expressed as

$$\begin{aligned} F_t &= \sum_{i=1}^{nobs} \gamma^{type_i} \left\| O^{type_i}(\mathbf{p}_i, \phi) \right\|^2 + \gamma_z \sum_{j=t}^{t+\tau} \|\mathbf{z}_j - \hat{\mathbf{z}}_j\|^2 \\ & +\gamma_o \sum_{j=t}^{t+\tau} \|\mathbf{o}_j - \hat{\mathbf{o}}_j\|^2 + \gamma_\mu \sum_{j=t}^{t+\tau} (\mu_j - \hat{\mu}_j)^2 + \gamma_x \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \ , \end{aligned} \tag{4.8}$$

where O^{type} is the function that corresponds to a particular observation type, $\mathbf{z}_t$ and $\mathbf{o}_t$ are the global translation and rotation such that $\mathbf{g}_t = (\mathbf{z}_t, \mathbf{o}_t)$, γ_z, γ_o, γ_μ and γ_x are scalar weights, and $\hat{x}_i$ denote the coefficients estimated in the previous window of $\tau + 1$ frames. The deterministic predictions, $\hat{\mathbf{z}}_j$, $\hat{\mathbf{o}}_j$ and $\hat{\mathbf{y}}_j$, assume zero acceleration and therefore take the form:

$$\hat{\mathbf{z}}_j = 2\mathbf{z}_{j-1} - \mathbf{z}_{j-2} \ , \quad \hat{\mathbf{o}}_j = 2\mathbf{o}_{j-1} - \mathbf{o}_{j-2} \ , \quad \hat{\mu}_j = 2\mu_{j-1} - \mu_{j-2} \quad .$$

The variables $\mathbf{z}_{t-1}$, $\mathbf{z}_{t-2}$, $\mathbf{o}_{t-1}$, $\mathbf{o}_{t-2}$, μ_{t-1} and μ_{t-2} are taken to be the values estimated from previous two time instants, and are therefore fixed during estimation at time t.

Given a function F_t that is differentiable, and bearing in mind that the derivatives of F_t with respect to the D individual joints angles $\frac{\partial F_t}{\partial \theta_j}$ can be easily computed [111], the estimation of the Jacobian simply involves computing

$$\frac{\partial F_t}{\partial x_i} = \sum_{j=1}^{D} \frac{\partial \theta_j}{\partial x_i} \cdot \frac{\partial F_t}{\partial \theta_j} \ , \qquad \frac{\partial F_t}{\partial \mu_t} = \sum_{j=1}^{D} \frac{\partial \theta_j}{\partial \mu_t} \cdot \frac{\partial F_t}{\partial \theta_j} \ , \tag{4.9}$$

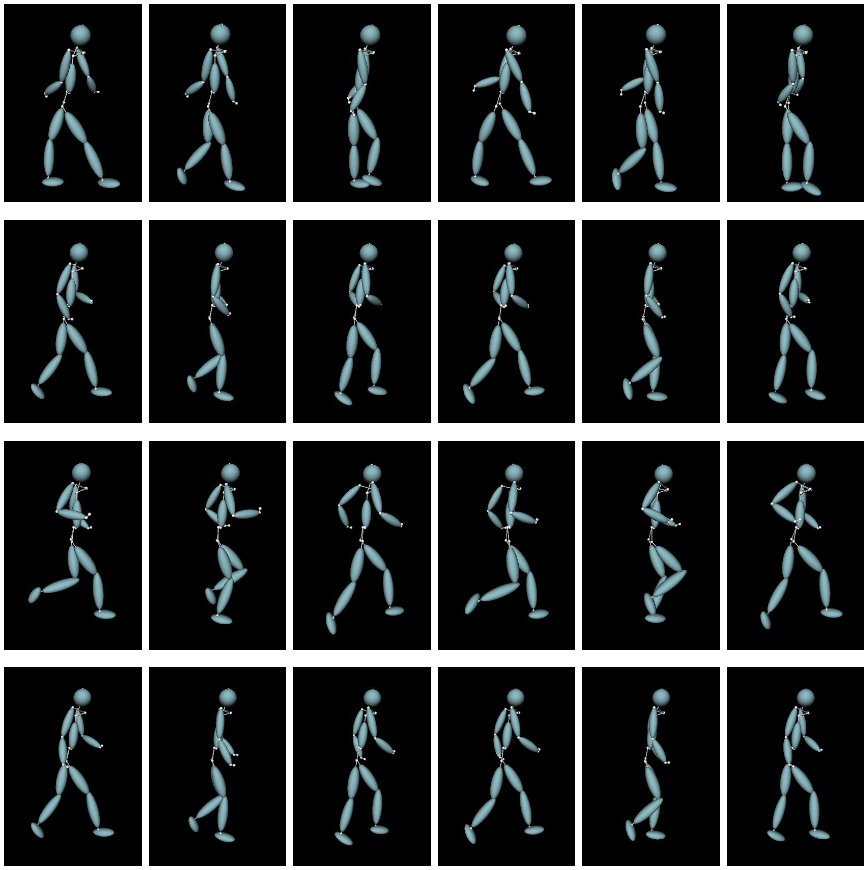


Figure 4.11: **Sampling** the first two components of a multi-activity database compose of walking and running motions can produce physically impossible motions. The coefficients of the motions depicted in this figure are shown in Fig. 4.10. **Top row:** Physically impossible motion. The input motion space compose of walking and running is not convex. **Middle row:** Physically possible motion close to a walking. **Bottom row:** Motion close to a running. As the convexity of the input space is assumed when doing PCA, and it may not be the case, the resulting motion as a combination of principal directions can be physically impossible.

where $\psi \equiv [\theta_1, \cdots, \theta_D]$, and $\mathbf{y} \equiv [\psi_1, \cdots, \psi_N]$ is given by (4.4). Because the θ_j are linear combinations of the Θ_i eigenvectors, $\frac{\partial \theta_j}{\partial x_i}$ reduces to Θ_{ij}, the jth coordinate of Θ_i. Similarly, we can write

$$\frac{\partial \theta_j}{\partial \mu_t} = \sum_{i=1}^{m} x_i \frac{\partial \Theta_{ij}}{\partial \mu_t} ,$$

where the $\frac{\partial \Theta_{ij}}{\partial \mu_t}$ can be evaluated using finite differences and stored when building the motion models.

4.2.2 Computational Requirements

Monte Carlo approaches, like that in [126], rely on randomly generating particles and evaluating their fitness. Because the cost of creating particles is negligible, the main cost of each iteration comes from evaluating a log likelihood, such as F_t in (4.9), for each particle. In a typical particle filter, like the Condensation algorithm [63], the number of particles needed to effectively approximate the posterior on a D-dimensional state space grows exponentially with D [27, 88]. With dimensionality reduction, like that obtained with the subspace motion model, the state dimension is greatly reduced. Nevertheless, the number of particles required can still be prohibitive [126].

By contrast, the main cost at each iteration of our deterministic optimization scheme comes from evaluating F_t and its Jacobian. In our implementation, this cost is roughly proportional to $1 + \log(D)$ times the cost of computing F_t alone, where D is the number of joint angles of (4.9). The reason this factor grows slowly with D is that the partial derivatives, $\frac{\partial F_t}{\partial \theta_j}$, which require most of the computation, are computed analytically and involve many intermediate results than can be cached and reused. As a result, with R iterations per frame, the total time required by our algorithm is roughly proportional $R(1 + \log(D))$ times the cost of evaluating F_t. Since we use a small number of iterations (less than 15 for all experiments in this chapter), the total cost of our approach remains much smaller than typical probabilistic methods.

4.3 Monocular Tracking

We first demonstrate our approach in the context of monocular tracking [149]. Since we wish to operate outdoors in an uncontrolled environment, tracking people wearing normal clothes, it is difficult to rely solely on any one image cue. Here we therefore take advantage of several sources of information.

4.3.1 Projection Constraints

To constrain the location of several key joints, we track their approximate image projections across the sequence. These 2D joint locations were estimated with a 2D image-based

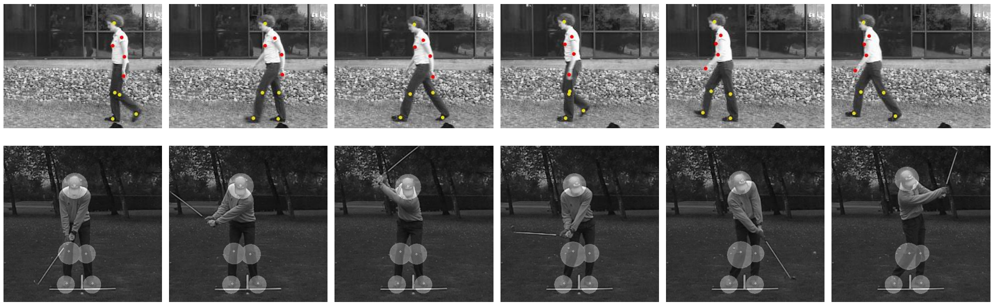


Figure 4.12: **2D Tracking using the WSL tracker. Top row:** Tracking the chest, knees, head, ankles and visible arm. The tracked upper body joints are shown in red, with the head and tracked lower joints points shown in yellow. **Bottom row**: Regions used for tracking the ankles, knees, and head are shown.

tracker. Fig. 4.12 shows the 2D tracking locations for two test sequences; we track 9 points for walking sequences, and 6 for the golf swing.

For joint j, we therefore obtain approximate 2–D positions $\mathbf{p^j}$ in each frame. From the target parameters ϕ we know the 3–D position of the corresponding joint. We then take the corresponding constraint function, $O^{joint}(\mathbf{p^j}, \phi)$, to be the 2–D Euclidean distance between the joint's projection into the image plane and the measurement of its 2-D image position.

4.3.2 Foreground and Background

Given an image of the background without the subject, we can extract rough binary masks (silhouettes) of the foreground, like those in Fig. 4.13. Because the background in our video is not truly static the masks are expected to be noisy. Nevertheless, they can be exploited as follows. We randomly sample the binary mask, and for each sample $\mathbf{p_i}$ we define a *Background/Foreground function* $O^{fg/bg}(\mathbf{p_i}, \phi)$ that is 0 if the line of sight through $\mathbf{x_i}$ intersects the model. Otherwise, it is equal to the 3D distance between the line of sight and the nearest model point. In other words, $O^{fg/bg}$ is a differentiable function that introduces a penalty for each point in the foreground binary mask that does *not* back-project onto the model.

Minimizing $O^{fg/bg}$ in the least squares sense tends to maximize the overlap between the model's projection and the foreground binary mask. This helps to prevent target drift.

4.3.3 Point Correspondences (Optical Flow)

We use 2–D point correspondences in pairs of consecutive images as an additional source of information: We project the 3–D model into the first image of the pair. We then sample image points to which the model projects and use a normalized cross-correlation algorithm to compute displacements of these points from that frame to the next. This provides us with measurement pairs of corresponding points in two consecutive frames, $\mathbf{p}_i = (\mathbf{p}_i^1, \mathbf{p}_i^2)$. The correspondence penalty function, $O^{corr}(\mathbf{p}_i, \phi)$ is given as follows: We back-project $\mathbf{p}_i^1$ to

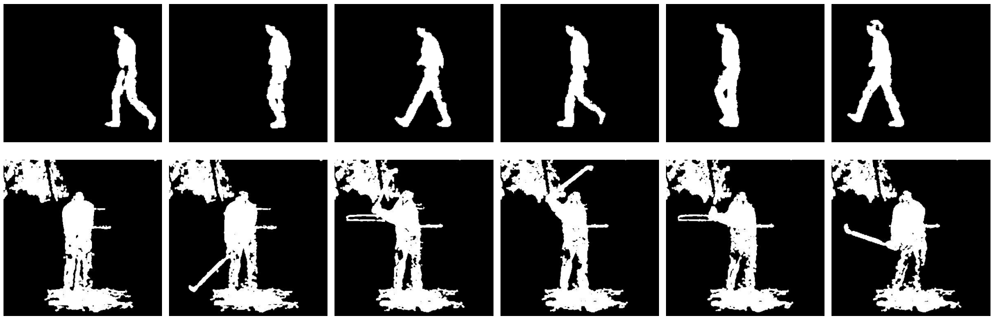


Figure 4.13: **Poor quality foreground binary mask. First row:** Extracted from the walking sequence of Fig. 4.1 and **Second row:** from the golf swing of Fig. 4.17.

the 3–D model surface and reproject it to the second image. We then take $O^{corr}(\mathbf{p}_i, \phi)$ to be the Euclidean distance in the image plane between this reprojection and corresponding $\mathbf{p}_i^2$.

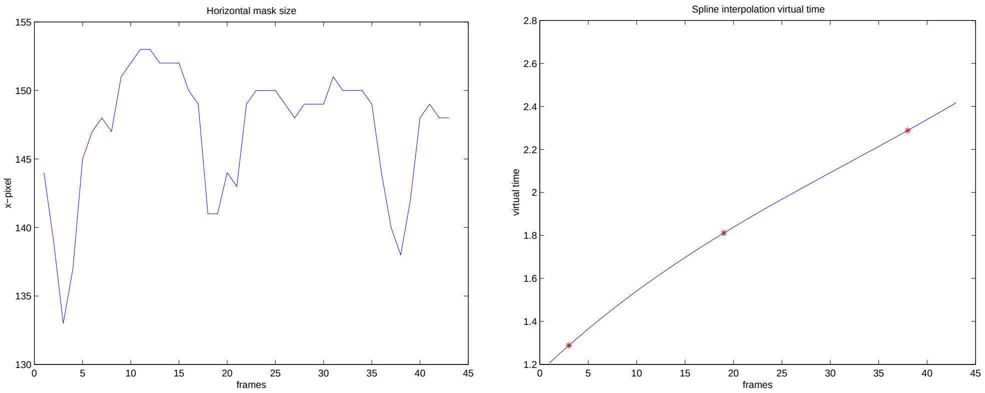
4.3.4 Experimental Results

The results shown here were obtained from uncalibrated images. The motions were performed by subjects of unknown sizes wearing ordinary clothes that are not particularly textured. To perform our computation, we used rough guesses for the subject sizes and for the intrinsic and extrinsic camera parameters. For each test sequence we manually initialize the position and orientation of the root node of the body in the first frame so that it projects approximately to the right place. Similarly we manually give the 2D locations of the joints to be tracked by WSL. This entire process requires only a few mouse clicks and could easily be improved with the use of automated posture detection techniques (e.g., [1, 3, 41, 99, 137]).

Simple methods were used to detect the key postures defined in Section 4.1 for each sequence. Using spline interpolation, we assign an initial value for μ_t for all the frames in the sequence, as depicted in Figs. 4.14(b) and 4.16(b). Finally, the motion is initially taken to be the mean motion Θ_0, i.e., the subspace coefficients α_i are initially set to zero. Given these initial conditions we minimize F_t in (4.9) using Levenberg-Marquardt.

4.3.4.1 Walking

Fig. 4.1 shows a well-known walking sequence [126, 127, 2]. To perform the 2D tracking we used a version of the WSL tracker [67]. WSL is a robust, motion-based 2D tracker that maintains an online adaptive appearance model. The model adapts to slowly changing image appearance with a natural measure of the temporal stability of the underlying image structure. By identifying stable properties of appearance the tracker can weight them more heavily for motion estimation, while less stable properties can be proportionately down-



(a) (b)

Figure 4.14: **Automatic Initialization of the normalized time** parameter μ_t for the walking sequence of Fig. 4.1. (a) Width of the detected silhouette. (b) Spline interpolation for the detected key-postures.

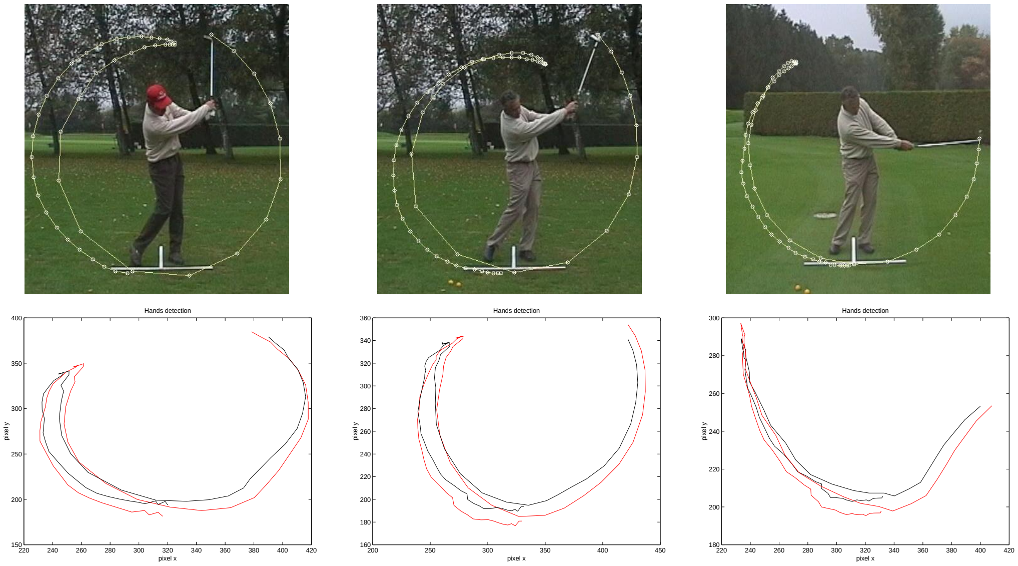
weighted. This gives it robustness to partial occlusions. In the first frame we specified 9 points that we wish to track, namely, the ankles, knees, chest, head, left shoulder, elbow and hand.

To initialize the phase parameter, μ_t, we used a simple background subtraction method to compute foreground masks (e.g., see Fig. 4.13). Times at which the mask width was minimal were taken to be the times at which the legs were together (i.e., $\mu_t = 0.25$ or $\mu_t = 0.75$). Spline interpolation was then used to approximate μ_t at all other frames in the sequence (see Fig. 4.14b). More sophisticated detectors [1, 41, 99, 137] would be necessary in more challenging situations, but were not needed here.

The optimal motion found is shown in Figure 4.1. There we show the estimated 3D model projected onto several frames of the sequence. We also show the rendered 3D volumetric model alone. The tracker was successful, producing a 3D motion that is plausible and well synchronized with the video. Note that even though the right (occluded) arm is not well reconstructed by the model (does not fit the image data), since it was not tracked by the WSL tracker, and hence was only weakly constrained by the objective function, it has a plausible rotation.

4.3.4.2 Golf Swing

As discussed in Section 4.1.3, the golf swings used to train the model were *full swings* from the CMU database. They were performed by neither of the golfers shown in Figs. 4.2, 4.17 and 4.18. With the WSL tracker we tracked five points on the body, namely, the knees, ankles and head (see Fig. 4.12). Because the hand tends to rotate during the motion, to track the wrists we have found it more effective to use a club tracking algorithm [82] that takes advantage of the information provided by the whole shaft. Its output is depicted by the



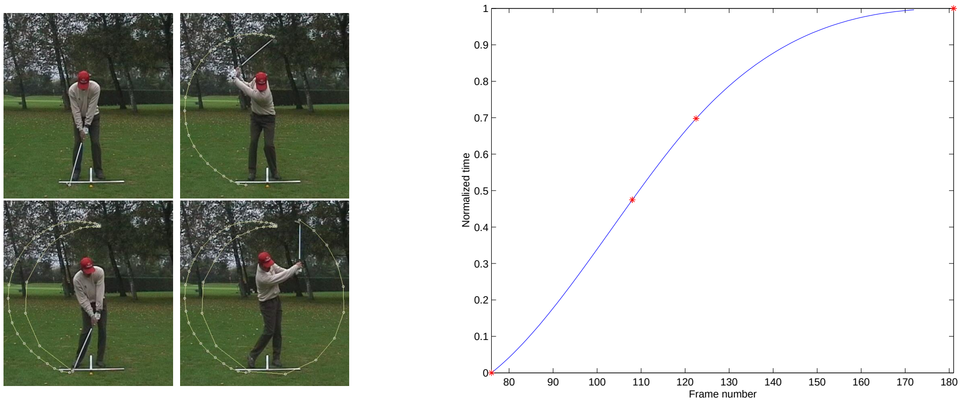
(a) (b) (c)

Figure 4.15: **Detected hand trajectories** for the *full* swing in Fig. 4.2 and the *approach* swing in Fig. 4.18. The left and right hand positions (pixel units) are represented in black and red respectively.

first row of Fig. 4.15, and the corresponding recovered hand trajectories by the second row. This tracker does not require any manual initialization. It is also robust to mis-detections and false alarms and has been validated on many sequences. Hypotheses on the position are first generated by detecting pairs of close parallel line segments in the frames, and then robustly fitting a 2D motion model over several frames simultaneously. From the recovered club motion, we can infer the 2D hand trajectories of the bottom row of Fig. 4.15.

For each sequence, we first run the golf club tracker [82]. As shown in Fig. 4.16(a), for each sequence, the detected club positions let us initialize the phase parameters by telling us in which four frames the key postures of Fig. 4.6 can be observed. With the times of the key postures, spline interpolation is then used to assign a phase to all other frames in the sequence (see Fig. 4.16(b)). As not everybody performs the motion at the same speed, this time is only a guess, which is refined during the actual optimization. Thus the temporal alignment does not need to be precise, but it gives a rough initialization for each frame.

Figures 4.2 and 4.17 show the projections of the recovered 3D skeleton in a representative subset of images of two *full* swings performed by subjects whose motion was not used in the motion database. Note the accuracy of the results. Fig. 4.18 depicts a *short* swing that is performed by a different person. Note that this motion is quite different both from the full swing motion of Fig. 4.2 and from the swing used to train the system. The club does not go as high and, as shown in Fig. 4.15, the hands travel a much shorter distance. As shown



(a) (b)

Figure 4.16: **Assigning normalized times** to the frames of Fig. 4.2. (a) We use the automatically detected club positions to identify the key postures of Fig. 4.6. (b) The corresponding normalized times are denoted by red dots. Spline interpolation is then used to initialize the μ_t parameter for all other frames in the sequence.

by the projection of the 3D skeleton, the system has enough flexibility to generalize to this motion. Note, however, that the right leg bends too much at the end of the swing, which is caused by the small number of training motions and the fact that every training swing exhibited the same anomaly. A natural way to avoid this problem in the future would be to use a larger training set with a greater variety of motions.

Finally, Fig. 4.19 helps to show that the model has sufficient flexibility to do the *wrong* thing given insufficient image data. That is, even though we use an activity-specific motion model, the problem is not so constrained that we are guaranteed to get valid postures or motions without using the image information correctly.

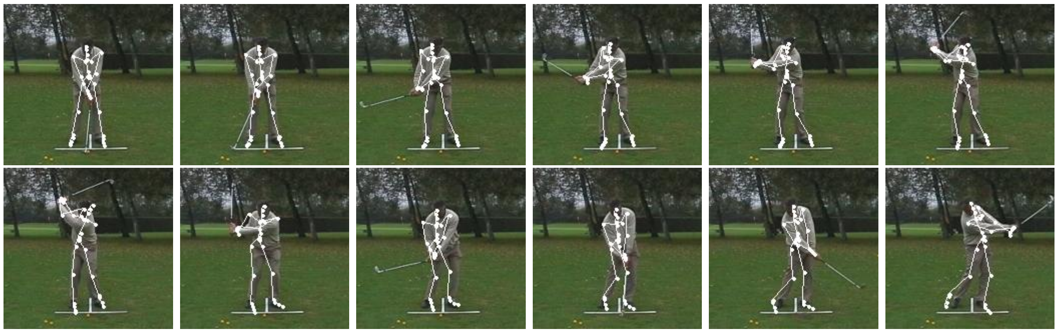


Figure 4.17: **Monocular tracking a full swing** of 68 frames. The skeleton of the recovered 3–D model is projected onto the images.

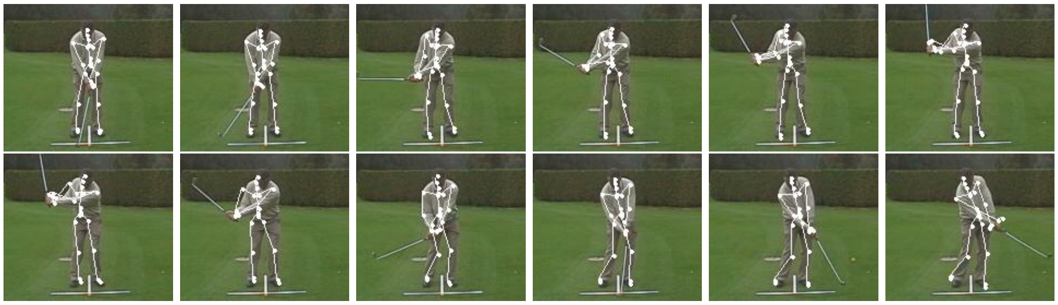


Figure 4.18: **Monocular Tracking a short swing** during which the club goes much less high than in a driving swing. The skeleton of the recovered 3–D model is projected onto the images.

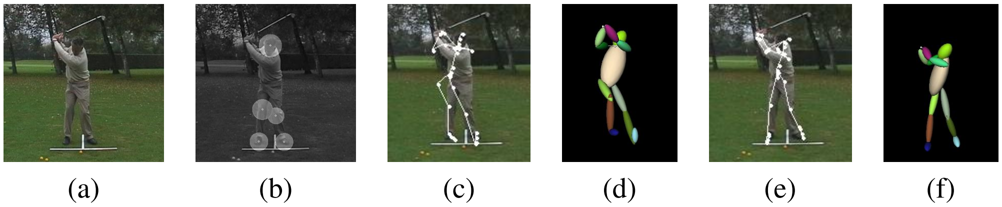


Figure 4.19: **Tracking using only joint constraints vs using the complete objective function**. (a) Original image. (b) 2–D appearance based tracking result. (c) 2–D projection of the tracking results using only joint constraints. The problem is under-constrained and a multiple set of solutions are possible. (d) 3–D tracking results using only joint constraints. (e) and (f) The set of solutions is reduced using correspondences.

4.4 Multi-view Tracking

When several synchronized video streams are available, we use a correlation-based stereo algorithm [152] to extract a cloud of 3–D points at each frame, to which we fit the motion model.

4.4.1 Objective Function

Recall from section 4.1 that we represent the human body as a set of volumetric primitives attached to an articulated 3–D skeleton. For multi-view tracking we treat them as implicit surfaces as this provides a differentiable objective function which can be fit to the 3D stereo data while ignoring measurement outliers. Following [111] the body is divided into several body parts; each body part b includes n_b ellipsoidal primitives attached to the skeleton. Associated with each primitive is a field function h_i, of the form

$$h_i(\mathbf{p}, \phi) = b_i \exp(-a_i d_i(\mathbf{p}, \phi)) \quad , \tag{4.10}$$

where $\mathbf{p}$ is a 3–D point, a_i, b_i are constant values, d_i is the algebraic distance to the center of the primitive, and ϕ, is the state vector in (4.6). The complete field function for body part b is taken to be

$$h^b(\mathbf{p}, \phi) = \sum_{i=1}^{n_b} h_i(\mathbf{p}, \phi) \quad , \tag{4.11}$$

and the skin is defined by the level set

$$\mathcal{S}K(\mathbf{p}, \phi) = \bigcup_{b=1}^{B} \{\mathbf{p} \in \mathbf{R}^3 | h^b(\mathbf{p}, \phi) = C\} \tag{4.12}$$

where C is a constant, and B is the total number of body parts. A 3D point $\mathbf{p}$ is said attached to body part b if

$$h^b(\mathbf{p}, \phi) = \min_{1 \le i \le B} |h^i(\mathbf{p}, \phi) - C| \tag{4.13}$$

For each 3D stereo point, $\mathbf{p}_i$, we write

$$O^{stereo}(\mathbf{p}_i, \phi) = h^b(\mathbf{p}_i, \phi) - C \ . \tag{4.14}$$

Fitting the model to stereo-data then amounts to minimizing (4.9), the first term of which becomes

$$\sum_{j=t}^{t+f-1} \sum_{b=1}^{B} \sum_{\mathbf{p}_i \in b} (h^b(\mathbf{p}_{i,j}, \phi) - C)^2 \ , \tag{4.15}$$

where $\mathbf{p}_{i,j}$ is a 3D stereo point belonging to frame j. Note that O^{stereo} is differentiable and its derivatives can be computed efficiently [111].

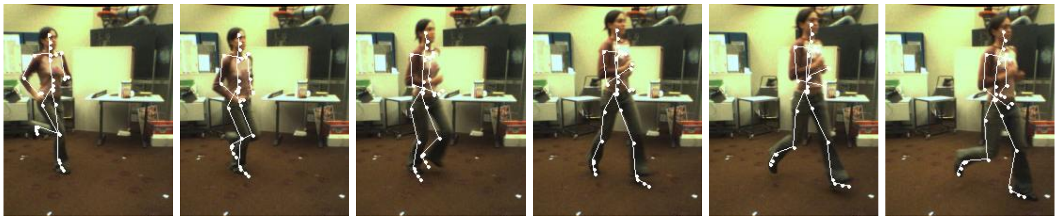


Figure 4.20: **Tracking a running motion**. The legs are now correctly positioned in the whole sequence.

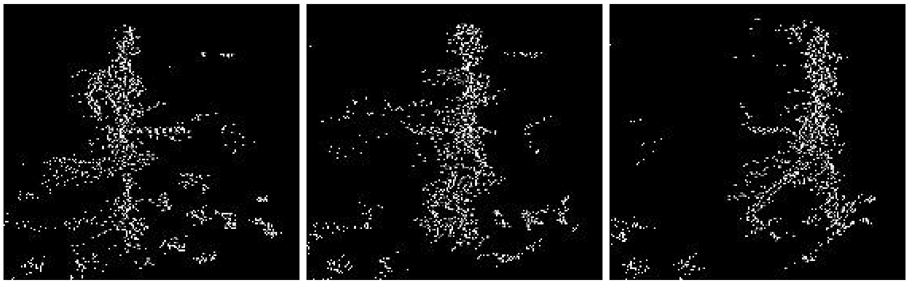


Figure 4.21: **Input stereo data** for the running sequence of Fig. 4.20. Side views of the 3–D points computed by the Digiclops ^{tm} system. Note that they are very noisy and lack depth because of the low quality of the video sequence.

4.4.2 Experimental Results

We use stereo data acquired using a Digiclops^{tm} operating at a 640×480 resolution and a 14Hz frame rate. Because the frame rate is slow, the running subject of Fig. 4.20 remains within the capture volume for only 6 frames. The data shown in Fig. 4.21 are noisy and have low resolution for two reasons. First, to avoid motion blur, we used a high shutter speed that reduced exposure. Second, because the camera was fixed and the subject had to remain within the capture volume, she projected onto a small region of the image during the sequences. Of course, the quality of this stereo data could have been improved by using more sophisticated equipment. Nevertheless, our results show that the tracker is robust enough to exploit data acquired with cheap sensors.

Initially, the motion subspace coefficients are set to zero, as above. We manually initialized the phase of the motion μ_t in the first and last frame of the sequence. These points were then interpolated to produce an initial phase estimate in every frame. The initial guess does not have to be precise because the tracking does not work directly with the images but with the 3D data.

Fig. 4.22 shows results on walking sequences performed by two subjects whose motion capture data were also used as training data for the motion models. One can see from the figures that the legs are correctly positioned. The errors in the upper-body are caused by the

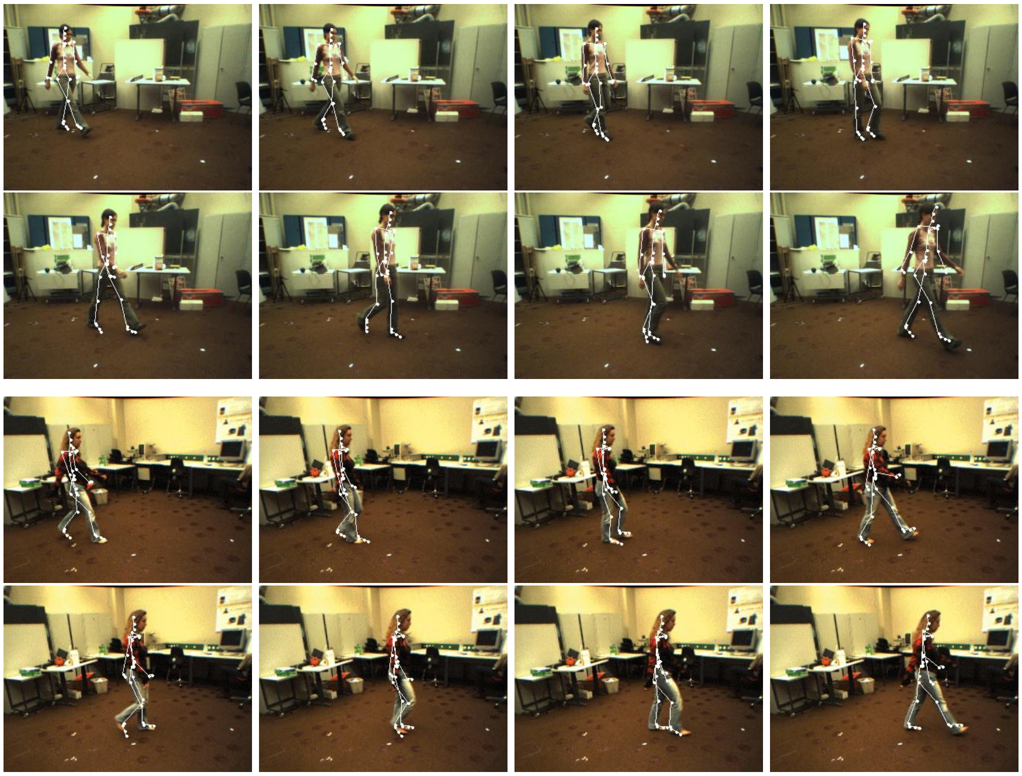


Figure 4.22: **Multi-view walking tracking**. Using low resolution stereo data to track the two women whose motions were not used to learn the motion model. The recovered skeleton poses are overlaid in white.

large amount of noise in the stereo data.

Figure 4.23 depicts results from a walking sequence with a subject whose motion was not included in the training data. In this case he was also wearing four gyroscopes on his legs, one for each sagittal rotation of the hip and knee joints. The angular speeds they measured were used solely for comparison purposes. Their output was integrated to yield the absolute angles shown as dotted curve in Fig. 4.24. We overlay on these plots the values recovered by our tracker, showing that they are close, even though the left leg is severely occluded. Given the position of the visible leg, the PCA motion model constrains the occluded one to be in a plausible position close to the real one.

Figure 4.20 shows results for the running sequence of Fig. 4.21 using the running motion model. The pose of the legs is correctly recovered. The upper body tracking remains relatively imprecise because average errors in the stereo data are larger than the distance between the torso and the arms. Improving this would require the use of additional information, such as silhouettes. Here we restrict ourselves to stereo data to show that our framework can be used with very different objective functions.

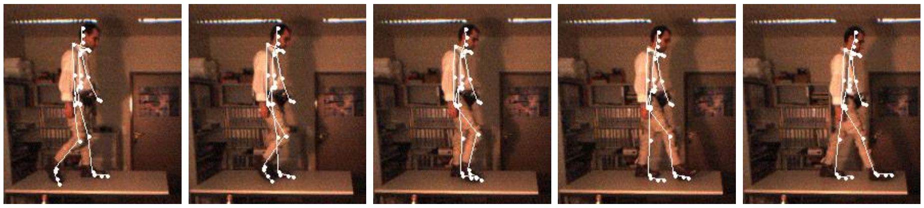


Figure 4.23: **Tracking a walking motion** from a subject whose motion was not recorded in the database. The legs are correctly positioned.

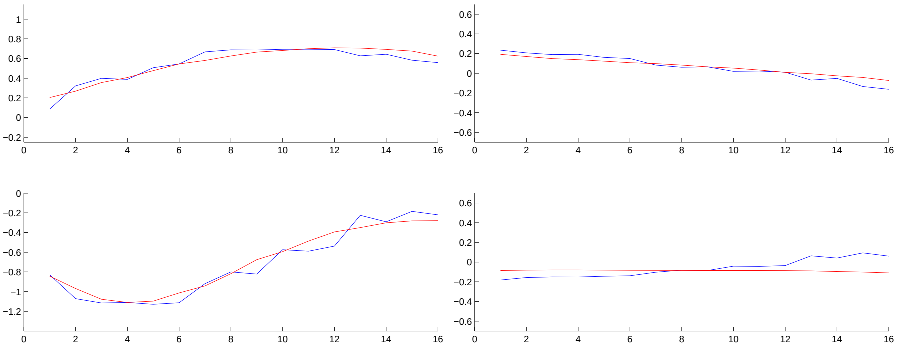


Figure 4.24: Comparing recovered rotation angles using visual tracking (solid curve), and by integrating gyroscopic data (smooth curve) for the walk of Fig. 4.23. **Left column**: Right hip and knee sagittal rotations. **Right Column**: Same thing for the left leg. Note that both curves are very close in all plots, even though the left leg is severely occluded.

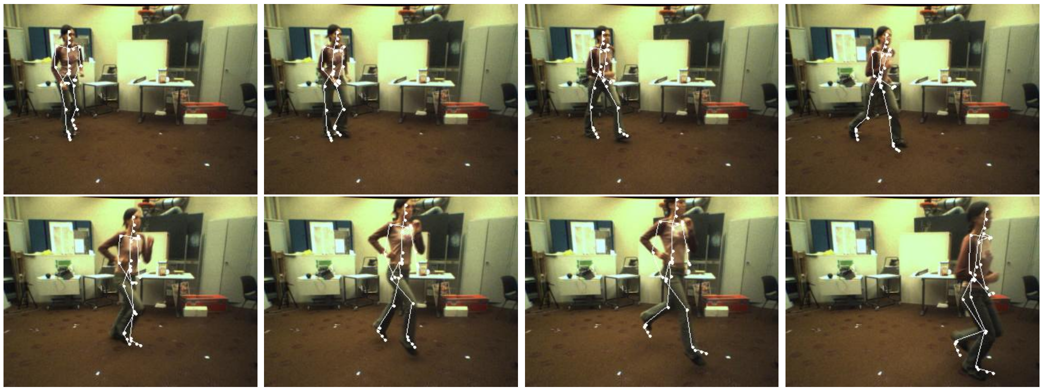


Figure 4.25: **Tracking the transition between walking and running.** In the first four frames the subject is running. The transition occurs in the following three frames and the sequence ends with running. The whole sequence is shown.

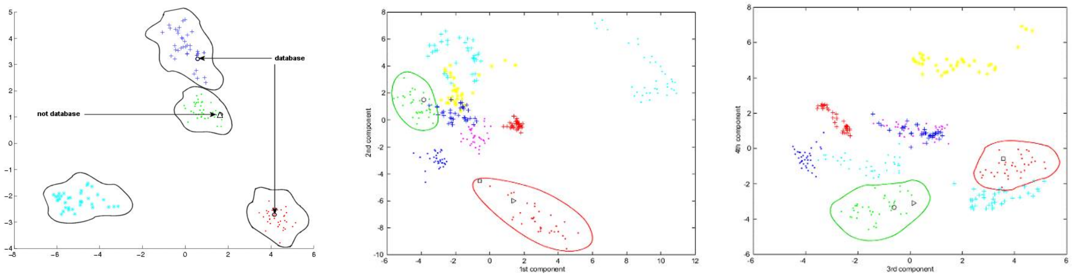
Having a set of subspace coefficients per frame gives the system the freedom to automatically evolve from one activity to another. To demonstrate this we used our motion model learned for the combined running and walking data to track a transition from walking to running (see Fig. 4.25). In the first few frames the subject is walking, then for a couple of frames she performs the transition and runs for the rest of the sequence. The arms are not tracked because we focus on estimating the motion parameters of the lower body only. Here again, the legs are successfully tracked with small errors in foot positioning that are due to the fact that ankle flexion is not part of the motion database.

4.4.3 Recognition

The motion style is encoded by the subspace coefficients in (4.4). They measure the deviation from the average motion along orthogonal directions. Recall that during tracking above, the subspace coefficients are permitted to vary from frame to frame. For recognition, we reconstruct the 3D motion of the person with a single set of subspace coefficients for the entire sequence [153].

In more detail, the tracking algorithm used for recognition is divided into two steps. First, the normalized time μ_t and the global motion $\mathbf{g}_t$ are optimized frame by frame, assuming a constant style equal to the mean motion Θ_0. This provides a good initial estimate for a second step, where a global optimization is performed. In the global fit, the normalized time and global motion parameters are allow to vary in every frame, but only one set subspace coefficients is used to represent the entire motion sequence. This is equivalent to minimizing (4.9), where the size of the sliding window is $\tau + 1 = T$.

Figure 4.26 (a) depicts the first two x_i coefficients for the database used for the tracking. The four subjects of the subspace are well separated in the first two dimensions. The estimated coefficients for each one of the two examples depicted by Fig. 4.22 are shown



(a) (b) (c)

Figure 4.26: **Recognition of the walking person from stereo data.** Different subjects appear in different colors and symbols. (a) Clusters formed by the first two PCA coefficients in the database used for tracking, composed of 4 subjects. Black circles represent tracking results for database subjects. The black triangle represent tracking result for a new subject. Note that when the database is conformed of this five subjects it does not cluster anymore in 2–D. (b) Fist two components of a bigger database than the one used for tracking, composed of 9 subjects. Black circles represent tracking results for database subjects and black triangles for a subject whose motions was not recorded when building the database. Note that the database clusters in 4–D. (c) Third and Fourth coefficients of the 9 subjects database.

as circles and a triangle represents the estimated value for the subject in Fig. 4.23 whose motion is not included in the training dataset. For both women, the first two recovered coefficients fall in the center of the cluster formed by their recorded motion vectors. Also note that while the new subject's motion does appear consistent with one of the training subjects in the first two subspace dimensions, they are quite different in the next two dimensions.

Figure 4.26 (b,c), depicts the first four x_i coefficients for a model learned using nine subjects. The estimated coefficients for each one of the two examples depicted in Fig. 4.22 are shown as circles and as triangles for the subject of Fig. 4.23 whose motion is not recorded in the database. Once more, for both women, the first four recovered coefficients fall in the center of the cluster formed by their recorded motion vectors using optical motion capture, meaning that they have been well estimated. Higher order coefficients exhibit small variations that can be attributed to the fact that walking on a treadmill changes the style. Typically the subjects tend to bend the back more when performing the walking in a treadmill to maintain balance. For the man whose motion was not recorded in the database, the recovered coefficients fall within two different clusters when looking at the first two coefficients or at the third and fourth, meaning that this person forms a different cluster in four dimensions. It is not recognized as none of the nine persons of the database.

4.5 Motion Generation

Representing motions as linear sums of principal components has become a widely accepted animation technique [5, 48, 84, 147]. These principal components are computed by motion capturing as many people as possible performing a specific activity, representing each motion as a temporally quantized vector of joint angles, and performing a Principal Component Analysis (PCA) [68] on the resulting database of motion vectors. Linear combinations of these vectors can then be considered as valid motions and used to produce new animations.

While powerful, the simplest version of this approach is not particularly well suited to modeling the specific style of an individual whose motion had not yet been recorded when building the database: It would take an expert to adjust the PCA weights to obtain a motion style that is indistinguishable from his. Consequently, when realism is required, the current practice is to perform a full motion capture session each time a new person must be considered.

In this section, we show that the PCA approach can be extended so that this requirement can be drastically reduced: For whole classes of cyclic and non-cyclic motions such as walking, running or jumping, it is enough to observe the newcomer walking or running only once at a particular speed or jumping a particular distance using either an optical motion capture system or a simple pair of synchronized video cameras. This one observation is used to compute a set of principal component weights that best approximates the motion and to extrapolate in real-time realistic animations of the same person moving at different speeds or jumping at different distances. This has an important advantage over traditional blending approaches that simply rely on a linear combination of the captured data to create new styles [48, 74, 119]: Extrapolation allows us to reach a comparatively larger subspace of physically correct motions. Furthermore, unlike techniques such as the one described in [18], our approach does not need fine parameter settings for initialization purposes. Our animations are produced in real-time, with potential changes of physical motion properties such as walking speed or jumping distance.

We first validated our approach for both cyclical and non-cyclical motions by exclusively using reliable optical motion capture data: We built a walking database by capturing nine people walking at speeds ranging from 3 to 7 km/h, a running database by capturing five people running at speeds ranging from to 6 to 12 km/h, and a jumping database by capturing jumps ranging from 40 to 120 cm in length for four different people. Given a PCA decomposition and a new captured motion that had not been used to perform this decomposition, we project it into PCA space and compute distances, that weights the influence of each PCA component, to database motions corresponding to the same speed or jump length. These can then be used to synthesize new motions corresponding to different speeds or jumping lengths and we have verified that these synthesized motions and the actual ones that we have also recorded are both statistically and visually close.

We then replaced the optical motion capture data for the new person by stereo imagery acquired with a cheap and commercially available device [35]. To this end, we take advantage of the technique described in section 4.2 to estimate the PCA weights from the video sequences. This step replaces the projection into PCA space discussed above and allows us again to speed-up or slow-down the motion while preserving the style.

Note that, even though the database we used for validation purposes is specific to three kinds of motions, the approach itself is completely generic. Transposed onto a production set, it has great labor saving potential: The actors' motion need only be captured once to generate a whole range of realistic and personalized animations, thus sparing the need for time-consuming motion capture sessions and expensive gear.

In the remainder of this section, we will first discuss briefly related work on motion generation and show our approach to computing PCA weights for observed motions that are not part of the initial motion database and using them to extrapolate new ones. Finally, we will validate it using both optical motion capture and video data.

4.5.1 Related Work on Motion Generation

We include here a related work section, since in chapter 3 we have discuss only the state of the art in tracking, and not in motion generation since it is not the goal of this work but an application of our motion models.

The literature on walking and running animation is so rich that a full chapter would be necessary to discuss the advances for walking alone since the last major review of the field [101]. Three main classes of approach can nevertheless be distinguished.

Inverse kinematics. This involves specifying at each key time the corresponding key positions of some joints and obtaining the joint angles according to biomechanical data information. This can be done efficiently [9, 47] but there is no guarantee of physical realism and this often leads to overly mechanical movements.

Inverse Dynamics. These techniques look for the correct forces and torques to apply to joints to reach a given position. This produces smooth results but may involve postures that are not humanly feasible. It therefore becomes necessary to check and potentially correct these postures by applying appropriate constraints [59, 86, 85, 163].

Motion Capture and Editing. New motions are typically created by blending and interpolation. Composite motions can then be obtained by combining several captured ones. Comparable methods presented in [6, 75, 80, 148] do this by connecting them into a directed graph. Its edges represent motion clips and nodes are potential connecting points between clips. The user can generate new motions by moving along an optimized path in the graph.

To synthesize a motion that closely resembles that of a specific person, as opposed to a generic virtual human, using motion capture data is clearly the favored approach because there is no easy way to set the parameters for either Inverse Kinematics or Inverse Dynamics to achieve the desired goal. The latter class of techniques is therefore the most widely used. However, they are not usually designed to allow the modification of intrinsic properties of the database motions, which is the issue we address in this section and discuss in more details below.

4.5.1.1 Motion Editing

Constraint-based techniques, discussed and classified in [51], alter an original motion while preserving some specific geometric features. Between them, space-time constraint [50, 81] or physically-based approaches [113, 86] provide effective tools to interactively manipulate a motion clip by changing some important properties of the movement. While performing almost in real-time, these methods are appropriate for slight modifications of the motion but not to introduce stylistic variations, mainly because they are difficult to formulate as mathematical constraints.

A motion can be treated as a time-varying signal. Signal processing techniques have therefore been developed both to edit the complete motion by varying the frequency bands of the signal [20], or the Fourier coefficients [148], and to randomize the original motion [109]. However, controlling the randomization is far from straightforward and may yield unpredictable results that can be physically impossible.

Blending or interpolation are the typical approaches to generating new motions. For example, Ashraf and Wong [52] interpolate walking and running motions in 3D space where the axes correspond to significant parameters of a locomotion cycle. The method uses bilinear interpolation to synthesize a new motion given four motions having different values for each of the three dimensions. This approach, however, is limited to a small number of input motions and parameters.

Multivariate interpolation can be used to solve this problem. In [119], multivariate interpolation is performed on a wide range of motion capture data. Motions are defined by B-Spline coefficients and manually classified according to their characteristics, which yields a parameter vector for each motion. Similar motions are time-normalized using a time warping process that structurally aligns them. New ones are then generated by applying polynomial and Radial Basis Function interpolation between the B-Spline coefficients. Kovar and Gleicher [74] present a general method allowing motions generation from a various input motions. They introduce registration curves that ensure a consistent time-warping and root alignment and apply physical constraints to produce blended motions.

While powerful, these methods are highly dependent on the weights values the animators assign to a set of input motions. Thus, determining a good combination of these weights becomes difficult for the creation of a very specific motion. To enhance this control, a possible alternative is presented in [37]. A motion is modified interactively by an animator manipulating a reflective device whose motions are captured by an optical system and transferred to a virtual character.

The approaches presented above have the disadvantages that the newly generated motion cannot be retargeted to subjects of different sizes. Park et al. [107] propose a locomotion generation, adaptable to any target character, based on the motion interpolation of [119, 129]. A motion retargeting based on the approach introduced by Shin et al. [124] provides a real-time animation framework. However, even if stylistic variations were incorporated into this approach, generating the motions corresponding to a specific person will involve the same problems as before.

4.5.1.2 Principal Component Analysis for motion synthesis

The methods discussed above suffer from a number of limitations: First, there is no intuitive way to create a motion with specific characteristics. Second, they do not provide for extrapolation. As a result, to create a whole range of motions such as those of a specific athlete running at varying speeds, one must perform a full motion capture session of that athlete actually running at a number of different speeds. This can prove cumbersome and the technique we advocate in this section takes advantage of Principal Components Analysis (PCA) to alleviate this problem by giving our system an extrapolation capability.

PCA [14, 84, 147] has recently been extensively used in motion synthesis. It has also been used to compress keyframed animation data [5] and to emphasize similarities between instances of objects such as heads [15, 36] to deform them for example by changing their apparent age or gender. Unfortunately for walking, running and jumping motions, the PCA weights have no obvious direct interpretation. More sophisticated blending techniques are required. Glardon et al. [48] used hierarchical structures to isolate specific motion parameters. It can be used to extrapolate new motions but is not designed to reproduce the specific style of an individual whose motions are not in the motion database. In theory, it could be modeled as a weighted sum of database motions, which would require finding the right weights. However, in practice, finding the weights by hand is very difficult and that is precisely what the technique proposed in this section lets us do from a single example for each new style.

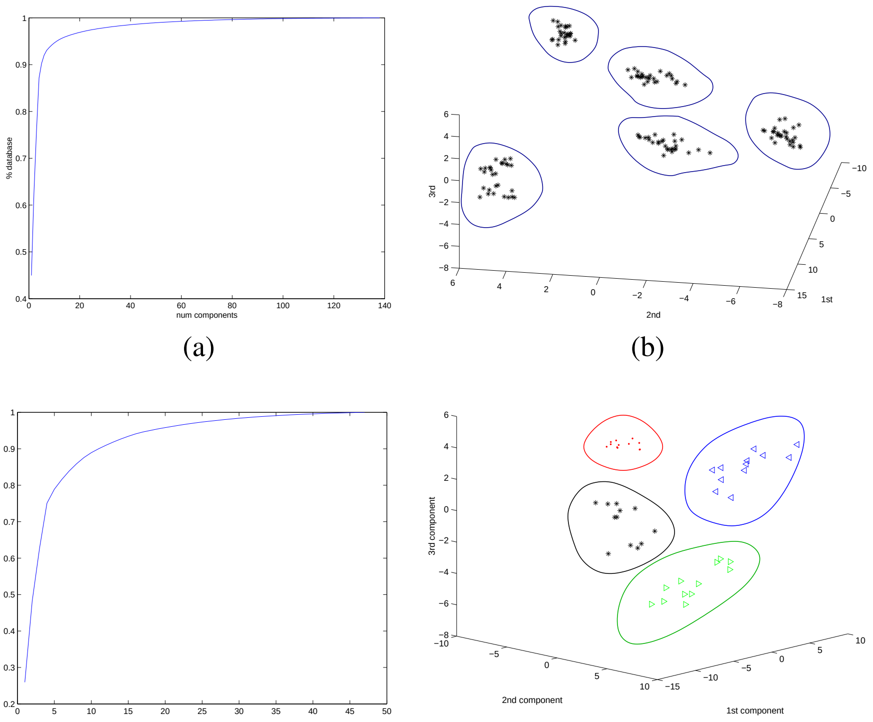
While our intention is quite similar to [18], by automatically extrapolating stylistic animations, we enhance the motion creation process by offering a separate control of its physical parameters (speed for walking and running, length for jumping) and the ability to retarget the motion to virtual humans of different sizes. Moreover, we provide a mathematical framework that does not depend on a specific parameterization and fulfills real-time constraints.

4.5.2 Models for Motion Synthesis and Analysis

In this section, we introduce the motion models we use both to synthesize walking, running and jumping animations and to capture such motions from video sequences.

4.5.2.1 Walking and Running

We use bigger walking and running databases than the ones used for tracking, sampled at a higher frequency. To build walking models, we used a Vicon optical motion capture system [155] to capture nine people walking at speeds ranging from 3 to 7 km/h by increments of 0.5 km/h on a treadmill. Similarly, to build a running model, we captured five people running at speeds ranging from 6 to 12 km/h by increments of 1 km/h. The data was segmented into cycles and sampled at regular time intervals using quaternion spherical interpolation [125] so that each example can be treated as $M = 100$ samples of a motion.



(a) (b)

(c) (d)

Figure 4.27: **The motion database.** (a) Percentage of the database that can be generated with a given number of eigenvalues for the running database. (b) Clustering behavior of the first 3 x_i coefficients of Eq. 4.4 for the 140 motion vectors measured for 5 subjects running at speeds ranging from 6 to 12 km/h. They form relatively compact clusters in 3–D space that can be used for recognition purposes. c) Percentage of the database that can be generated with a given number of eigenvalues for the jumping database. d) Clustering behavior of the first 3 components for 48 motions vectors measured for 4 subjects jumping distances ranging from 40 to 120 cm. They cluster in 3–D.

As different subjects may be shorter or taller, size normalization is required. Murray [102] has shown that, for adults, relative angles for the hip, knee and ankle in the sagittal plane have very similar trajectories for the same value of normalized speed V, obtained by dividing the walking velocity v by the hip joint height H that represents the leg length. Glardon et al. [48] generalized this approach to running and jumping. We normalize the training motions by dividing the translation values of the humanoid root by the leg length of the captured subject. This process produces $N = 324$ (9 subjects, 9 speeds, 4 cycles) angular motion vectors for walking and 140 (5 subjects, 7 speeds, 4 cycles) for running.

Fig. 4.27 (a) depicts $Q(d)$ as a function of d for the runnning database. It is taken to be 0.9 for the walking and 0.95 for the running databases, given $d \simeq 10$.

The top row of Fig. 4.27 depicts the first three x_i components of the original running motion vectors when expressed in terms of the Θ_i eigenvectors. Note that the vectors corresponding to specific subjects tend to cluster. The walking database exhibits the same clustering behavior but in higher dimension as depicted in Fig. 4.26 (b,c). This is due to the fact that the inter-variability between subjects is smaller than for a running motion. This was expected since the number of subjects that compose the walking database is higher than in the ones of the running database.

4.5.2.2 Jumping

To build the jumping database, we also used a Vicon optical motion capture system to capture four people jumping distances ranging from 40 to 120 cm by increments of 40 cm. The data was segmented into key-events, such as start and end of the jump, and sampled at regular time intervals using spherical interpolation, so that each example has $M = 100$ samples. The same procedure as in the case of the walking and running motions can then be applied with $N = 48$ (4 subjects, 3 jumps, 4 trials). The vectors of the jumping database corresponding to specific subjects tend also to cluster, validating the proposed approach, as depicted in the bottom row of Figure 4.27.

4.5.3 Motion Generation

In this subsection, we show how to extrapolate from a motion that is captured after the motion database of Section 4.5.2 has been built. This is a two-step process: First, we project the new motion into the PCA space and measure its distance to each recorded motion corresponding to the same speed or jump length. The generated motion is then taken to be a weighted average of motions at the target speed with the weights being inversely proportional to those distances.

More precisely, let $\mathbf{y}^{p,s}$ be the motion vector of Eq. 4.4 corresponding to database person p, where s represents either the speed or the jump length. In the remainder of the section we will refer to s as the *motion parameter*. Each one of these vectors can be approximated by its projection in PCA space $\mathbf{y}^{\hat{p},s}$ computed as

$$\begin{aligned}\hat{\mathbf{y}}^{p,s} &= \Theta_0 + \sum_i x_i^{p,s}\Theta_i \ , &(4.16)\\ x_i^{p,s} &= (\mathbf{y}^{p,s} - \Theta_0)\cdot\Theta_i \ , &(4.17)\end{aligned}$$

where the Θ_i are the principal component vectors of Eq. 4.4, and the $x_i^{p,s}$ are the scalar coefficients that characterize each motion.

Let Ψ^{x,s_1} be a vector characterized by motion parameter s, corresponding to a motion performed by person x who has not been captured before. The length is arbitrary and we wish to extrapolate motion Ψ^{x,s_2} of the same person moving with motion parameter $s_2 \neq s_1$ from it. As before, if the motion is cyclic we break Ψ^{x,s_1} into cycles and perform quaternion spherical interpolation [125] to produce a set of $\mathbf{y}^{x,s_1}$ motion vectors of the same dimension as the principal component vectors. If the motion is non-cyclic, we identify the start and end of the key-events of the motion and use the resulting vector of taking the frames in between such key-events in the same way as a cycle in a cyclic motion. By projecting one of these cycles into PCA space, we can the corresponding latent variable $\mathbf{x}^{x,s_1}$ analogous to the one of Eq. 4.17.

Because the influence of each Θ_i principal component vector is proportional to the corresponding λ_i eigenvalue, we use the normalized distance of Eq. 4.18 instead of the Euclidean distance to compare motion vectors. For each p in the database, we therefore take the distance between $\mathbf{y}^{x,s_1}$ and $\mathbf{y}^{p,s_1}$ to be

$$d_m(\mathbf{y}^{x,s_1}, \mathbf{y}^{p,s_1}) = \sqrt{\frac{\sum_i \lambda_i (x_i^{x,s_1} - x_i^{p,s_1})^2}{\sum_i \lambda_i}} \ . \tag{4.18}$$

Kovar [75] defined a more realistic distance function in terms of the distances between meshes. However the cost of using such distance is prohibitive in our context since we would need to evaluate $M * S$ distances between meshes, where S is the total number of subjects and M the number of frames. The distance proposed in this section requires only $d * S$ norm evaluations, where d is the number of eigenvalues (dimensionality of the latent space), allowing a real-time animation of multiple subjects. The only preprocessing needed is the generation of the PCA database, which takes a few seconds.

The weights are then taken to be the normalized inverse of these distances.

$$w^{x,p} = \frac{[d_m(\mathbf{y}^{x,s_1}, \mathbf{y}^{p,s_1})]^{-1}}{\sum_q [d_m(\mathbf{y}^{x,s_1}, \mathbf{y}^{q,s_1})]^{-1}} \tag{4.19}$$

This completes the interpolation step of our motion synthesis scheme and we are now ready to generate a new motion. If s_2 is one of the motion parameters recorded in the database, we can simply take the new motion $\widetilde{\mathbf{y}}_{s_1}^{x,s_2}$ to be an extrapolation of $\mathbf{y}^{x,s_1}$:

$$\begin{aligned}\widetilde{\mathbf{y}}_{s_1}^{x,s_2} &= \Theta_0 + \sum_i \widetilde{x}_i^{x,s_2}\Theta_i &(4.20)\\ \widetilde{x}_i^{x,s_2} &= \sum_p w^{x,p} x_i^{p,s_2} \ .\end{aligned}$$

Otherwise, to produce smooth transitions between motion parameters, we take the $\widetilde{x}_i^{x,s_2}$ coefficients to be Cubic Spline Interpolations of those computed as described above.

4.5.4 Validation and Results

In this section we use a *cross-validation* framework to validate statistically and visually our approach. For each database subject p in turn, we remove all the $\mathbf{y}^{p,s}$ motion vectors that correspond to him or her and perform a new PCA. For any two motion parameters s_1 and s_2, we can then use the procedure of Section 4.5.3 to synthesize $\widetilde{\mathbf{y}}_{s_1}^{p,s_2}$ from $\mathbf{y}^{p,s_1}$ and compare it to $\mathbf{y}^{\widehat{p,s_2}}$, the actual projection of the recorded motion. Ideally, the Mahalanobis distances of these two motions should be zero for all p, s_1 and s_2.

In practice, this can of course never be exactly true. As discussed in Section 4.5.2, recall that the database contains, for each subject, several motion cycles at the same speed and that they are never exactly similar to one another. We then show how our technique is capable of extrapolating walking styles completely different from the standard ones use for training. Finally examples of extrapolation from video are shown.

4.5.4.1 Animation Results

We now show running and walking animation results obtained by synthetically varying the speed of a motion captured at *one* single speed. These animations are visualized using the time and space normalization described in [48], which allows smooth transitions between motions and adaptation to different human sizes.

To perform size normalization, the root node translation is simply multiplied by the hip joint height H_i of the subject to animate. Recall from section 4.5.2.1 that all input motions have been re-sampled to a fixed number of frames, N. A time normalization stage establishes a correspondence between the elapsed time Δt and the normalized time μ_t, $0 \leq \mu_t \leq 1$ of Eq. 4.3. Given C, the cycle frequency, defined as an adapted version of the Inman law [48], the current normalized time μ_t is used to defined the frame j to display as:

$$\mu_t = \mu_{t-1} + \Delta t C \tag{4.21}$$

$$j = \mu_t N. \tag{4.22}$$

Figure 4.29 depicts running at speeds increasing from 6 to 12 km/h. For comparison purposes, we superpose the results with an animation obtained by interpolating the actual motion capture data at *all* the relevant speeds. Note that the two synthetic characters are superposed almost perfectly. To highlight the quality of the results, in the bottom of Figure 4.29, we superpose two animations corresponding to two *different* women. There the differences are obvious. The same phenomenon can be seen in Figure 4.28 where we plot the alpha coefficients as a function of speed.

Figure 4.30 depicts a similar behavior for walking at speeds ranging from 3 to 7 km/h. Again, as can be seen in the top row, the motions generated from a single example and those interpolated using a whole set of examples match very well, except for small discrepancies

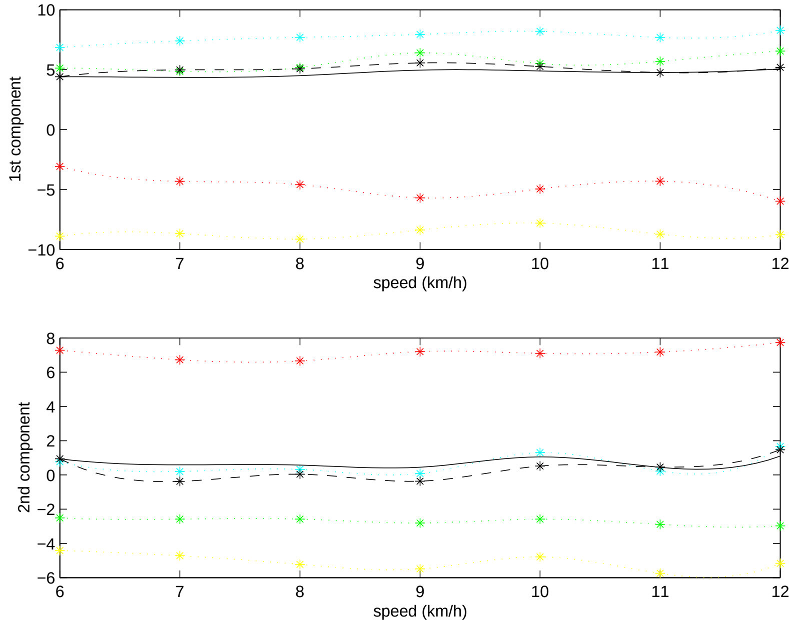


Figure 4.28: **Cubic Spline interpolations** of the first two components as a function of the speed for the running database. The synthesized motion depicted in Figure 4.29 is generated from a single optical motion at 6 km/h and is shown in solid black. The original captured motion is depicted in dashed black while the database subjects are shown in different dotted colors.

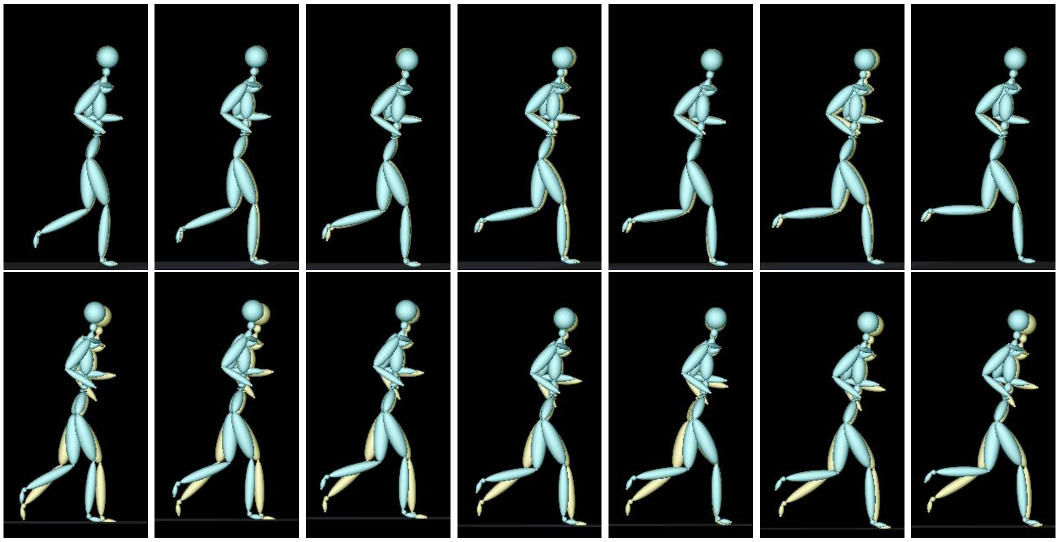


Figure 4.29: **Running at speeds increasing from 6 to 12 km/h. Top row:** Superposition of the synthesized motion generated from a *single* optical motion capture at 6 km/h, in yellow, to an animation observed in the actual motion capture data at *all* the relevant speeds, in blue. Note that the two synthesized characters are superposed almost perfectly. **Bottom row:** Superposition of the animations corresponding to *two* different women. Note the big differences compared to the results in the top row.

of the arm motion. As will be discussed below, this can be ascribed to the fact that people do not perform a motion twice in exactly the same fashion.

The same principle can then be applied to jumping motion, but instead of parameterizing as a function of the speed, we parametrize as a function of the jump length. Figure 4.31 depicts two jumps of 40 and 120 cm generated from a single example of 80 cm. Note that the interpolated results and the original sequence again superpose very well. Once more small discrepancies appear at the arm level. Figure 4.32 depicts the entire sequence of a 120 cm jump extrapolated from an original 80 cm one. Note that both superpose well during the whole sequence.

The physical correctness of the results could be further improved by correcting the output animations by using standard inverse kinematic techniques to avoid the foot penetration into the floor or sliding effects.

4.5.4.2 Statistical Validation

We now introduce the statistical cross-validation framework we use to validate the results shown above. To this end, we define the following quantitative measures:

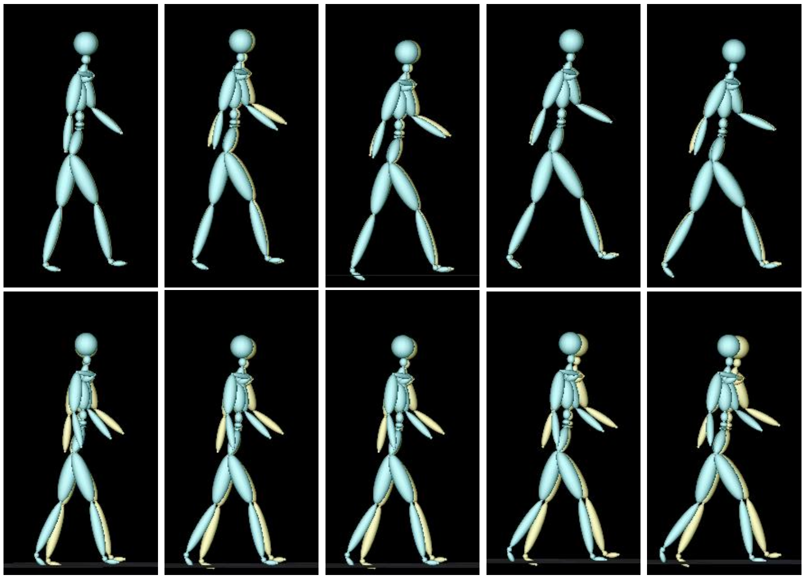


Figure 4.30: **Walking at speeds ranging from 3 to 7 km/h. Top row:** Superposition of the motions generated from a *single* example, in yellow, to those observed in the optical motion capture, in blue. They matched very well except from small discrepancies of the arm motion, that can be ascribed to the fact that people do not perform a motion twice exactly in the same fashion. **Bottom row:** Superposition of the animation corresponding to two different subjects. Note the big differences compared to the results in the top row. Note that the motion captured has not been corrected. This is the reason why the left leg is penetrating into the floor.

- **Interpolation Error:** The average over all subjects and pairs of speeds of the normalized distance, d_m of Eq. 4.18, between the recorded and synthesized motion vectors discussed above.
- **Intra-variability:** The dispersion between different realization by the same subject of the same motion at the same speed. It is taken to be the mean over all subjects and all speeds of the distance of the $\hat{\mathbf{y}}^{p,s}$ motion vectors corresponding to different cycles.
- **Inter-variability:** The dispersion between different clusters belonging to different subjects. It is calculated as the mean over all subjects of the distance between motion vectors corresponding to different cycles and speeds.

In Figure 4.33, we give the Interpolation Errors, Intra-variability and Inter-variability values for our walking, running and jumping databases. Because we perform cross-validation, we take each person out of the databases in turn and therefore list as many values as there

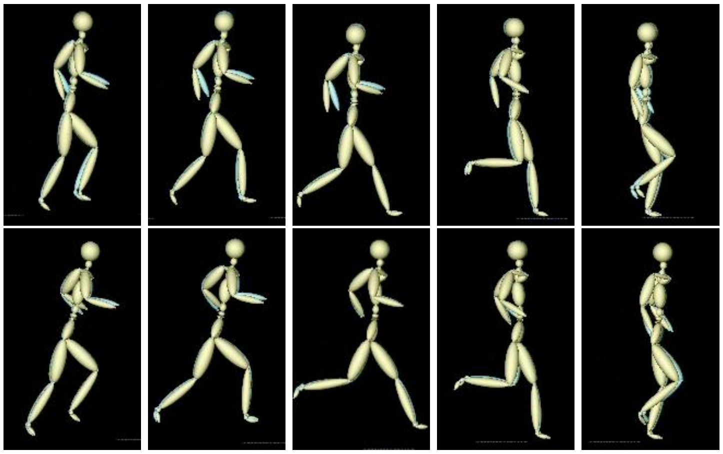


Figure 4.31: **Motion generation from a jump of length 80 cm. Top row:** 40 cm jump. **Bottom row:** 120 cm jump. We superpose motions generated from a *single* example (80 cm), in yellow, to those observed by the optical motion capture, in blue. They match very well except for small discrepancies in the arms.

are subjects in each one. Note that the interpolation error is consistently larger than the intra-variability but much smaller than the inter-variability. In other words, our motion generation scheme, while not perfect, nevertheless yields motions that are close enough to those of their rightful owner to be associated with him or her rather to anybody else.

The inter-variability of jumping is bigger than the one of the walking or running because it is more difficult for a subject to perform a jump twice in the same fashion, and to control the jump length while behaving normally. The interpolation error of the walking database is smaller than the others because the number of subjects in larger. This is also the reason why the inter-variability is smaller: The larger the number of subjects the smaller the distance between clusters.

4.5.4.3 Stylistic extrapolation

The motion generation technique presented in Section 4.5.3 can now be used to generate walking styles completely different from the standard ones that form the walking database. The principle is the same as before: The new stylized motion is projected into the motion database. Weights are then computed based on the distance of Eq. 4.18 and used to create the same style at a different speed. Figure 4.34 depicts a sneaking motion at 7 km/h generated by using a single example at 4.5 km/h and the standard walking database. Note that the subject bends her back and increases the step size when the speed increases, which is very realistic since that is what humans do while accelerating the motion in order to keep their balance.

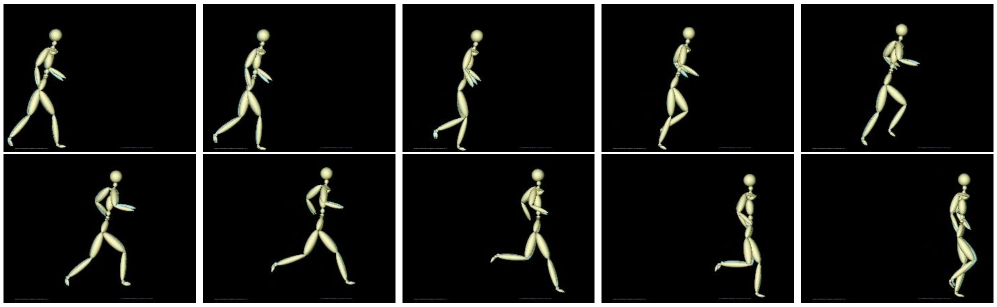


Figure 4.32: Superposition of the 120 cm jumping motion generated from a *single* example of length 80 cm, in yellow, to those observed by the optical motion capture, in blue. **Top row:** Frames 1-40. **Bottom row:** Frames 41-80.

4.5.4.4 From Video to Animation

In this subsection we show that we can replace the optical motion capture data we have used so far by synchronized video-sequences acquired using an inexpensive commercial product [35]. The PCA coefficients extracted from the images are accurate enough to produce valid and realistic motions used as input by our motion generation scheme.

The set of coefficients recovered by the tracking used for recognition and discuss in Section 4.4.3 is used to generate animations of the subjects shown in the first two rows of Figure 4.22 and 4.23 walking at different speeds.

Fig. 4.35 depicts a side view of the optical motions captured from 3 to 7 km/h (yellow) superposed on the synthesized motion extrapolated from the video-sequence of Fig. 4.22 where the person was walking at a speed of 5 km/h. For the legs the correspondence is almost perfect. The small differences in the arms stem from three different reasons: the intra-variability of the walking motion, the low resolution of the images in which the arms contain less than 10 pixels, and the fact that the optically captured motions were performed using a treadmill which is not the case for the video-sequence. Note that the motion capture data has not been corrected, which is why the left leg is penetrating into the floor.

Finally, we show how our framework can generate movements from a large space by tracking a subject that is not part of the database depicted in Fig. 4.23 at 3 km/h and by synthesizing his movement from 3 to 7km/h. The motion remains natural and physically possible even though it is *different* from all the recorded motions. In Fig. 4.36, we highlight this difference by superposing the generated motion to the closest one in the database, which is clearly dissimilar. This was to be expected because, as shown in Fig. 4.26, the corresponding alpha coefficients do not match any of the clusters that correspond to specific individuals.

Running	1	2	3	4	5
Intra Variability	0.22	0.21	0.22	0.19	0.20
Inter Variability	3.17	2.45	3.19	3.05	3.97
Interp Error	0.38	0.78	0.92	0.34	0.51

Walking	1	2	3	4	5	6	7	8	9
Intra Variability	0.24	0.21	0.21	0.22	0.23	0.23	0.24	0.23	0.20
Inter Variability	1.17	1.16	1.10	1.22	1.13	1.24	1.20	1.19	1.13
Interp Error	0.35	0.62	0.76	0.47	0.35	0.63	0.49	0.55	0.60

Jumping	1	2	3	4
Intra Variability	0.50	0.63	0.52	0.63
Inter Variability	1.82	1.88	1.73	1.97
Interp Error	0.93	0.73	0.72	0.87

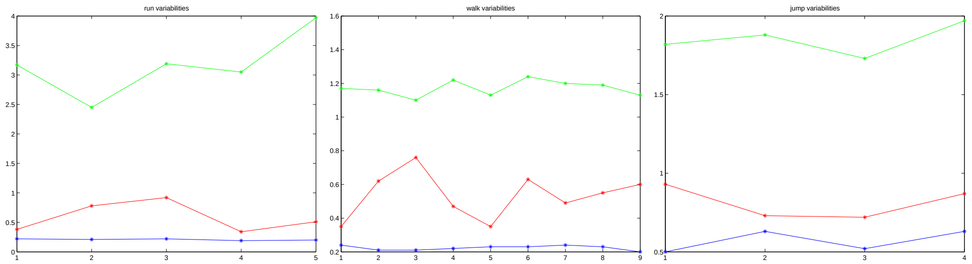


Figure 4.33: **Interpolation Errors**, Intra-variability and Inter-variability values. **Top row:** Running. **Second row:** Walking. **Third row:** Jumping. **Bottom row:** Graphic depiction of the above tables. Note that the interpolation error is consistently larger than the intra-variability but much smaller than the inter-variability in the three databases.

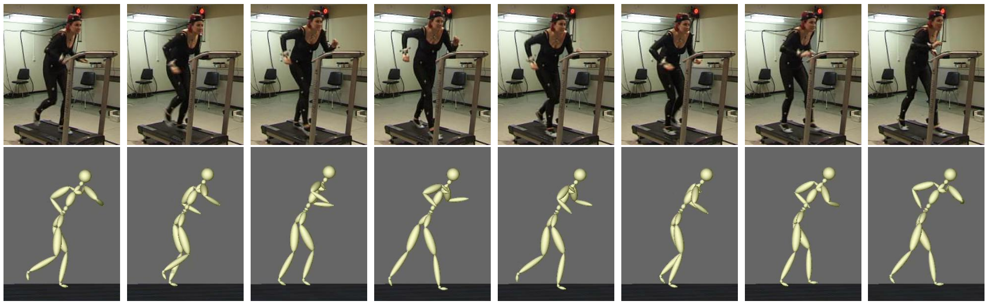


Figure 4.34: **Generation of stylized motion. Top row:** Original sneaking walking at 4.5 km/h. **Bottom row:** Walking with a sneaking style at 7km/h generated using only one example of such style at 4.5 km/h. Note that the database used is composed only of normal walking styles.

4.6 Conclusion and Future Work

We have presented an approach to incorporating strong motion models that yields full 3–D reconstruction using a single-hypothesis hill-climbing approach. This results in much lower computational complexity than the current multi-hypothesis techniques. We have demonstrate the effectiveness of our approach for monocular and multi-view tracking of cyclic motions as walking and running and acyclic motions as golf swinging. Moreover, we have shown that the tracking is accurate enough to perform subject identity recognition.

In the actual context we have traded the complexity of tracking for the complexity of knowing which model to apply. This might mean keeping several models active at any one time and selecting the one that fits best. This brings us back to multiple hypotheses tracking, but the multiple hypotheses are over models and not states. This might be much more effective than what many particle filters do because it ensures that the multiple hypotheses are sufficiently different to be worth exploring.

We have presented a real-time motion generation technique that allows us to generate the motion of a particular individual performing parameterized displacement activities. More specifically, we have investigated the case of walking, running, and jumping. The first two are cyclical and parametrized by speed. The third one is non-cyclical and parametrized in terms of jump length. Given one single example, we can modify the speed, length, or body size while preserving the individual's specific style. The required example can be obtained using either a sophisticated optical motion capture system or a much simpler set of synchronized cameras. While we have only validated our approach in the case of three specific cyclic and non-cyclic motions, we believe it to be fully generic and applicable to a whole range of activities.

Currently the crucial limitation of the motion generation method comes from the fact that we have not investigated the curvilinear motion patterns that would be required by a complete system to blend the straight line motion sequences we synthesize. We did not

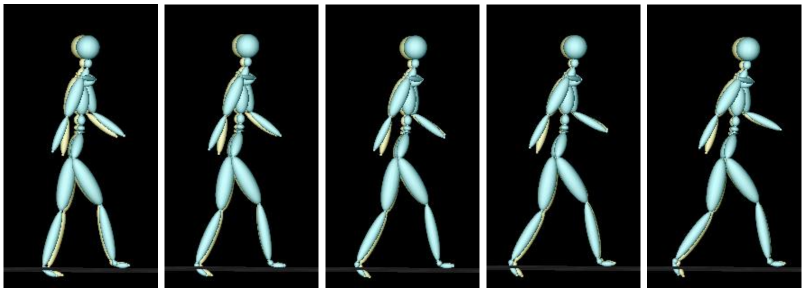


Figure 4.35: **Walking at speeds from 3 to 7 km/h.** Superposition of the video of Figure 4.22, in blue, on a whole set of optical motion capture examples, in yellow. The leg motion matches almost perfectly, which results in the lower body of the two figures being almost perfectly superposed. Small discrepancies in the arms are due to the intra-variability of the motion, the low resolution of the video, and the fact that the optical motions were performed using a treadmill. Note that the motion capture data has not been cleaned-up, which is why the left leg is penetrating the floor.

include such motions into our database because they cannot be captured on a treadmill and therefore require a more complex experimental set-up than the one we have. However, since they are also controlled by a well-identified parameter, namely the radius of curvature, we believe them to be amenable to our approach. Of course, increasing our repertoire of motions could result in non-linearities, which may require us to replace PCA by more sophisticated statistical tools such as Isomap [141] or the Gaussian Processes techniques described in chapters 5 and 6, that allow the same kind of treatment for non-linear models.

Another area that requires further investigation is the combination of our motion generation and tracking methods with inverse kinematics technique to clean up the artifacts, such as foot sliding, which can be observed in some of our results. The simplest approach would be to use Inverse Kinematics in a post-processing step. A more ambitious approach would be to detect foot support phases in real-time and enforce them with an IK solver [9, 49].

The major limitation of these linear motion models is the number of examples needed to create a complete database with good generalization properties. Moreover, as shown in Section 4.1.5 when learning multi-activity databases the convexity assumption is violated, and better probabilistic models are needed in order to allow the model to produce only physically possible motions. Chapters 5 and 6 describe non linear techniques that reduce considerably the number of examples required [151], and that provide more realistic probabilistic models of the probability of the latent coordinates than the simplistic underlying Gaussian model used here.

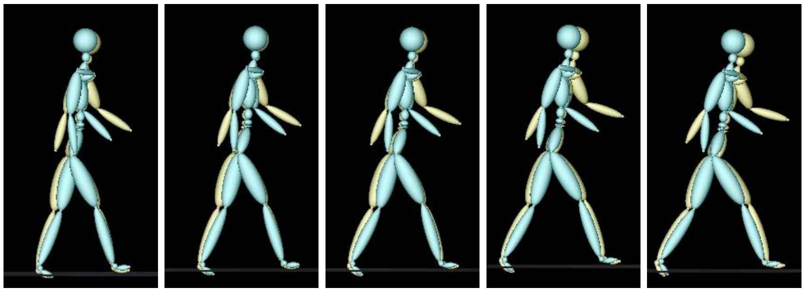


Figure 4.36: Side view of the original and synthesized motion from 3 to 7 km/h for the subject of the video sequence of Figure 4.23. The subject is compared to its closest neighbor in the database according to the Mahalanobis distance. The original motion capture data is represented in blue, and the synthesized one in yellow. They are quite different, which goes to show that our approach can generate a wide range of realistic styles.

5 Gaussian Process Latent Variable Models for 3D people Tracking

The 3D estimation of human pose from video, specially in the monocular case, is often poorly constrained, owing to reflection ambiguities, self-occlusion, cluttered backgrounds, non-rigidity of tissue and clothing, and poor image resolution. As a consequence, prior information is essential to resolve ambiguities, minimize estimator variance, and to cope with partial occlusions. Unfortunately, because of the high-dimensional parameterization of human models, learning prior models is difficult with small or modest amounts of training data. Manual design of suitable models is also very difficult.

In Chapter 4, we showed that learning motion models, as opposed to pose models, lets us use simple linear models to perform the tracking. These models are very useful since they are easy to learn and provide a dynamical model. However, the amount of training data they need, although not immensely large, can be prohibitive for some applications. For example the golf database composed of 9 swings performed by the same subject is too small to generalize well and to produce accurate tracking results. The training data has to be composed of segmented and time warped sequences (motions), instead of just poses. Moreover, it is difficult to learn transitions between motions because they need to be parameterized in terms of the same motion phase variable (Eq. 4.3), and their linear combination may result in unfeasible motions, or in motions very different from the ones used for training.

In this chapter, we therefore come back to pose models and show that, when used in conjonction with the appropriate non-linear statistical techniques, they can also be very effective. We exploit the recently developed Scaled Gaussian Process Latent Variable Model [54, 78] (SGPLVM) to learn a low-dimensional embedding of high-dimensional human pose data. The model can be learned from much smaller amounts of training data than competing techniques, such as [41, 130], and it involves very few manual parameter tuning.

SGPLVM provides a continuous, kernel-based density function $p(\mathbf{x}, \mathbf{y})$ over positions $\mathbf{x}$ in a low-dimensional latent space and positions $\mathbf{y}$ in the full pose space. The density function is generally non-Gaussian and multimodal. Importantly, it provides a natural preference for poses close to the training data, smoothly falling off with distance. The model also provides a simple, nonlinear, probabilistic mapping from the latent space to the full pose space. As explained below, $\mathbf{y}$ conditioned on $\mathbf{x}$ is a Gaussian random variable. Its variance reflects the uncertainty of the mapping, and therefore increases with the dissimilarity between $\mathbf{x}$ and the training data. This explicit representation of the variance is extremely useful.

This chapter explores the use of SGPLVM priors for monocular 3D people tracking. To

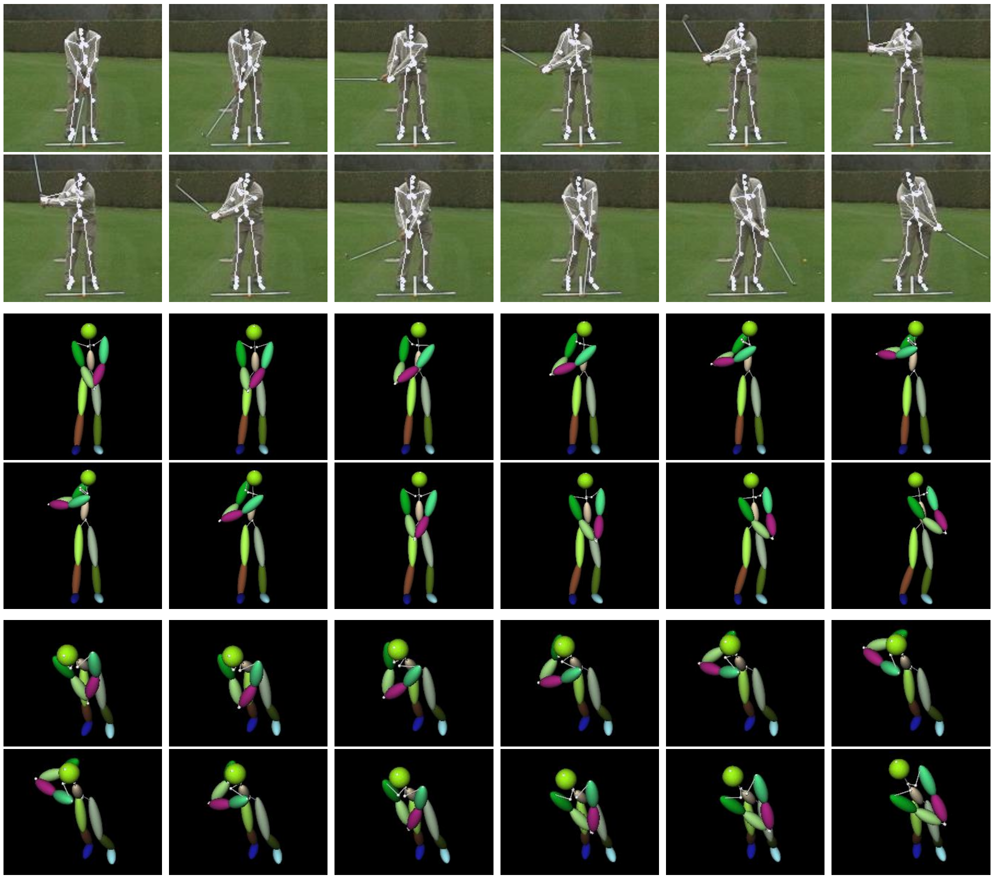


Figure 5.1: Tracking of a 62-frame *short* golf swing. **Top two rows**: The skeleton of the recovered 3D model is projected into a representative subset of images. **Middle two rows**: Volumetric primitives of the recovered 3D model projected into the same views. **Bottom two rows**: Volumetric primitives of the 3D model as seen from above.

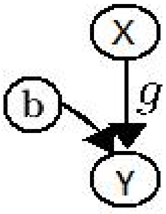


Figure 5.2: **Graphical model**: for the (S)GPLVM.

this end we consider two distinct domains, golfing and walking, and show that prior models can be learned from a single exemplar of each motion class. Both of these examples last just a second but are, nevertheless, shown to be sufficient for tracking. The generative model for the tracker comprises the SGPLVM, a simple likelihood function, and a second-order dynamical model. On-line tracking is accomplished with straightforward deterministic optimization.

5.1 Gaussian Process Models

Gaussian Processes are often introduced in the context of regression, to learn a mapping $\mathbf{y} = g(\mathbf{x}, \mathbf{B})$ from training pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$ [90]. In least-squares regression, the quality of the result often depends greatly on the specific form of g that one fits. Gaussian Processes arise from a Bayesian formulation in which one marginalizes over a family of functions for g. In this way, one mitigates common problems due to overfitting and underfitting. One can additionally learn the smoothness and noise parameters. Remarkably, for functions expressed as a linear combination of nonlinear basis functions $\{\chi_j\}$,

$$\mathbf{y} \;=\; \sum_j b_j\, \chi_j(\mathbf{x}) \,+\, \mathbf{n}_{\mathbf{y}} \tag{5.1}$$

with IID Gaussian noise $\mathbf{n}_{\mathbf{y}}$, and a Gaussian prior over $\{b_j\}$, the marginalization produces a Gaussian process model [90]. Fig. 5.2 depicts its graphical model.

5.1.1 Gaussian Process Latent Variable Models

In contrast to regression problems, here we are given training data, $\{\mathbf{y}_i\}$, but not the corresponding latent positions $\{\mathbf{x}_i\}$. As a consequence, we need to learn the unknown latent positions $\{\mathbf{x}_i\}$ along with the mapping from $\mathbf{x}$ to $\mathbf{y}$. Toward this end, the GPLVM [78] and SGPLVM [54] models formulate the likelihoods of the training data points $\{\mathbf{y}_i\}_{i=1}^N$, $y_i \in \mathcal{R}^D$, in terms of Gaussian processes for which the corresponding values $\{\mathbf{x}_i\}$ are initially unknown. These formulations can be viewed as a generalization of probabilistic PCA [145] where, instead of marginalizing over latent variables $\mathbf{x}$ to find the linear mapping from $\mathbf{x}$ to $\mathbf{y}$, we marginalize over mapping functions and optimize the latent positions $\{\mathbf{x}_i\}$.

A kernel function is introduced to allow for nonlinear mappings [78]. Scaling of individual data dimensions was introduced by Grochow et al. [54] to account for different variances of different dimensions of the data.

More formally[1], let $\mathbf{Y} \equiv [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T$ be a matrix, each row of which is one of the training data points. We assume that the mean $\mu \in \mathcal{R}^D$ has been subtracted from the data, so that the $\mathbf{y}_i$ are mean zero. Marginalizing over $\mathbf{B}$,

$$p(\mathbf{Y}|M) = \int_{\mathbf{B}} p(\mathbf{Y}, \mathbf{B}|M) d\mathbf{B} = \int_{\mathbf{B}} p(\mathbf{Y}|\mathbf{B}, M) p(\mathbf{B}|M) d\mathbf{B}, \tag{5.2}$$

results for a variety of functions in a Gaussian Process [78]. Under the Gaussian Process model, the conditional density for the data is multivariate Gaussian

$$p(\mathbf{Y} \mid M) = \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{ND} |\mathbf{K}_Y|^D}} \exp(-\frac{1}{2} \operatorname{tr}(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T)) . \tag{5.3}$$

Here, $M \equiv \{ \{\mathbf{x}_i\}, \{\beta_i\}, \{w_j\}_{j=1}^D \}$ comprises a vector of all the unknown SGPLVM model parameters. In particular, the matrix $\mathbf{K}_Y$ is called a kernel matrix, the elements of which are given by a kernel function, $(K_Y)_{ij} = k_Y(\mathbf{x}_i, \mathbf{x}_j)$, the specific form of which depends on the form of the basis functions in (5.1). Following [54, 78], we use a RBF kernel function, with parameters β_i given by

$$k_Y(\mathbf{x}_i, \mathbf{x}_j) = \beta_1 \exp(-\frac{\beta_2}{2} \|\mathbf{x}_i - \mathbf{x}_j\|^2) + \frac{\delta_{\mathbf{x}_i, \mathbf{x}_j}}{\beta_3} . \tag{5.4}$$

Here $\delta_{\mathbf{x}_i, \mathbf{x}_j}$ is the Kronecker delta function, β_3 is the variance of the additive noise in (5.1), while β_1 and β_2^{-1} represent the overall scale and the width of the RBF kernel. Finally, $\mathbf{W} \equiv \operatorname{diag}(w_1, ..., w_D)$ denotes a diagonal matrix containing one scale factor for each data dimension. In effect, the SGPLVM can be viewed as a GPLVM in which there are different kernel functions for each dimension, i.e., with $k_{Y,d}(\mathbf{x}_i, \mathbf{x}_j) \equiv k_Y(\mathbf{x}_i, \mathbf{x}_j)/w_d^2$ for dimension d, or to a Warped GP [134] with warping $\mathbf{YW}$.

Finally, we combine the data likelihood in (5.3) with prior distributions for the latent positions and the kernel hyperparameters and obtain a posterior density over the model M:

$$p(M \mid \mathbf{Y}) \;\propto\; p(\mathbf{Y} \mid M)\, p(M) \;=\; p(\mathbf{Y}|M)\, p(\mathbf{X})\, p(\bar{\beta}) p(\mathbf{W}) . \tag{5.5}$$

In what follows we adopt simple prior models [78], namely a IID mean-zero Gaussian prior over latent positions with unit covariance, an inverse prior over kernel hyperparameters, and a uniform prior over the weights:

$$p(\mathbf{X}) \;=\; \prod_i \mathcal{N}(\mathbf{x}_i | \, \mathbf{0}, \mathbf{I}) \;, \quad p(\beta_{1:3}) \;\propto\; \prod_{i=1}^{3} \frac{1}{\beta_i} \;, \tag{5.6}$$

---

[1] We refer the interested reader to [54, 78, 90] for more details.

Learning: During training, we learn the model parameters M by maximizing the posterior $p(M \mid \mathbf{Y})$. In doing so, we simultaneously learn the latent positions corresponding to the training data points, along with a continuous mapping from the latent space to the full pose space. By contrast with other techniques (e.g., [41, 130, 160]), where the embedding is specified first and a mapping is then learned separately.

The maximization of the posterior in (5.6), given the priors in (5.6), is equivalent to minimizing the negative log posterior. Up to an additive constant the negative log posterior, $-\ln p(M \mid \mathbf{Y})$, is given by

$$L_{learn} = \frac{D}{2} \ln |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right) + \frac{1}{2} \sum_i \|\mathbf{x_i}\|^2 + \sum_{i=1}^{3} \ln(\beta_i) - N \ln |\mathbf{W}| \,. \tag{5.7}$$

Learning requires the choice of the latent dimension and an initial guess for the latent positions and kernel hyperparameters, after which we use a scaled conjugate gradient method to minimize (5.7). More details and examples are given below.

Pose Prior: Once the model parameters M are learned, the joint density over a new latent position $\mathbf{x}$ and an associated pose $\mathbf{y}$ is given by [90, 151]

$$p(\mathbf{x}, \mathbf{y} \mid M, \mathbf{Y}) = p(\mathbf{y} \mid \mathbf{x}, M, \mathbf{Y}) p(\mathbf{x} \mid M, \mathbf{Y}) = \frac{p(\mathbf{y}, \mathbf{Y} \mid \mathbf{x}, M)}{p(\mathbf{Y} \mid \mathbf{X}, M)} p(\mathbf{x} \mid M, \mathbf{Y}). \tag{5.8}$$

Ignoring the terms that are constant for our maximization, the joint density becomes

$$p(\mathbf{x}, \mathbf{y} \mid M, \mathbf{Y}) \quad \propto \quad \frac{|\mathbf{W}|^{N+1}}{\sqrt{(2\pi)^{(N+1)D} |\hat{\mathbf{K}}_{\mathbf{Y}}|^D}} \exp(-\frac{1}{2} \mathrm{tr}(\hat{\mathbf{K}}_{\mathbf{Y}}^{-1} \hat{\mathbf{Y}} \mathbf{W}^2 \hat{\mathbf{Y}}^T)) \exp(-\frac{\mathbf{x}^T \mathbf{x}}{2}) \tag{5.9}$$

where $\hat{\mathbf{Y}} \equiv [\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{y}]^T$ comprises the training data points and the new pose $\mathbf{y}$, and $\hat{\mathbf{K}}_{\mathbf{Y}}$ is the corresponding new kernel matrix:

$$\hat{\mathbf{K}}_{\mathbf{Y}} = \begin{pmatrix} \mathbf{K}_{\mathbf{Y}} & \mathbf{k}_{\mathbf{Y}}(\mathbf{x}) \\ \mathbf{k}_{\mathbf{Y}}(\mathbf{x})^T & k_Y(\mathbf{x}, \mathbf{x}) \end{pmatrix}, \tag{5.10}$$

where $\mathbf{k}_{\mathbf{Y}}(\mathbf{x}) \equiv [k_Y(\mathbf{x}_1, \mathbf{x}), \cdots, k_Y(\mathbf{x}_N, \mathbf{x})]^T$.

Following [54, 90], one can derive a more useful expression for the likelihood of a new pair $(\mathbf{x}, \mathbf{y})$. That is, up to an additive constant, the negative log probability, $-\ln p(\mathbf{x}, \mathbf{y} \mid M, \mathbf{Y})$, is equal to

$$L(\mathbf{x}, \mathbf{y}) = \frac{\|\mathbf{W}(\mathbf{y} - \mu_{\mathbf{Y}}(\mathbf{x}))\|^2}{2\sigma^2(\mathbf{x})} + \frac{D}{2} \ln \sigma^2(\mathbf{x}) + \frac{1}{2} \|\mathbf{x}\|^2 \,, \tag{5.11}$$

with

$$\mu_{\mathbf{Y}}(\mathbf{x}) = \mu + \mathbf{Y}^T \mathbf{K}_{\mathbf{Y}}{}^{-1} \mathbf{k}_{\mathbf{Y}}(\mathbf{x}) \,, \tag{5.12}$$

$$\sigma_Y^2(\mathbf{x}) = k_Y(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{\mathbf{Y}}(\mathbf{x})^T \mathbf{K}_{\mathbf{Y}}{}^{-1} \mathbf{k}_{\mathbf{Y}}(\mathbf{x}) \,. \tag{5.13}$$

Here, $\mu_{\mathbf{Y}}(\mathbf{x})$ is the mean pose reconstructed from the latent position $\mathbf{x}$, i.e., the mean of $p(\mathbf{y} \,|\, \mathbf{x}, M, \mathbf{Y})$. Using (5.12), the mapping from the latent space to the pose space is continuous and relatively simple to compute. The variance, $\sigma_Y^2(\mathbf{x})$, gives the uncertainty of the reconstruction; it is expected to be small in the vicinity of the training data, and large far from them. Therefore, minimizing $L(\mathbf{x}, \mathbf{y})$ aims to minimize reconstruction errors (i.e., to keep $\mathbf{y}$ close to $\mu_Y(\mathbf{x})$), while keeping latent positions close to the training data (i.e., to keep $\sigma_Y^2(\mathbf{x})$ small). The third term in (5.11) is the result of a broad prior over latent positions that usually has relatively little influence on the optimized latent positions.

5.1.2 Human Body parametrization

In our specific application, each training point, $\mathbf{y}_i$, is a vector of joint angles that describes a body pose. We represent the human body as an articulated structure with 84 degrees of freedom for walking and 72 for golfing. These numbers are those in the training databases over which we had no control. While careful choice of joint representations can have a large influence on the success of parameter estimation and tracking, here we simply used the data as provided in the motion capture databases. The influence of the different parameterizations will be discuss in detail in chapter 7. We remove the joint angles from the training data that appear irrelevant (i.e., do not vary) in order to prevent the weighting matrix to becoming degenerate. While one might also wish to include global information, such as translational or orientational velocity, here, we do not since our walking data were obtained from a subject walking on a treadmill, for which we could not easily obtain global translation.

We chose the dimensionality of the latent space to be 2, since it is the smallest dimension to represent the motions we want to track. This was confirmed by Isomap; the error was smaller for that dimension. Moreover, for tracking we want to use a small dimension, since the less the parameters to estimate the easiest is in terms of minimization.

5.1.3 Sparsification and Overfitting

There are two goals when learning any type of prior model: We want the models to be computationally not expensive, and to be learned from small amounts of training data avoiding overfitting.

The main computational burden when learning a Gaussian Process is the inversion of the $N \times N$ kernel matrix, where N is the number of data samples. Its computational complexity is $\mathcal{O}(N^3)$. We also wish to limit the size of $\mathbf{K}_{\mathbf{Y}}$ in order to obtain a sufficiently smooth prior and avoid overfitting. When doing tracking we wish the prior models not to be overconfident with the training data, having similar variances in a wide neighborhood of the training data, and not just in the proximity of them, in order to allow a wide variety of motions, since it is often the case that the motions used for training and the motions to track are quite different.

The easiest way to decrease the computational complexity is to reduce the number of input data N. This can be done by subsampling the original training sequences. Fig. 5.3

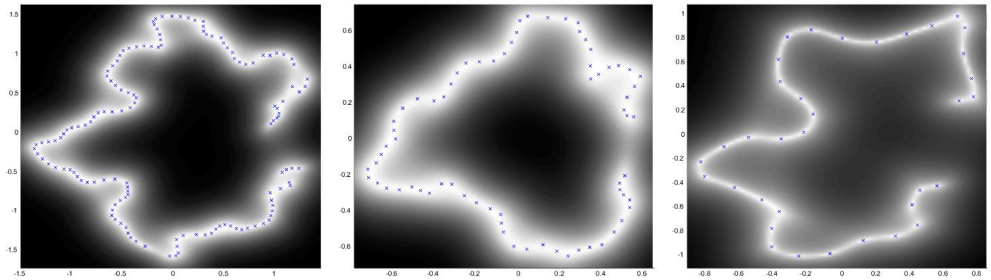


Figure 5.3: **Reducing the complexity by subsampling:** without sparsification. The grayscale plot represents $-\frac{D}{2}\ln\sigma^2(\mathbf{x}) - \frac{1}{2}\|\mathbf{x}\|^2$. The blue crosses are the optimized $\mathbf{x}_i$ positions associated with the training poses. (a) SGPLVM using the 139 samples, sampling at 120 Hzs (b) SGPLVM sampling at 60 Hzs. (b) SGPLVM sampling at 40 Hzs.

depicts different walking latent spaces with different sampling frequencies. The grayscale plot represents $-\frac{D}{2}\ln\sigma^2(\mathbf{x})-\frac{1}{2}\|\mathbf{x}\|^2$, thus depicting the regions of latent space that produce more likely poses. The walking model was learned from a single walk cycle performed on a treadmill, comprising 139 poses obtained with an optical motion capture system at 120 Hz.

When reducing the sampling frequency, one may filter the high frequencies, and introduce aliasing. Moreover, as depicted in Fig. 5.3, the model clearly overfits, and only poses very close to the data are likely (i.e., the variance is small only close to the training data). Instead, one can consider sparsification. Sparsification methods for Gaussian Process have been deeply investigated in the last years; see [115] for a complete review. One can also combine sparsification with subsampling.

The simplest possible sparse approximation is to only use a subset of the data (**SoD**[115]). Following [78, 79], while learning is based on the entire training set; the SGPLVM is constructed from a subset of the data referred to as the *active set*. In a greedy fashion, data points are added to the model one point at a time; at each step one chooses the point with the highest reconstruction variance (5.13). In this way, the active set tends to include training data points that are reasonably well spaced throughout the latent space.[2] The computational complexity is reduced to $\mathcal{O}(m^3)$, where m is the cardinality of the active set [115]. Fig. 5.4 depicts latent spaces learned using different sizes of the active set. Note that by using $m = 35$ or $m = 80$, the latent positions are very similar, while the complexity is very different. One of the consequences of using SoD with m small is that the overfitting decreases; when decreasing their number, the prior looks more and more smooth.

The sparse approximation using active sets suffers from a variety of problems: It does not get a realistic picture of the uncertainties, since only points in the training data are consider

---

[2] See [78] for details concerning the active set and heuristics used during learning and http://www.dcs.shef.ac.uk/ neil/gplvm/ for the GPLVM code.

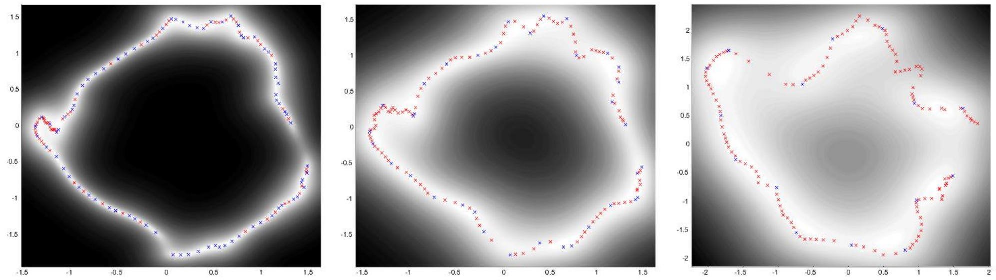


Figure 5.4: **Smoothing the prior by decreasing the number of points in the active set:** Learned SGPLVMs with (a) 80, (b) 35, (c) 15 points in the active set.

Initial Guess	Circle	PCA	Isomap	LEIG	LLE
L_{learn} (Fig. 5.5) (right)	-380.43	-558.51	-816.16	-574.75	-776.54

Table 5.1: **Reconstruction log likelihood** L_{learn}: for the 2D latent spaces shown in Fig. 5.5. Note that the reconstruction errors are clearly dependent of the initial conditions.

for the active set [115]. The solution can flip between two very different solutions, and it is very sensitive to initial conditions.

Fig. 5.5 (right) depicts the result of fitting an SGPLVM with the same cardinality of the active set starting from different initial conditions (left). Note that the learned spaces and their reconstruction errors (see Table 5.1), are very sensitive to the initial conditions.

Quinonero et al. [115] described a detailed set of different sparse approximations that defined the kernel matrix $\mathbf{K}_Y$ in terms of a set of m points that may or may not be part of the training set. This results in an algorithm fully convergent where the final mapping from latent space to data space takes into account all of the data (not just the points in the active set), resulting in a better approximation to the uncertainty of the model than the one obtained with SoD. In our particular application, because we want a smooth model, these methods provide less useful priors for tracking, and they do not produce good results with such small amounts of training data.

5.1.4 Non-smooth latent trajectories

One of the major problems with SGPLVM and GPLVM is that they often contain discontinuities[3] in the mapping. The main reason is that there is no prior in that points close in the input space (i.e., $\mathbf{y}_i \simeq \mathbf{y}_j$), should be close in the latent space, (i.e., $\mathbf{x}_i \simeq \mathbf{x}_j$). This is in contrast with geometric techniques such as LLE that specifically imposed that. The oppo-

---

[3] We define a discontinuity as the fact that two poses that are close in pose space are very far in latent space. When considering a temporal sequence, this fact produces non-smooth latent trajectories.

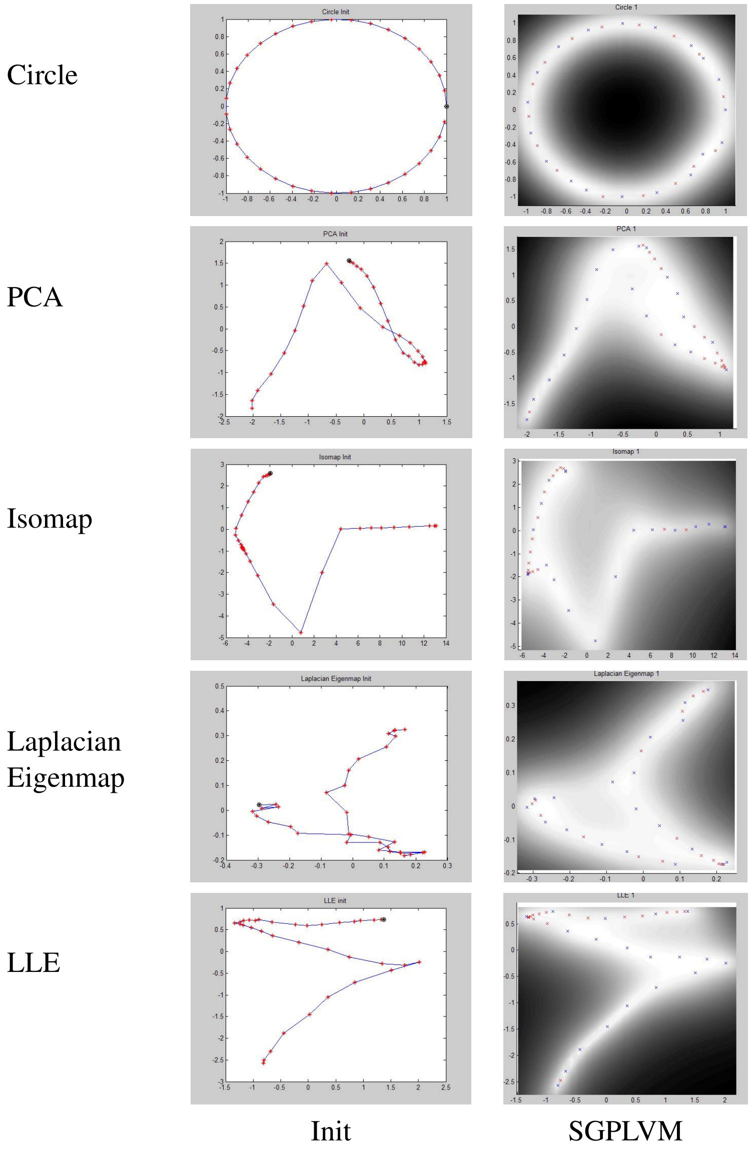


Figure 5.5: **SGPLVMs learned starting from different initial conditions** for the golfing sequence sampled at 30Hzs. (Left) Initial latent coordinates. From top to bottom: Circle, PCA, Isomap, Laplacian Eigenmaps, and LLE. (Right) Learned SGPLVM with active sets and $m = 19$.

site is true; points close in the latent space are close in the input space, since the Gaussian Process is modeling a mapping from $\mathbf{x}$ to $\mathbf{y}$.

This is shown in Fig. 5.6 where $L(\mathbf{y}_i, \mathbf{x})$ is depicted as a function of $\mathbf{x}$, for different training vectors $\mathbf{y}_i$. Each plot represents a specific training vector. The $\mathbf{x}$ that are close to $\mathbf{x}_j$, with $||\mathbf{y}_j - \mathbf{y}_i|| \gg$, are strongly unlikely. The minimun of $L(\mathbf{y}_i, \mathbf{x})$ is depicted as a black dot. Note that the $\mathbf{x}$ far away from the training data are also very likely, and may results in discontinuities when learning.

These discontinuities appear more frequently when the initialization contains loops, as depicted for the golf sequence sample at 60Hz on the left column of Fig. 5.5. In the intersection of the loop the two latent positions $\mathbf{x}$ are similar, while their associated input vectors $\mathbf{y}$ are different. This is fortly penalized by the SGPLVM. Both latent positions are pushed far apart, resulting in a discontinuity. Once this happens, there is no prior to attract them back together, and the discontinuities remain while learning. As a consequence, the SGPLVM cannot be learned from random initial conditions.

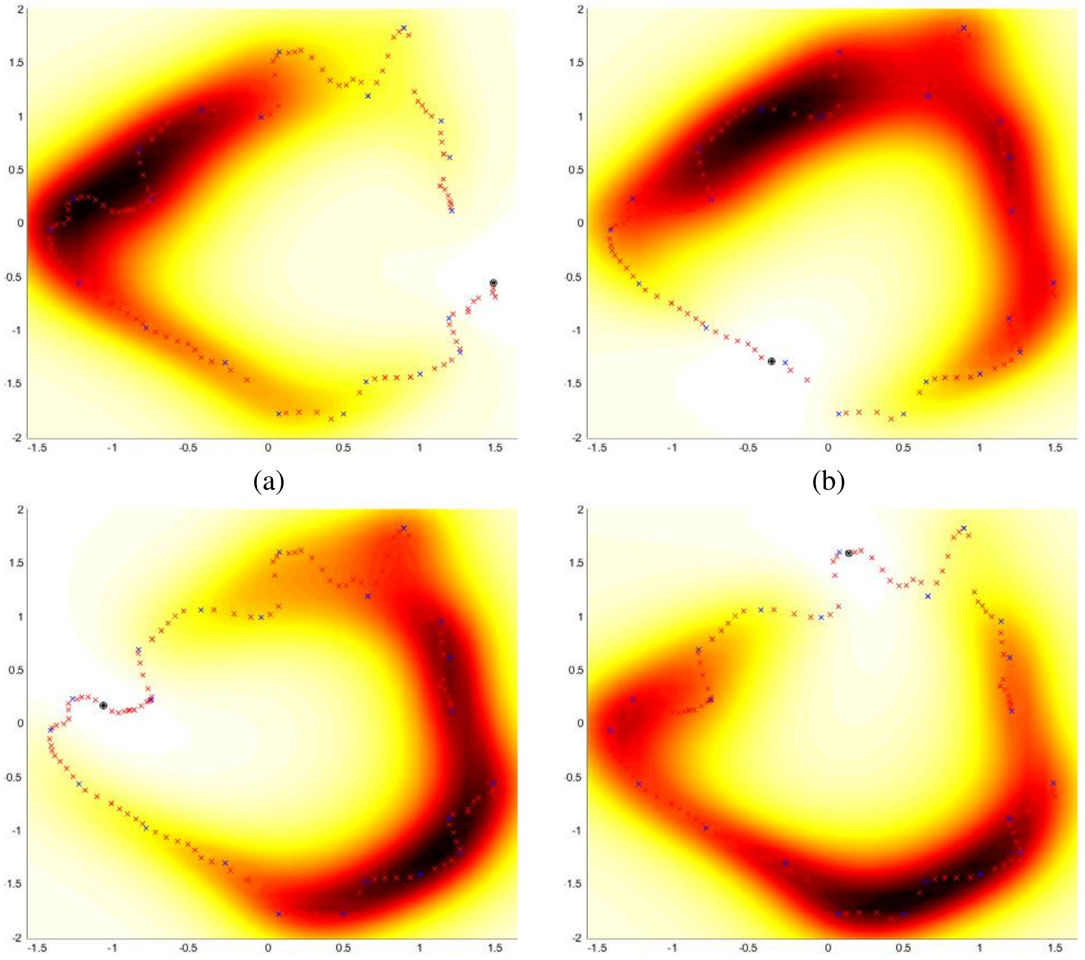
Sometimes the loops can be avoided by changing the initial conditions. Fig. 5.7 depicts results for the golfing sequence sampled at 60Hzs. The PCA initialization contains a loop at the end of the upper swing and beginning of the down swing, resulting in a big discontinuity. When using other initial positions, such as the ones given by LLE or Laplacian Eigenmaps, the learned latent spaces contain smooth latent trajectories.

But it is not always the case that changing the initial conditions will results in smooth latent trajectories. Better solutions exist. Lawrence [77] has proposed the use of back-constraints that 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. This requires an a priori knowledge to choose the type of back-constraints, and it does not necessary model a prior over the $\mathbf{X}$, that will justify them mathematically.

One can also run the optimization without searching for the maximum likelihood estimates of the latent positions, and only optimize the kernel hyperparameters and the scales w_i, given $\mathbf{X}$ by some other dimensionality reduction technique (e.g., PCA, LLE, Laplacian Eigenmaps). This is similar in spirit to [130], but replacing the RBF mapping and the GMM by the Gaussian Process. Fig. 5.8 shows a SGPLVM learned from fix PCA, LLE and Laplacian Eigenmaps latent positions, for the golfing sequence of Fig. 5.7. The reconstruction errors are worse than the ones obtained when learning SGPLVMs with the same initial conditions.

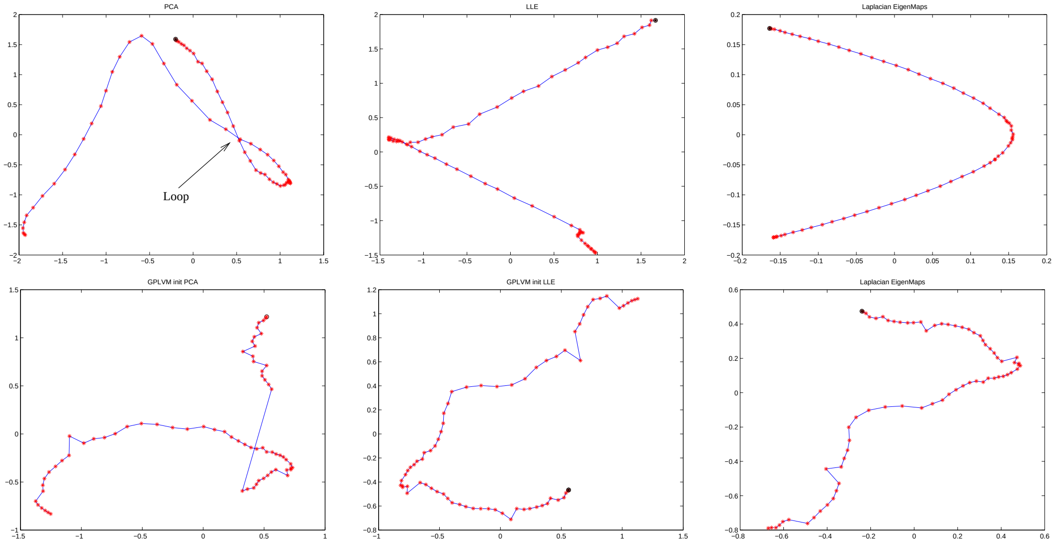
The use of all these methods are out of the scope of this work, and will be a subject of future research. We include them as possible solutions to some of the problems addressed in this chapter.

A better way to obtain smoothness is to model the dynamics, for example by using a Gaussian Process Dynamical Model (GPDM) [159]. GPDM consists in two Gaussian Process working in parallel modeling the pose reconstruction, and the dynamics. Implicitly, the GPDM forces two consecutive (in time) poses to be close in latent space. Chapter 6 will discuss how to use these models to learn pose dynamical models.



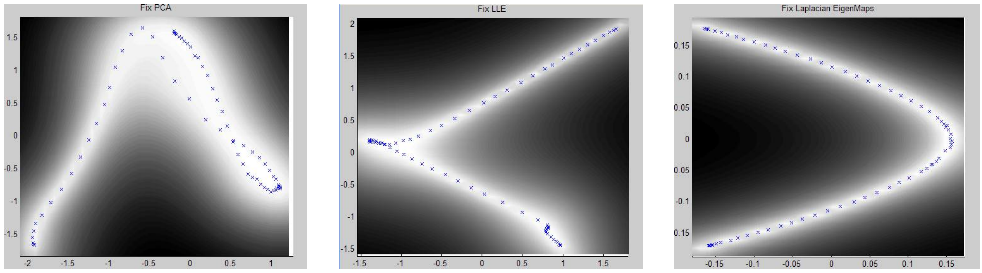
(a) (b) (c) (d)

Figure 5.6: **SGPLVM likelihood:** $L(\mathbf{y}_i, \mathbf{x})$ of Eq. 5.11 for a given $\mathbf{y}_i \in \mathbf{Y}$ as a function of x, for the walking sequence, composed of $N = 139$ poses sampled at 120Hz. (a) $L(\mathbf{y}_1, \mathbf{x})$, (b) $L(\mathbf{y}_{N/4}, \mathbf{x})$, (c) $L(\mathbf{y}_{N/2}, \mathbf{x})$, (d) $L(\mathbf{y}_{3N/4}, \mathbf{x})$. The minimun of each plot is depicted as a black dot. The $\mathbf{x}_i$ that are part of the active set are depicted as blue crosses, and the rest of the data as red crosses. White colors depict the most probably positions ($L(\mathbf{y}, \mathbf{x})$ small), and the dark ones the less probable ones (big $L(\mathbf{y}, \mathbf{x})$). Note that there is a very likely flat valley far from the training data, which may result in discontinuities.



PCA LLE Laplacian

Figure 5.7: **Jumps in the latent space:** can be produce when the initialization contains loops. **Top row:** Init conditions. **Botom row:** latent coordinates learned using a GPLVM without active sets, and the golfing sequence sampled at 60 Hzs.



PCA LLE Laplacian

Figure 5.8: **GP models:** where the $\mathbf{X}$ are given by PCA, LLE and Laplacian Eigenmaps using the same input data as for the models of Fig. 5.7.

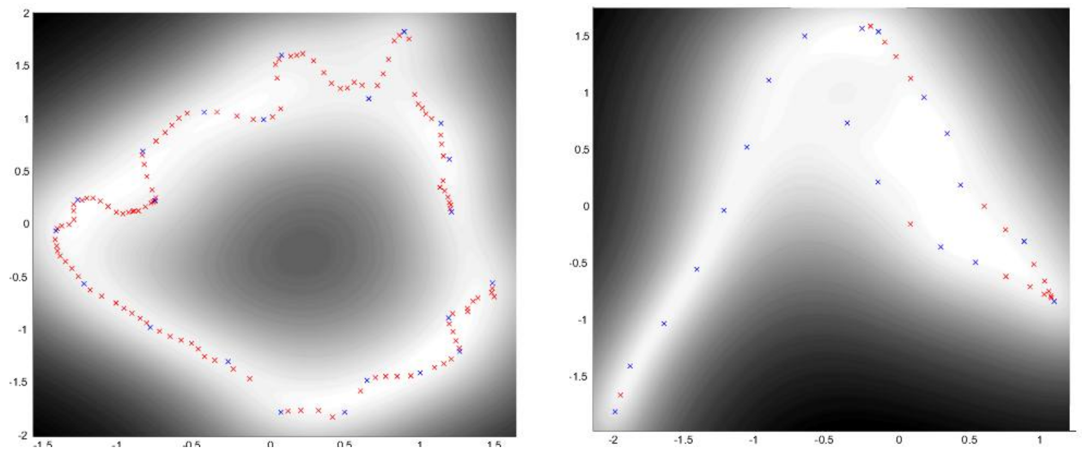


Figure 5.9: **2D SGPLVM latent spaces used for tracking**. The training data is compose of, **(left)**: 139 frames sampled at 120Hz of a walking cycle performed on a treadmill and retaining 22, **(right)**: 35 frames sampled at 30Hzs of a golf swing and retaining 19 active set points.

5.1.5 Reconstruction Error vs Tracking

While the reconstruction error is the minimization criteria when learning a Gaussian Process, it may not be the most desirable property for tracking. Instead one wants a smooth prior that do not overfit and generalize well to motions not in the training set. Moreover, since in hill climbing tracking, temporal consistency between consecutive frames is required not to get trap in local minima, we wish the latent trajectories to be smooth. This could be solved by using multi-hypothesis trackers, but may result in non-smooth results.

If we wish to use the SGPLVM as a prior for detecting individual frames, then the reconstruction error becomes more important since there is no need for smooth latent trajectories [142].

5.1.6 Learning Specific Motions for Tracking

When learning small amounts of training data, SGPLVM produces a smooth density function over training poses that can be used as a prior for tracking [151]. Fig. 5.9 depicts latent spaces for a walking and a golfing sequence that were used for tracking the sequences of Fig. 5.1, 5.12 and 5.13. PCA was used for initialization. For both models we used a 2D latent space, in part for simplicity and to allow for periodic motions.

An active set of 22 points was chosen for the walking to reduce the computational complexity and avoid overfitting. Even though the walking cycle is not exactly symmetric, as indicated by the gap between the beginning and end of the gait cycle on the right side of Fig. 5.9 (left), the SGPLVM effectively completes the curve with a low variance region that fills the gap.

The golfing model was learned from a single full swing composed of 35 poses sampled at 30 Hzs from the CMU database [29]. The active set contains 19 points and produces the smooth model shown in Fig. 5.9 (right). We choose this model, which uses subsampling and active sets, since the uncertainty is clearly under estimated (the model is very smooth). This is necessary since the motions to track are very different from the ones use for learning, especially the sequence of Fig. 5.1 that is a short swing.

5.2 Monocular Tracking

Our tracking formulation is based on a state-space model, with a SGPLVM prior over poses and a second order Markov model. The state at time t is defined as $\phi_t = [\mathbf{g}_t, \mathbf{y}_t, \mathbf{x}_t]$, where $\mathbf{g}_t$ denotes the global position $\mathbf{z}_t$ and orientation $\mathbf{o}_t$ of the body, $\mathbf{y}_t$ denotes the articulated joint angles, and $\mathbf{x}_t$ is a latent position. The goal is to estimate a state sequence, $\phi_{1:T} \equiv (\phi_1, ..., \phi_T)$, given an image sequence, $\mathbf{I}_{1:T} \equiv (\mathbf{I}_1, ..., \mathbf{I}_T)$, and a learned SGPLVM, M.

At each time t we form the posterior distribution over a subsequence of $\tau+1$ states

$$p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t+\tau},\, M) \;=\; c\; p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau})\, p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t-1},\, M)\;. \tag{5.14}$$

The sliding window allows us to exploit the influence of past data on estimates of the state at time t, as well as data from a small sequence of "future" observations. This typically produces smoother tracking results, but at the cost of a small delay of τ frames.

Rather than approximating the full posterior distribution we adopt a simple approximate MAP estimator. In particular, given previous MAP estimates, we use a simple form of hill-climbing to find new MAP estimates within the temporal window. In effect, this assumes the following approximation:

$$p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t+\tau},\, M) \;\approx\; c\, p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau})\, p(\phi_{t:t+\tau} \,|\, \phi^{MAP}_{1:t-1},\, M) \tag{5.15}$$

where $\phi^{MAP}_{1:t-1}$ denotes the MAP estimate history. In other words, it is assumed that all relevant information about past observations is contained solely in the previous MAP estimates. To find the MAP estimates at each time step we minimize the negative log posterior over states from time t to time $t+\tau$. At this minima we obtain the approximate MAP estimate at time t (given images $\mathbf{I}_{1:t+\tau}$).

5.2.1 Image Likelihood:

The current version of our 3D tracker uses a remarkably simple observation model. That is, the image observations were just the approximate 2D image locations of a small number (J) of 3D body points (see Fig. 5.10) that were obtained with a 2D tracker, e.g. [67].

The likelihood function is derived with the help of two further simple assumptions. First, we assume that the image measurements at each time step, conditioned on the respective states, are independent; i.e.,

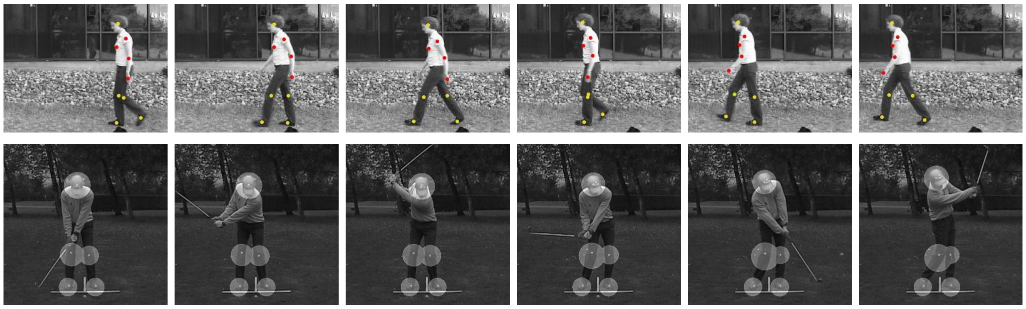


Figure 5.10: **2D Tracking using the WSL tracker. Top row:** Tracking the chest, knees, head, ankles and visible arm. The tracked upper body joints are shown in red, with the head and tracked lower joints points shown in yellow. **Bottom row**: Regions used for tracking the ankles, knees, and head are shown.

$$p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau}) \;=\; \prod_{i=t}^{t+\tau} p(\mathbf{I}_i \,|\, \phi_i) \; . \tag{5.16}$$

This assumption is made for computational convenience, since it is common for tracking measurement errors to be correlated through time.

Our second main assumption is that measurement noise in the 2D image positions are zero-mean Gaussian. In particular, let the perspective projection of the j^{th} body point, $\mathbf{p}^j$, in pose ϕ_t, be denoted $P(\mathbf{p}^j(\phi_t))$, and let the associated 2D image measurement from the tracker be $\hat{\mathbf{m}}_t^j$. Then, the negative log likelihood of the observations at time t is

$$-\ln p(\mathbf{I}_t \,|\, \phi_t) \;=\; \frac{1}{2\sigma_e^2} \sum_{j=1}^{J} \left\| \hat{\mathbf{m}}_t^j - P(\mathbf{p}^j(\phi_t)) \right\|^2 \; . \tag{5.17}$$

Here we set $\sigma_e = 3$ pixels, based on empirical results. No robust estimator is used since our input data does not contain outliers, but noise.

For example, Fig. 5.10 shows the 2D tracking locations for two test sequences. With the walking sequence we tracked 9 points on the body. For the golfing sequences we used 6 points. The fact that we use such a small number of tracked points is notable. By comparison, most successful 3D people trackers exploit several sources of image information, including edges, flow, silhouettes, skin detection, etc. The small number of constraints is also remarkable when compared to the dimension of the training poses. While it is known that 2D joint locations are useful for 3D pose estimation (e.g., [140]), it would not be possible for the optimization to find the pose parameters without a suitable prior.

5.2.2 Prediction Distribution

We further assume that predictions are constrained both by the temporal continuity of the motion as well as our GPLVM prior over plausible poses. In doing so the log prediction density is given by the sum of two potential functions E_1 and E_2. In particular, the negative log prediction density is given by

$$-\ln p(\phi_{t:t+\tau} \,|\, \phi_{1:t-1}^{MAP},\, M) \;\approx\; E_1(\phi_{t:t+\tau} \,|\, M, \mathbf{Y}) \;+\; E_2(\phi_{t+\tau} \,|\, \phi_{t-1}^{MAP}, \phi_{t-2}^{MAP})\;, \quad (5.18)$$

Here, E_1 is obtained from the SGPLVM pose prior, independently at each time instant:

$$E_1(\phi_{t:t+\tau}) \;=\; \sum_{j=t}^{t+\tau} L(\mathbf{x}_j, \mathbf{y}_j) \quad (5.19)$$

where $L(\mathbf{x}, \mathbf{y})$ is the SGPLVM negative log likelihood above in (5.11).

The second potential function E_2 encourages smoothness. In particular, we let the smoothness energy be the negative log transition density for a simple second-order Gauss-Markov model for the state evolution. Up to an additive constant this is given by

$$E_2(\phi_{t:t+\tau} \,|\, \phi_{t-1}^{MAP}, \phi_{t-2}^{MAP}) \;=\; -\sum_{j=t}^{t+\tau} ln\, p(\phi_j | \phi_{j-1:j-2}) \quad (5.20)$$

with the initial condition at time t provided by previous MAP estimates at times $t-1$ and $t-2$, i.e., $\phi_{t-1} \equiv \phi_{t-1}^{MAP}$ and $\phi_{t-2} \equiv \phi_{t-2}^{MAP}$.

We further assume that the joint angle dynamics are independent of the evolution of global position $\mathbf{z}_t$ and orientation $\mathbf{o}_t$ of the body:

$$p(\phi_j | \phi_{j-1:j-2}) \;=\; p(\mathbf{y}_j | \mathbf{y}_{j-1:j-2})\, p(\mathbf{z}_j | \mathbf{z}_{j-1:j-2})\, p(\mathbf{o}_j | \mathbf{o}_{j-1:j-2}). \quad (5.21)$$

Assuming a Gaussian process noise, we let the potential function be the log transition density given, up to an additive constant, by

$$E_2(\phi_{t:t+\tau} \,|\, \phi_{t-1:t-2}^{MAP}) \;=\; \sum_{j=t}^{t+\tau} \frac{||\mathbf{y}_j - \hat{\mathbf{y}}_j||^2}{2\sigma_y^2} + \frac{||\mathbf{z}_j - \hat{\mathbf{z}}_j||^2}{2\sigma_z^2} + \frac{||\mathbf{o}_j - \hat{\mathbf{o}}_j||^2}{2\sigma_o^2} \quad (5.22)$$

where the deterministic predictions, $\hat{\mathbf{z}}_j$, $\hat{\mathbf{o}}_j$ and $\hat{\mathbf{y}}_j$, assume zero acceleration and therefore take the form:

$$\hat{\mathbf{y}}_j \;=\; 2\mathbf{y}_{j-1} - \mathbf{y}_{j-2}\;,\;\; \hat{\mathbf{z}}_j \;=\; 2\mathbf{z}_{j-1} - \mathbf{z}_{j-2}\;,\;\; \hat{\mathbf{o}}_j \;=\; 2\mathbf{o}_{j-1} - \mathbf{o}_{j-2}\;.$$

In all the examples shown in this chapter, the dynamical model played a minor role in tracking (the variances were set to a high value).

5.2.3 Initialization and Optimization

For each test sequence we manually initialized the 3D position and orientation of the root node of the body in the first frame, $(\mathbf{z}_0, \mathbf{o}_0)$, so that it projects approximately to the right place. Similarly we manually gave the 2D locations of a few joints to be tracked by the

2D tracker. This entire process only required a few mouse clicks and could easily be automated using posture detection techniques [1, 41]. The initial states for the joint angles and latent positions, $\mathbf{y}_{0:\tau}, \mathbf{x}_{0:\tau}$, were chosen to be those in the training database that best projected onto the first $\tau + 1$ frames. The global positions and orientations, $(\mathbf{z}_{1:\tau}, \mathbf{o}_{1:\tau})$, were initialized to $(\mathbf{z}_0, \mathbf{o}_0)$, the ones set manually for the first frame.

Finally, MAP estimates were obtained using deterministic optimization. In particular, we minimize the negative log posterior obtained by substituting (5.15) into (5.14). After initializing the tracker in the first frame, the optimization is performed on-line, one frame at a time. The initial state for the optimization at each frame t, is given by the mean of the transition density, $\hat{\mathbf{y}}_t$, $\hat{\mathbf{z}}_t$ and $\hat{\mathbf{o}}_t$. Given that initial pose $\hat{\mathbf{y}}_t$, we first obtain an initial latent position $\hat{\mathbf{x}}_t = \arg\min_{\mathbf{x}} L(\mathbf{x}, \hat{\mathbf{y}}_t)$. Using this initial guess, we then use sequential quadratic optimization techniques to minimize the log posterior in (5.14), thereby finding the desired MAP estimate.

5.3 Results

The results shown in this chapter were obtained from uncalibrated images. The motions were performed by subjects of unknown sizes wearing ordinary clothes that are not particularly textured. To perform our computation, we used rough guesses for the subject sizes and for the intrinsic and extrinsic camera parameters to match the 2D projections.

5.3.1 Walking Motion

Figure 5.12 shows a well-known walking sequence. For 2D tracking we used the WSL tracker [67]. WSL is a robust, motion-based 2D tracker that maintains an on-line adaptive appearance model. The model adapts to slowly changing image appearance with a natural measure of the temporal stability of the underlying image appearance. It also permits the tracker to handle partial occlusions. As depicted in Fig. 5.10 (top row), 9 joints were tracked using WSL, namely, the ankles, knees, chest, head, left shoulder, elbow and hand.

Figure 5.12 shows the estimated 3D model projected onto several frames of the sequence, as well as some rendered 3D volumetric models. Note how well the skeleton reprojects onto the limbs even though the motion was learned from a single cycle of a different person on a treadmill. It is also interesting to see how well the arm is tracked (cf. [126]).

5.3.2 Golf Swing

As discussed in Section 5.1.6, the golf swing used to train the SGPLVM was a *full swing* from the CMU motion database [29]. It was performed by neither of the two golfers used for tracking (see Figs. 5.1 and 5.13). Here, we tracked five points using the WSL tracker, namely the knees, ankles and head. The initialization of these points could be also automated using posture detection techniques since the pose at the beginning of the swing is quite stereotyped.

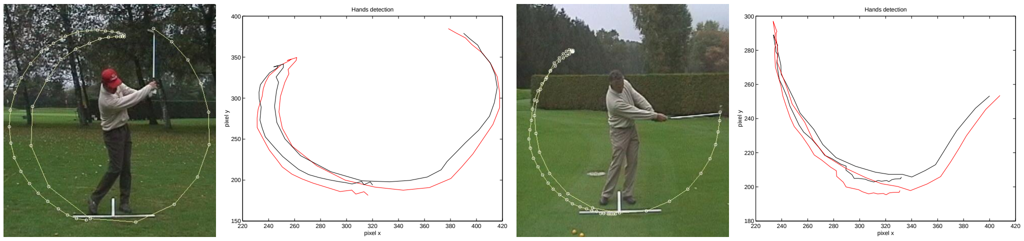


Figure 5.11: **Detected club and hand trajectories**: for the *full* swing of Fig. 5.13 and the *short* swing of Fig. 5.1. Left and right hand positions (pixel units) are represented in black and red respectively. Note that the full swing has a much longer trajectory than the other.

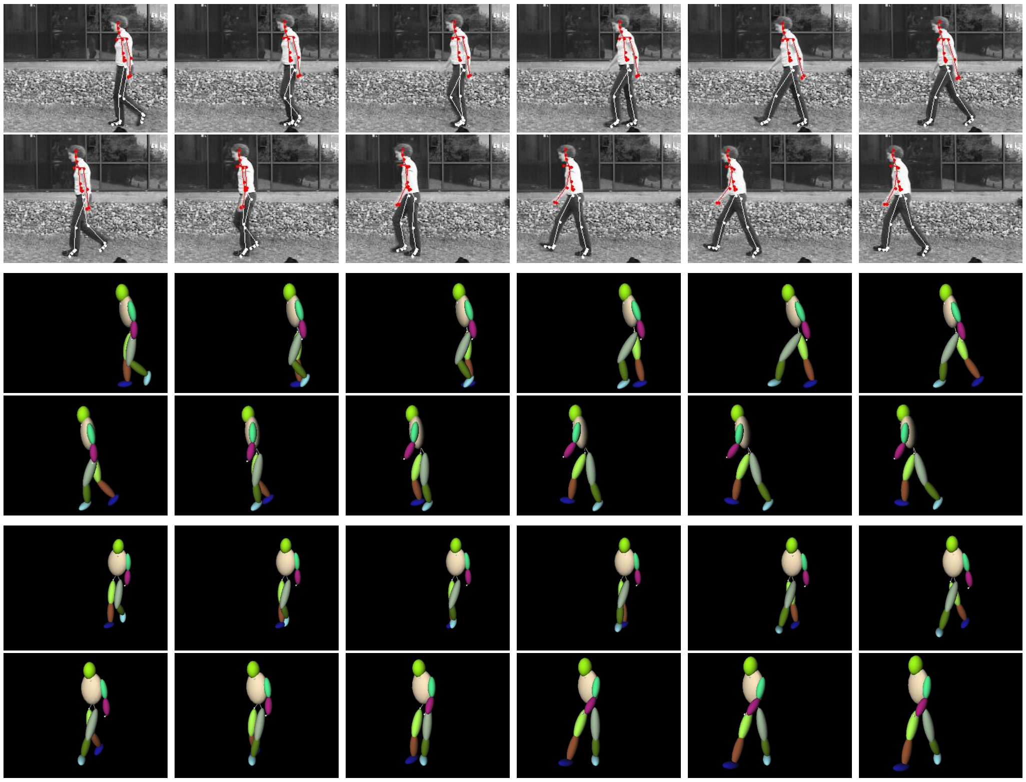


Figure 5.12: **Tracking a walking motion** of 32 frames. **First two rows**: The skeleton of the recovered 3D model is projected onto the images. **Middle two rows**: Volumetric primitives of the recovered 3D model projected into a similar view. **Bottom two rows**: Volumetric primitives of the 3D model as seen from the front view.

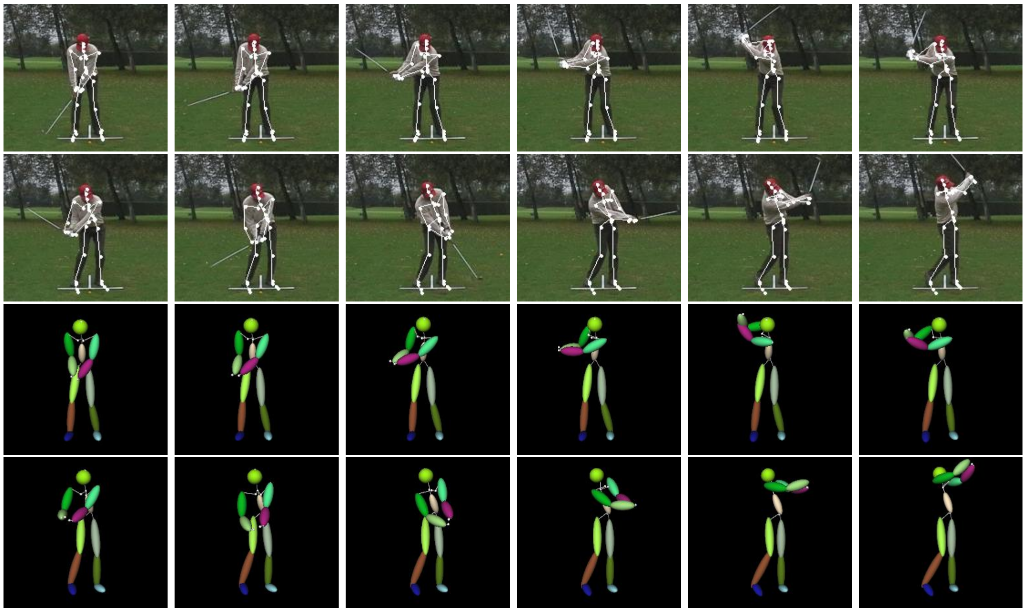


Figure 5.13: **Tracking of a full golf swing** in a 50 frames sequence. **First two rows**: The skeleton of the recovered 3D model is projected into a representative subset of images. **Last two rows**: Volumetric primitives of the 3D model as seen from viewpoint similar to the camera used.

Because the hands tend to rotate during the motion, to track the wrists we have found it effective to use a club tracking algorithm [82] that takes advantage of the information provided by the whole shaft. Its output is depicted in Fig. 5.11. This tracker does not require any manual initialization. It is also robust to mis-detections and false alarms and has been validated on many sequences. From the recovered club motion, we can infer the 2D hand trajectories as shown in Fig. 5.11.

The first two rows of Fig. 5.13 depict the projections of the recovered 3D skeleton in a representative subset of images of a *full* swing. The bottom two rows show projections of the 3D model using a viewpoint similar to the one of the original camera.

Fig. 5.1 depicts a *short* swing that is performed by a different person. Note that this motion is quite different both from the full swing motion of Fig. 5.13 and from the swing used to train the SGPLVM. The club does not go as high and, as shown in Fig. 5.11, the hands travel a much shorter distance. The tracking nevertheless remains very accurate. This helps illustrate the usefulness of the SGPLVM generalization.

In Fig. 5.14, we compare our current results with those obtained with a prior motion model learned with PCA from all ten swings in the CMU motion database. In that tracker (chapter 4) we used the same 2D tracked points, along with a brightness constancy constraint and 2D silhouette information. It also required pose detection of keyframes for initialization. By comparison, the method here is entirely on-line. Notice that with the SG-

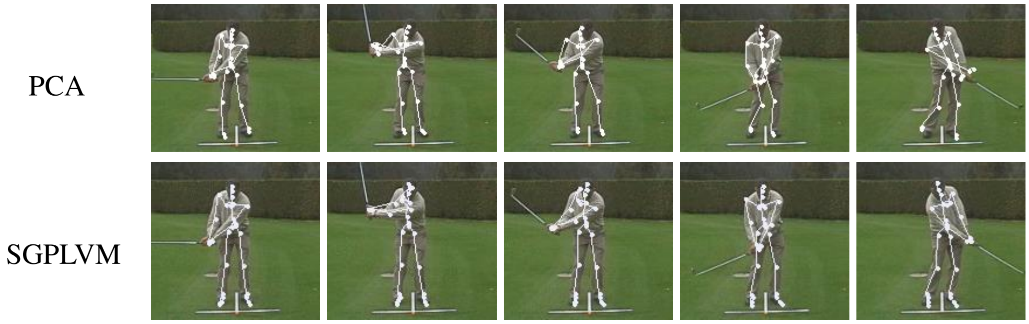


Figure 5.14: **PCA vs SGPLVM.** Comparison between the PCA-based tracker (chapter 4) in the **first row** and the SGPLVM-based tracker in the **second row**. Note the gain in accuracy due to the generalization ability of the SGPLVMs.

PLVM the skeleton's projection matches the limbs better than with the linear PCA-based model.

5.3.3 Failure Modes

The results shown above were obtained using models learned from very small amounts of training data. This is one of the strengths of the GPLVMs but comes at price: If the motion we are tracking differs substantially from the one used for training, the recovered latent positions can easily end up far from those corresponding to the training data. Fig. 5.15 illustrates this behavior in the case of the walking sequence of Fig. 5.12. These recovered latent positions correspond to poses that are close to the mean pose, since the kernels fall off exponentially.

When the observed pose is very different from the training ones, the first term of Eq. 5.11 dominates the optimization, and the latent positions are not anymore constraint to be close to the training data. This is depicted in Fig. 5.16. Note that due to the non-uniqueness of the Euler angle parameterization $\mathbf{y}$ may be very different from $\mathbf{y}_i \in \mathbf{Y}$.

A potential solution would be to learn the GPLVMs from larger amounts of data, for example in the form of several motions of a given type. However, increasing the amount of training data brings it own problems, such as the discontinuities depicted by Fig. 5.17.

A better approach is to explicitly model the dynamics [159] to overcome these problems and constrain latent positions that correspond to poses that are close (in time) to be close in latent space. This will be explored in Chapter 6.

5.4 Summary and Conclusions

We have presented a SGPLVM-based method to learn prior models of 3D human pose, and showed that it can be used effectively for monocular 3D tracking. In the case of both

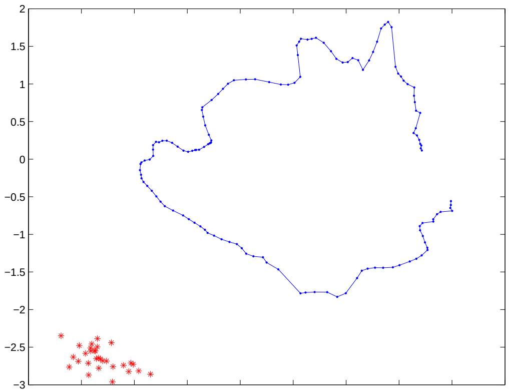


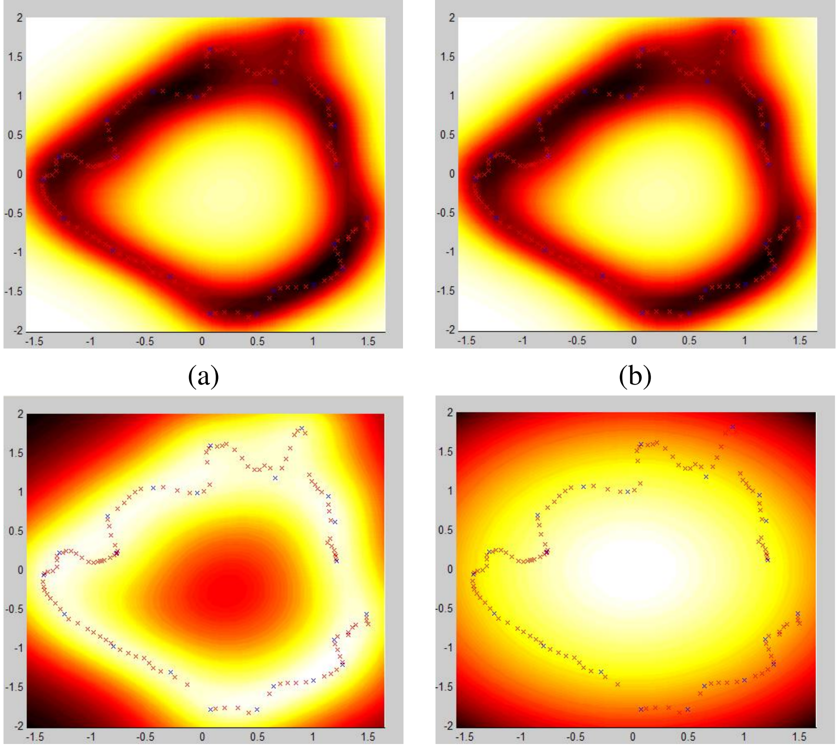
Figure 5.15: **Estimated latent positions:** when tracking the sequence of Fig. 5.12. Note that the latent positions obtained are far away from the training ones.

walking and golfing, we have been able to recover the motion from video sequences given a single exemplar of each motion to train the model. Furthermore, these priors sufficiently constrain the problem so that this could be accomplished with straightforward deterministic optimization. This is in sharp contrast to competing techniques that either involve large amounts of training data or computationally expensive multi-hypothesis tracking algorithms.

In this work, tracking was accomplished with particularly simple second-order dynamics and an observation model based on a very small number of tracked features. The quality of our results with such simple dynamics and appearance models clearly demonstrates the power of the SGPLVM prior models. More sophisticated appearance and dynamics models should produce even better results.

To increase the robustness of the tracking, one can replace the single hypothesis deterministic optimization by a probabilistic framework that will model the posterior instead of approximating it by a Dirac delta at the MAP estimate. Reinitialization is another important issue that is out of the scope of this work, by that will help preventing divergence.

In the presence of very noisy or missing data (e.g. occlusions), the simplistic second order Markov model is not complex enough to produce nice results. Moreover, when learning models that contain stylistic diversity (e.g., from different people or from the same person performing an activity multiple times), the SGPLVM results in models whose latent trajectories are not smooth, not being suitable for hill climbing tracking. Chapter 6 introduces a new model that learns smooth pose dynamical models with stylistic diversity, while producing very realistic motions under very challenging conditions, such as complete occlusions during a whole cycle, or tracking motions very different from the training ones. Further work will focus on incorporating multiple motion classes and transitions between them.



(a) (b)

(c) (d)

Figure 5.16: **SGPLVM likelihood:** (a) of Eq. 5.11 for a $\mathbf{y}$ that is far from the training data, as a function of the latent position $\mathbf{x}$. It is compose of three terms, (b) $\frac{\|\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))\|^2}{2\sigma^2(\mathbf{x})}$,(c) $\frac{D}{2}\ln\sigma^2(\mathbf{x})$, and (d)$\frac{1}{2}\|\mathbf{x}\|^2$. Lighter colors represent smaller values of $L(\mathbf{x},\mathbf{y})$. The prior in (a) is clearly dominated by (b).

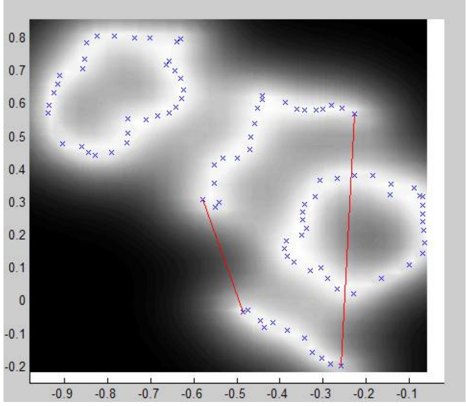


Figure 5.17: **Increasing the amount of training data.** SGPLVMs learned from a database containing 3 walking cycles, each one performed by a different subject. The latent trajectories are non smooth for one of the individuals, resulting in a model not suitable for tracking.

6 Gaussian Process Dynamical models

While powerful models of 3D human pose are emerging, sophisticated motion models remain rare. Most state-of-the-art approaches rely on simplistic linear-Gaussian Markov models that do not capture the complexities of human dynamics. Learning more accurate models is challenging because of the high-dimensional variability of human pose, the nonlinearity of human dynamics, and the relative difficulty of acquiring large amounts of training data.

Chapter 4 introduced motion models that take advantage of the convexity of the motion space to learn simple linear activity specific motion models. While useful, the linear combination of motions may result in unfeasible motions when learning different activities within the same model. Moreover, they require relatively big amounts of training data, which have to be segmented time warped motion sequences.

Chapter 5 explored a novel technique to learn pose models from as little as a single exemplar of the motion. The learning of training motions with stylistic diversity may result in models that contain non-smooth latent trajectories that are not suitable for tracking purposes. Moreover, the proposed tracking relies on simplistic second order Gaussian Markov models to encourage smoothness. These models do not cope with the complexities of human motion, resulting in tracking failures or unrealistic motions in the presence of extremely noisy and missing data, for example when dealing with occlusions.

This chapter shows that effective models for people tracking can be learned using the Gaussian Process Dynamical Model (GPDM) [159], even when modest amounts of training data are available. The GPDM is a latent variable model with a nonlinear probabilistic mapping from latent positions $\mathbf{x}$ to human poses $\mathbf{y}$, and a nonlinear dynamical mapping on the latent space. It provides a continuous density function over poses and motions that is generally non-Gaussian and multimodal. Given training sequences, one simultaneously learns the latent embedding, the latent dynamics, and the pose reconstruction mapping. Bayesian model averaging is used to lessen problems of over-fitting and under-fitting that are otherwise problematic with small training sets [78, 91].

We propose a form of GPDM, called Balanced GPDM, that learns smooth motion models given training motions with stylistic diversity. We show that these models are effective for tracking a range of human walking styles. The tracker is formulated as a MAP estimator on short pose sequences in a sliding temporal window, thereby producing motions that are smooth compared to conventional recursive estimators operating on one frame at a time. Estimates are obtained with straightforward deterministic optimization, and look remarkably good despite very noisy, missing or erroneous image data and significant occlusions.

6.1 Gaussian Process Dynamical Model

The GPDM is a latent variable dynamical model, comprising a low-dimensional latent space, a probabilistic mapping from the latent space to the pose space, and a dynamical model in the latent space [159]. Given training motions, one simultaneously learns the latent embedding, the latent dynamics, and the pose mapping. Even with small training sets it can learn effective models for 3D people tracking.

Consider a latent variable mapping with first order Markov dynamics

$$\begin{aligned} \mathbf{x}_t &= f(\mathbf{x}, \mathbf{A}) + \mathbf{n}_{x,t} && (6.1) \\ \mathbf{y}_t &= g(\mathbf{x}, \mathbf{B}) + \mathbf{n}_{y,t} && (6.2) \end{aligned}$$

where f and g are in general nonlinear parametric functions with parameters $\mathbf{A}$ and $\mathbf{B}$ respectively. Fig. 6.1 depicts its graphical model.

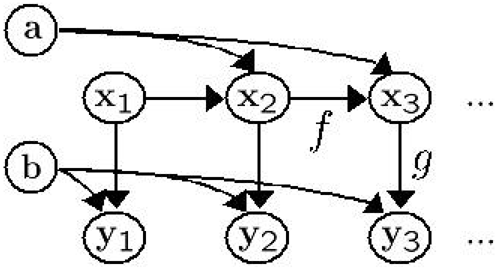


Figure 6.1: **Graphical model**: for the GPDM and Balanced GPDM.

Following [159], the GPDM is derived from a generative model for zero-mean poses $\mathbf{y}_t \in \mathcal{R}^D$ and latent positions $\mathbf{x}_t \in \mathcal{R}^d$, at time t, of the form

$$\begin{aligned} \mathbf{x}_t &= \sum_i \mathbf{a}_i \, \varphi_i(\mathbf{x}_{t-1}) + \mathbf{n}_{x,t} && (6.3) \\ \mathbf{y}_t &= \sum_j \mathbf{b}_j \, \chi_j(\mathbf{x}_t) + \mathbf{n}_{y,t} && (6.4) \end{aligned}$$

for weights $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ...]$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ...]$, basis functions φ_i and χ_j, and additive zero-mean white Gaussian noise $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$. For linear basis functions, (6.3) and (6.4) represent the common subspace AR model (e.g., [38]). With nonlinear basis functions, the model is significantly richer.

In conventional regression one fixes the number of basis functions and then fits the model parameters, $\mathbf{A}$ and $\mathbf{B}$. From a Bayesian perspective, these are nuisance parameters and should therefore be marginalized out through model averaging. With an isotropic Gaussian prior on each $\mathbf{b}_j$, as for the SGPLVM (chapter 5), one can marginalize over $\mathbf{B}$ in closed form [91, 103] to yield a multivariate Gaussian data likelihood of the form

$$p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}) = \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{ND} |\mathbf{K}_Y|^D}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right)\right) \qquad (6.5)$$

where $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$ is a matrix of training poses, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ contains the associated latent positions, and $\mathbf{K}_Y$ is a kernel matrix. The elements of kernel matrix are defined by a kernel function, $(\mathbf{K}_Y)_{i,j} = k_Y(\mathbf{x}_i, \mathbf{x}_j)$, which we take to be a common radial basis function (RBF) [91]:

$$k_Y(\mathbf{x}, \mathbf{x}') \; = \; \beta_1 \exp\left(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \frac{\delta_{\mathbf{x},\mathbf{x}'}}{\beta_3} \, . \tag{6.6}$$

As in SGPLVM [54] the scaling matrix $\mathbf{W} \equiv \mathrm{diag}(w_1, ..., w_D)$ is used to account for differing variances in the different data dimensions. Finally, $\bar{\beta} = \{\beta_1, \beta_2, ..., \mathbf{W}\}$ comprises the kernel hyperparameters that control the output variance, the RBF support width, and the variance of the additive noise $\mathbf{n}_{y,t}$.

The latent dynamics are similar; i.e., one forms the joint density over latent positions and the weights $\mathbf{A}$, and then marginalizes out $\mathbf{A}$ [159]

$$p(\mathbf{X}|\bar{\alpha}) = \int_{\mathbf{A}} p(\mathbf{X}, \mathbf{A}|\bar{\alpha}) d\mathbf{A} = \int_{\mathbf{A}} p(\mathbf{X}|\mathbf{A}, \bar{\alpha}) p(\mathbf{A}|\bar{\alpha}) d\mathbf{A}, \tag{6.7}$$

where $\bar{\alpha}$ is a vector of kernel hyperparameters. Assuming a first order Markov model

$$p(\mathbf{X}|\bar{\alpha}) = p(\mathbf{x}_1) \int_{\mathbf{A}} \prod_{t=2}^{N} p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{A}, \bar{\alpha}) p(\mathbf{A}|\bar{\alpha}) d\mathbf{A}. \tag{6.8}$$

With an isotropic Gaussian prior on the $\mathbf{a}_i$, the density over latent trajectories reduces to [158]

$$p(\mathbf{X} \,|\, \bar{\alpha}) \; = \; \frac{p(\mathbf{x}_1)}{\sqrt{(2\pi)^{(N-1)d} |\mathbf{K}_X|^d}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T\right)\right) \tag{6.9}$$

where $\mathbf{X}_{out} = [\mathbf{x}_2, ..., \mathbf{x}_N]^T$, $\mathbf{K}_X$ is the $(N-1) \times (N-1)$ kernel matrix constructed from $\mathbf{X}_{in} = [\mathbf{x}_1, ..., \mathbf{x}_{N-1}]$, and $\mathbf{x}_1$ is given an isotropic Gaussian prior. For dynamics the GPDM uses a "linear + RBF" kernel, with parameters α_i :

$$k_X(\mathbf{x}, \mathbf{x}') \; = \; \alpha_1 \exp\left(\frac{-\alpha_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \alpha_3 \mathbf{x}^T \mathbf{x}' + \frac{\delta_{\mathbf{x},\mathbf{x}'}}{\alpha_4}$$

The linear term is useful for motion subsequences that are approximately linear.

While the GPDM is defined above for a single input sequence, it is easily extended to multiple sequences $\{\mathbf{Y}_j\}$. One simply concatenates all the input sequences, ignoring temporal transitions from the end of one sequence to the beginning of the next. Each input sequence is then associated with a separate sequence of latent positions, $\{\mathbf{X}_j\}$, all within a shared latent space. Accordingly, in what follows, let $\mathbf{Y} = [\mathbf{Y}_1^T, ..., \mathbf{Y}_m^T]^T$ be the m training motions. Let $\mathbf{X}$ denote the associated latent positions, and for the definition of (6.9) let $\mathbf{X}_{out}$ comprise all but the first latent position for each sequence, and let $\mathbf{K}_X$ be the kernel matrix computed from all but the last latent position of each sequence.

6.1.1 Learning

Learning the GPDM entails estimating the latent positions and the kernel hyperparameters. Following [159] we adopt simple prior distributions over the hyperparameters[1],

$$p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}, \quad \text{and} \quad p(\bar{\beta}) \propto \prod_i \beta_i^{-1}. \tag{6.10}$$

The GPDM posterior becomes

$$p(\mathbf{X}, \bar{\alpha}, \bar{\beta} \,|\, \mathbf{Y}) \;\propto\; p(\mathbf{Y} \,|\, \mathbf{X}, \bar{\beta})\, p(\mathbf{X} \,|\, \bar{\alpha})\, p(\bar{\alpha})\, p(\bar{\beta})\;. \tag{6.11}$$

The latent positions and hyperparameters are found by minimizing the negative log posterior that, up to an additive constant, is equal to

$$\begin{aligned} \mathcal{L} \;=\; & \frac{d}{2} \ln |\mathbf{K}_X| \,+\, \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T\right) \\ & - N \ln |\mathbf{W}| \,+\, \frac{D}{2} \ln |\mathbf{K}_Y| \,+\, \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right) \\ & + \sum_i \ln \alpha_i \,+\, \sum_i \ln \beta_i \;, \end{aligned} \tag{6.12}$$

The first two terms come from the log dynamics (6.9), and the next three terms come from the log reconstruction density (6.5).

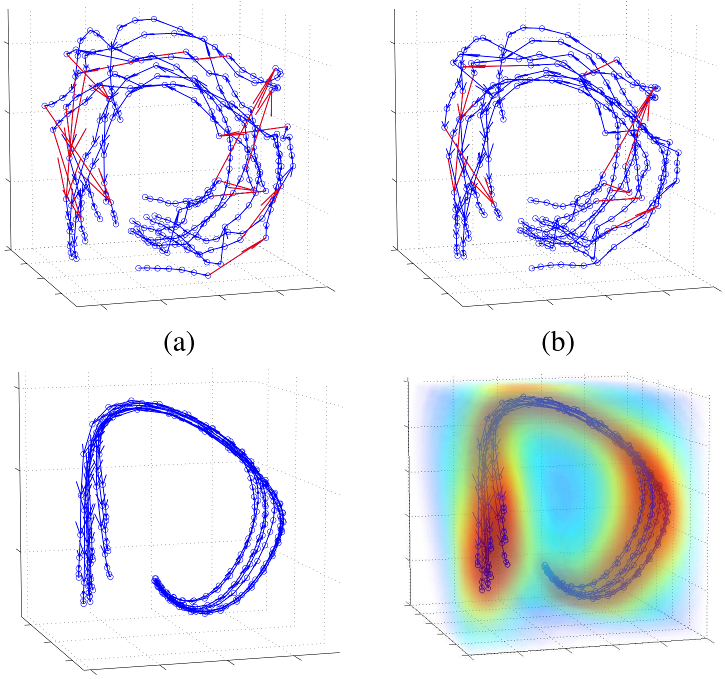
Over-Fitting: While the GPDM has advantages over the GPLVM, usually producing much smoother latent trajectories it can still produce large gaps between the latent positions of consecutive poses; e.g., Fig. 6.2 shows a GPLVM and a GPDM learned from the same golf swing data (large gaps are shown with red arrows). Such problems tend to occur when the training set includes a relatively large number of individual motions (e.g., from different people or from the same person performing an activity multiple times). The problem arises because of the large number of unknown latent coordinates and the fact that uncertainty in latent positions is not modeled. In practical terms, the GPDM learning estimates the latent positions by simultaneously minimizing squared reconstruction errors in pose space and squared temporal prediction errors in the latent space. In Fig. 6.2 the pose space is 80D and the latent space is 3D, so it is not surprising that the errors in pose reconstruction dominate the objective function, and thus the latent positions.

6.1.2 Balanced GPDM:

Ideally one should marginalize out the latent positions to learn hyperparameters, but this is expensive computationally. Instead, we propose a simple but effective GPDM modification

---

[1] Such priors prefer small output scale (i.e., $\alpha_1, \alpha_3, \beta_1$), large RBF support (i.e., small α_2, β_2), and large noise variances (i.e., small $\alpha_4^{-1}, \beta_3^{-1}$). The fact that the priors are improper is insignificant for optimization. See [158] for a discussion of different priors



(a) (b)

(c) (d)

Figure 6.2: **Golf Swing:** (a) GPLVM, (b) GPDM and (c) balanced GPDM learned from 9 different golf swings performed by the same subject. (d) Volumetric visualization of reconstruction variance; warmer colors (i.e., red) depict lower variance.

to balance the influence of the dynamics and the pose reconstruction in learning. That is, we discount the differences in the pose and latent space dimensions in the two regressions by raising the dynamics density function in (6.11) to the ratio of their dimensions, i.e., $\lambda = D/d$; for learning this rescales the first two terms in (6.12) to be

$$\lambda \left(\frac{d}{2} \ln |\mathbf{K}_X| + \frac{1}{2} \mathrm{tr} \left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T \right) \right) . \tag{6.13}$$

The resulting models are easily learned and very effective.

6.1.3 Model Results

Figs. 6.2–6.4 show models learned from motion capture data. In each case, before minimizing $\mathcal{L}$, the mean pose, μ, was subtracted from the input pose data, and PCA or Isomap were used to obtain an initial latent embedding of the desired dimension. We typically use a 3D latent space as this is the smallest dimension for which we can robustly learn complex motions with stylistic variability. The hyperparameters were initially set to one. The negative log posterior $\mathcal{L}$ was minimized using Scaled Conjugate Gradient.

Golf Swing: Fig. 6.2 shows models learned from 9 golf swings performed by one subject (from the CMU database). The body pose was parameterized with 80 joint angles, and the sequence lengths varied by 15 percent. The Balanced GPDM (Fig. 6.2(c)) produces smoother latent trajectories, and hence a more reliable dynamic model, than the original GPDM. Fig. 6.2(d) shows a volume visualization of the log variance of the reconstruction mapping, $2 \ln \sigma_{\mathbf{y}|\mathbf{x},\mathbf{X},\mathbf{Y},\bar{\beta}}$, as a function of latent position. Warmer colors correspond to lower variances, and thus to latent positions to which the model assigns higher probability; this shows the model's preference for poses close to the training data.

Walking: The model in Fig. 6.3 (c–f) was learned with one cycle from each of 6 subjects walking at the same speed on a treadmill. For each subject the first pose is replicated at the end of the sequence to encourage cyclical paths in the latent space. The body was parameterized with 30 joint angles. With the treadmill we do not have global position data, and hence we cannot learn the coupling between the joint angle times series and global translational velocity.

Fig. 6.3 (c,e) show the smooth, clustered latent trajectories learned from the training data. Fig. 6.3(e) shows a volume visualization of the log variance of the reconstruction mapping as a function of latent position. This shows the model's preference for poses close to the training data. Finally, Fig. 6.3(f) helps to illustrate the model dynamics by plotting 20 latent trajectories drawn at random from the dynamical model; again, they remain smooth and in the vicinity of the training data.

Speed Variation: Fig. 6.4 shows 2D GPDMs learned from four subjects, each of which walked four gait cycles at each of 9 speeds between 3 and 7km/h (equispaced). The learned latent trajectories are approximately circular, and organized by speed; the innermost and outermost trajectories correspond to the slowest and fastest speeds respectively. Interestingly, the subjects on the top row are healthy while the subjects on bottom row have a knee and hip pathology respectively. As the treadmill speed increases, the side of the body with the pathology performs the motion at slower speeds to avoid pain, and so the other side of the gait cycle must speed up to maintain the speed. This explains the anisotropy of the latent space.

6.1.4 Prior over New Motions

Finally, note that the GPDM also defines a smooth probability density over new motions $(\mathbf{Y}',\, \mathbf{X}')$: Just as we did with multiple sequences above, we write the joint density over the concatenation of the sequences. The conditional density of the new sequence is proportional to the joint density, but with training data and latent positions held fixed.

$$p(\mathbf{X}', \mathbf{Y}' \mid \mathbf{X}, \mathbf{Y}, \bar{\alpha}, \bar{\beta}) \;\propto\; p(\,[\mathbf{X}, \mathbf{X}'], [\mathbf{Y}, \mathbf{Y}'] \mid \bar{\alpha}, \bar{\beta}) \tag{6.14}$$

This density can also be factored to provide:

$$p(\mathbf{Y}' \mid \mathbf{X}',\, \mathbf{X},\, \mathbf{Y},\, \bar{\beta})\, p(\mathbf{X}' \mid \mathbf{X},\, \bar{\alpha})\ . \tag{6.15}$$

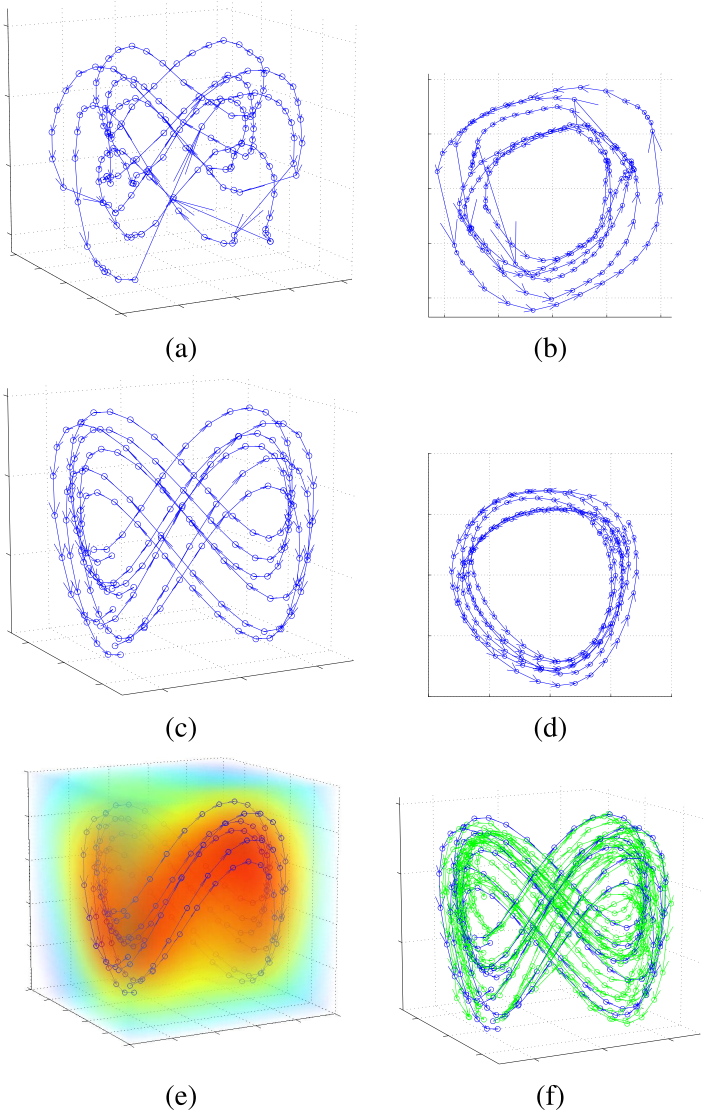

(a) (b) (c) (d) (e) (f)

Figure 6.3: **Walking GPLVM and GPDM:** Learned from 1 gait cycle from each of 6 subjects. Circles and arrows denote latent positions and temporal sequence. (a,b) Side/top views of the 3D latent space for the GPLVM. (c,d) Side/top views of the 3D latent space for the GPDM. (e) Volumetric visualization of the reconstruction variance. Warmer colors denote lower variance. (f) Green trajectories are fair samples from the dynamics model.

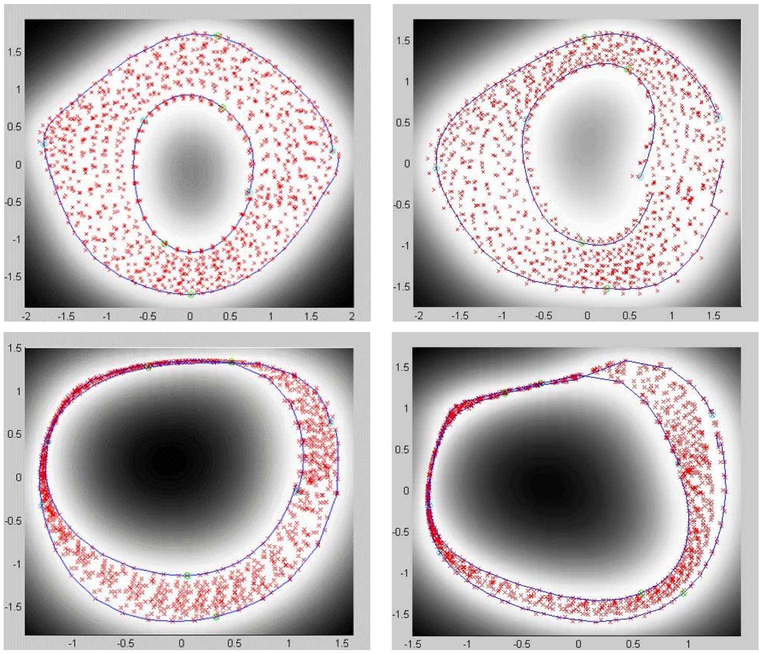


Figure 6.4: **Speed Variation:** 2D models learned for 4 different subjects. Each one walking at 9 speeds ranging from 3 to 7 km/h. Red points are latent positions of training poses. Intensity is proportional to $-2 \ln \sigma_{\mathbf{y}|\mathbf{x},\mathbf{X},\mathbf{Y},\bar{\beta}}$, so brighter regions have smaller pose reconstruction variance. **Top row:** healthy subjects. **Bottom row:** subjects with a knee and a hip pathology respectively, that walks asymmetrically.

For tracking we are typically given an initial state $\mathbf{x}'_0$, so instead of (6.15), we obtain

$$p(\mathbf{Y}' \mid \mathbf{X}', \, \mathbf{X}, \, \mathbf{Y}, \, \bar{\beta}) \, p(\mathbf{X}' \mid \mathbf{X}, \, \bar{\alpha}, \mathbf{x}'_0) \, . \tag{6.16}$$

Let $\mathbf{Y}' = [\mathbf{y}'_1, ..., \mathbf{y}'_{\mathbf{N}'}]^T$, and $\mathbf{X}' = [\mathbf{x}'_1, ..., \mathbf{x}'_{\mathbf{N}'}]^T$. Note that the first term of the density in Eq. (6.16) cannot be factored as a Markov chain. The poses are not independent when conditioned on the latent sequence, and we cannot write the density over a latent sequence as a product of low-order Markov transitions. Although this makes inference more difficult, we heuristically approximate it as

$$\prod_{i=1}^{\mathbf{N}'} p(\mathbf{y}'_i \mid \mathbf{x}'_i, \, \mathbf{X}, \, \mathbf{Y}, \, \bar{\beta}) \;\; p(\mathbf{X}' \mid \mathbf{X}, \, \bar{\alpha}, \mathbf{x}'_0) \, . \tag{6.17}$$

6.2 Tracking

Our tracking formulation is based on a state-space model, with a GPDM prior over pose and motion. The state at time t is defined as $\phi_t = [\mathbf{g}_t, \mathbf{y}_t, \mathbf{x}_t]$, where $\mathbf{g}_t$ denotes the global position and orientation of the body, $\mathbf{y}_t$ denotes the articulated joint angles, and $\mathbf{x}_t$ is a latent position. The goal is to estimate a state sequence, $\phi_{1:T} \equiv (\phi_1, ..., \phi_T)$, given an image sequence, $\mathbf{I}_{1:T} \equiv (\mathbf{I}_1, ..., \mathbf{I}_T)$, and a learned GPDM, $M = (\mathbf{X}, \mathbf{Y}, \bar{\alpha}, \bar{\beta})$. Toward that end there are two common approaches: *On-line methods* infer ϕ_t given the observation history $\mathbf{I}_{1:t-1}$. The inference is causal, and usually recursive, but suboptimal as it ignores future data. *Batch methods* infer states ϕ_t given all past, present and future data, $\mathbf{I}_{1:T}$. Inference is optimal, but requires all future images which is impossible in most tracking applications.

Here we propose a compromise that allows some use of future data along with predictions from previous times. In particular, at each time t we form the posterior distribution over a (non-causal) sequence of $\tau+1$ states

$$p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t+\tau}, M) \;=\; c\; p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau})\, p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t-1}, M)\;. \tag{6.18}$$

Inference at time t is improved, but at the cost of a small temporal delay.[2] With a Markov chain model one could use a forward-backward belief propagation algorithm [162] in which separate beliefs about each state from past and future data are propagated forward and backward in time. Here, instead we consider the posterior over the entire window, without requiring the Markov factorization.

We also exploit the power of the GPDM prior, and assume that we can use hill-climbing to find good state estimates. Thus, rather than approximating the entire posterior, we find MAP estimates. In effect, this assumes

$$p(\phi_{t:t+\tau} \,|\, \mathbf{I}_{1:t+\tau}, M) \;\approx\; c\, p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau})\, p(\phi_{t:t+\tau} \,|\, \phi^{MAP}_{1:t-1}, M) \tag{6.19}$$

where $\phi^{MAP}_{1:t-1}$ denotes the MAP estimate history. This has the disadvantage that complete beliefs are not propagated forward. But with the temporal window we still exploit data over several frames, yielding smooth tracking.

Accordingly, at each time step we minimize the negative log posterior over states from time t to time $t+\tau$. At this minima we obtain the approximate MAP estimate at time t (given images $\mathbf{I}_{1:t+\tau}$). The estimate is approximate in two senses. First, we do not represent and propagate uncertainty forward from time $t-1$ in (6.19). Second, because previous MAP estimates are influenced by future data, there is a bias in the information propagated forward.

Image Likelihood: The current version of our 3D tracker uses a remarkably simplistic observation model. That is, the image observations were just the approximate 2D image

---

[2] However an on-line estimate of $\phi_{t+\tau}$ would still be available at $t+\tau$.

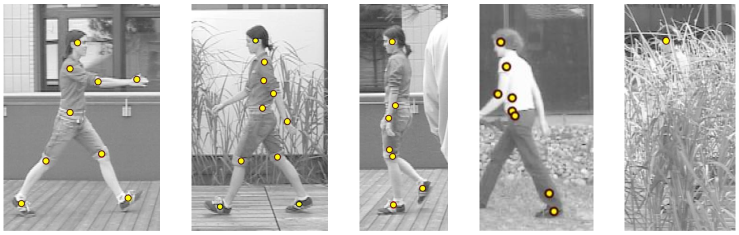

(a) (b) (c) (d) (e)

Figure 6.5: **WSL Tracks:** The 2D tracked regions for the different tracked sequences (in yellow) are noisy and sometimes missing.

locations of a small number (J) of 3D body points (see Fig. 6.5). They were obtained with the WSL image-based tracker [67].

While measurement errors in tracking are often correlated over time, as is common we assume that image measurements conditioned on states are independent; i.e.,

$$p(\mathbf{I}_{t:t+\tau} \,|\, \phi_{t:t+\tau}) \;=\; \prod_{i=t}^{t+\tau} p(\mathbf{I}_i \,|\, \phi_i) \;. \tag{6.20}$$

Further, we assume zero-mean Gaussian measurement noise in the 2D image positions provided by the tracker. Let the perspective projection of the j^{th} body point, $\mathbf{p}^j$, in pose ϕ_t, be denoted $P(\mathbf{p}^j(\phi_t))$, and let the associated 2D image measurement from the tracker be $\hat{\mathbf{m}}_t^j$. Then, the negative log likelihood of the observations at time t is

$$-\ln p(\mathbf{I}_t \,|\, \phi_t) \;=\; \frac{1}{2\sigma_e^2} \sum_{j=1}^{J} \left\| \hat{\mathbf{m}}_t^j - P(\mathbf{p}^j(\phi_t)) \right\|^2 \;. \tag{6.21}$$

Here we set $\sigma_e = 10$ pixels, based on empirical results.

Prediction Distribution We factor the prediction density $p(\phi_{t:t+\tau} \,|\, \phi_{1:t-1}^{MAP},\, M)$ into a prediction over global motion, and one over poses $\mathbf{y}$ and latent positions $\mathbf{x}$. The reason, as discussed above, is that our training sequences did not contain the global motion. So, we assume that

$$p(\phi_{t:t+\tau} \,|\, \phi_{1:t-1}^{MAP},\, M) \;=\; p(\mathbf{X'}_t,\, \mathbf{Y'}_t \,|\, \mathbf{x}_{t-1}^{MAP},\, M)\, p(\mathbf{g}_{t:t+\tau} | \mathbf{g}_{t-1:t-2}^{MAP}) \;, \tag{6.22}$$

where $\mathbf{X'}_t = \mathbf{x}_{t:t+\tau}$ and $\mathbf{Y'}_t = \mathbf{y}_{t:t+\tau}$.

For the global rotation $\mathbf{o}_t$ and translation $\mathbf{z}_t$, such that $g_t = (\mathbf{o}_t, \mathbf{z}_t)$, we assume a second-order Gauss-Markov model. The negative log transition density is, up to an additive constant,

$$-\ln p(\mathbf{g}_j \,|\, \mathbf{g}_{j-1:j-2}) \;=\; \frac{||\mathbf{z}_j - \hat{\mathbf{z}}_j||^2}{2\sigma_z^2} + \frac{||\mathbf{o}_j - \hat{\mathbf{o}}_j||^2}{2\sigma_o^2} \tag{6.23}$$

where the mean prediction is just

$$\hat{\mathbf{z}}_j \;=\; 2\mathbf{z}_{j-1} - \mathbf{z}_{j-2}\;,\;\; \hat{\mathbf{o}}_j \;=\; 2\mathbf{o}_{j-1} - \mathbf{o}_{j-2}\;.$$

with the initial condition at time t provided by previous MAP estimates at times $t-1$ and $t-2$, i.e., $\mathbf{g}_{t-1} \equiv \mathbf{g}_{t-1}^{MAP}$ and $\mathbf{g}_{t-2} \equiv \mathbf{g}_{t-2}^{MAP}$.

For the prior over $\mathbf{X}'_t$, $\mathbf{Y}'_t$, we approximate the GPDM in two ways. First we assume that the density over the pose sequence, $p(\mathbf{Y}'_t \mid \mathbf{X}'_t,\, M)$, can be factored into the densities over individual poses. This is convenient computationally since the GPDM density over a single pose, given a latent position, is Gaussian [54, 151]. Thus we obtain

$$\begin{aligned}
-\ln p(\mathbf{Y}'_t|\mathbf{X}'_t, M) &\approx -\sum_{j=t}^{t+\tau} \ln p(\mathbf{y}_j|\mathbf{x}_j, \bar{\beta}, \mathbf{X}, \mathbf{Y}) \\
&= \sum_{j=t}^{t+\tau} \frac{\|\mathbf{W}(\mathbf{y}_j - \mu_Y(\mathbf{x}_j))\|^2}{2\sigma^2(\mathbf{x}_j)} + \frac{D}{2}\ln\sigma^2(\mathbf{x}_j) + \frac{1}{2}\|\mathbf{x}_j\|^2
\end{aligned}$$

where the mean and variance are given by

$$\begin{aligned}
\mu_Y(\mathbf{x}) &= \mu + \mathbf{Y}^T\mathbf{K}_Y^{-1}\mathbf{k}_Y(\mathbf{x})\,, && (6.24)\\
\sigma^2(\mathbf{x}) &= k_Y(\mathbf{x},\mathbf{x}) - \mathbf{k}_Y(\mathbf{x})^T\mathbf{K}_Y^{-1}\mathbf{k}_Y(\mathbf{x})\,, && (6.25)
\end{aligned}$$

and $\mathbf{k}_Y(\mathbf{x})$ is the vector with elements $k_Y(\mathbf{x},\mathbf{x}_j)$ for all other latent positions x_j in the model.

Second, we anneal the dynamics $p(\mathbf{X}'_t|\mathbf{x}_{t-1}^{MAP}, M)$, because the learned GPDM dynamics often differ in important ways from the video motion. The most common problem occurs when the walking speed in the video differs from the training data. To accommodate this we effectively blur the dynamics; this is achieved by raising the dynamics density to a small exponent, or simply just using a smaller value of λ in (6.13), for which the kernel matrix must also be updated to include $\mathbf{X}'_t$. For tracking, we fix $\lambda = 0.5$.

Optimization: Tracking is performed by minimizing the approximate negative log posterior in (6.19). With the approximations above this becomes

$$\begin{aligned}
F_t \;=\; &-\sum_{j=t}^{t+\tau} \ln p(\mathbf{I}_j \mid \phi_j) \;-\; \sum_{j=t}^{t+\tau} \ln p(\mathbf{g}_j|\mathbf{g}_{j-1:j-2}) \\
&-\ln p(\mathbf{X}'_t|\bar{\alpha}, \mathbf{X}) \;-\; \sum_{j=t}^{t+\tau} \ln p(\mathbf{y}_j|\mathbf{x}_j, \bar{\beta}, \mathbf{X}, \mathbf{Y}) \qquad (6.26)
\end{aligned}$$

To minimize F_t in (6.26) with respect to $\phi_{t:t+\tau}$, we find that the following procedure helps to speed up convergence, and to reduce getting trapped in local minima. Each new state is first set to be the mean prediction, and then optimized in a temporal window. For the experiments we use $\tau = 2$.

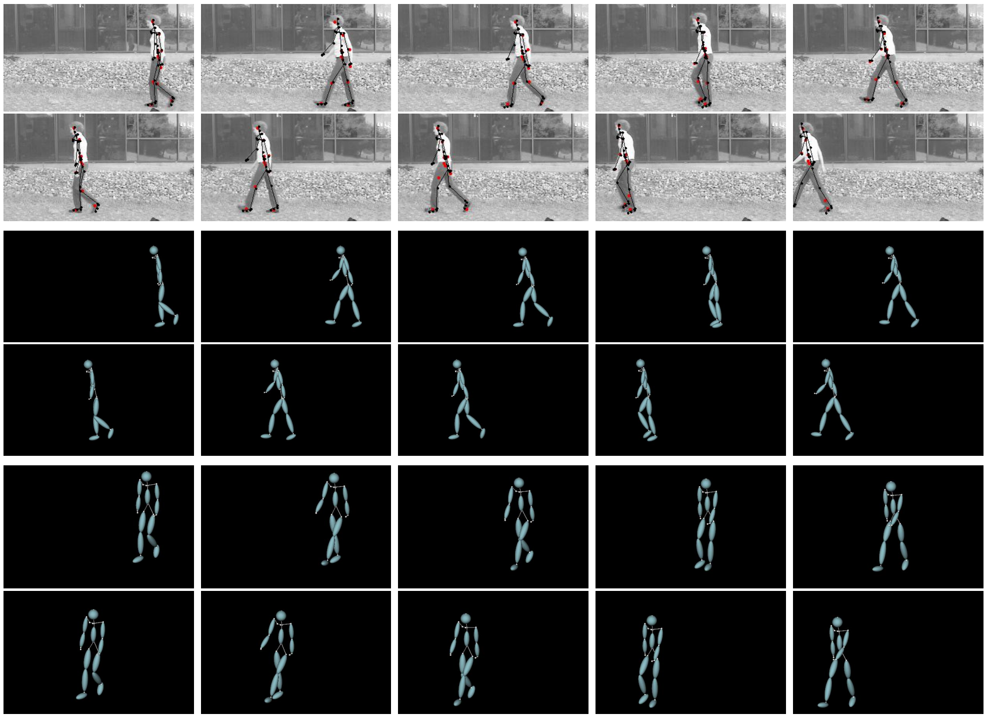


Figure 6.6: Tracking 63 frames of a walking, with noisy and missing data. The skeleton of the recovered 3D model is projected onto the images. The WSL tracks are shown in red. Note the accuracy of the result.

Algorithm 1 Optimization Strategy (at each time step t)

```
$\{\mathbf{x}_{t+\tau}\} \leftarrow \mu_X(\mathbf{x}_{t+\tau-1}) = \mathbf{X}_{out}^T \mathbf{K}_X^{-1} \mathbf{k}_X(\mathbf{x}_{t+\tau-1})$
$\{\mathbf{y}_{t+\tau}\} \leftarrow \mu_Y(\mathbf{x}_{t+\tau}) = \mu + \mathbf{Y}^T \mathbf{K}_Y^{-1} \mathbf{k}_Y(\mathbf{x}_{t+\tau})$
$\{\mathbf{g}_{t+\tau}\} \leftarrow 2\mathbf{g}_{t+\tau-1} - \mathbf{g}_{t+\tau-2}$
for $n = 1 \ldots iter$ do
  $\{\mathbf{X}'_t\} \leftarrow \min \mathcal{E}$ with respect to $\mathbf{X}'_t$
  $\{\phi_{t:t+\tau}\} \leftarrow \min \mathcal{E}$ with respect to $\phi_{t:t+\tau}$
end for
$\{\mathbf{X}'_t\} \leftarrow \min \mathcal{E}$ with respect to $\mathbf{X}'_t$
```

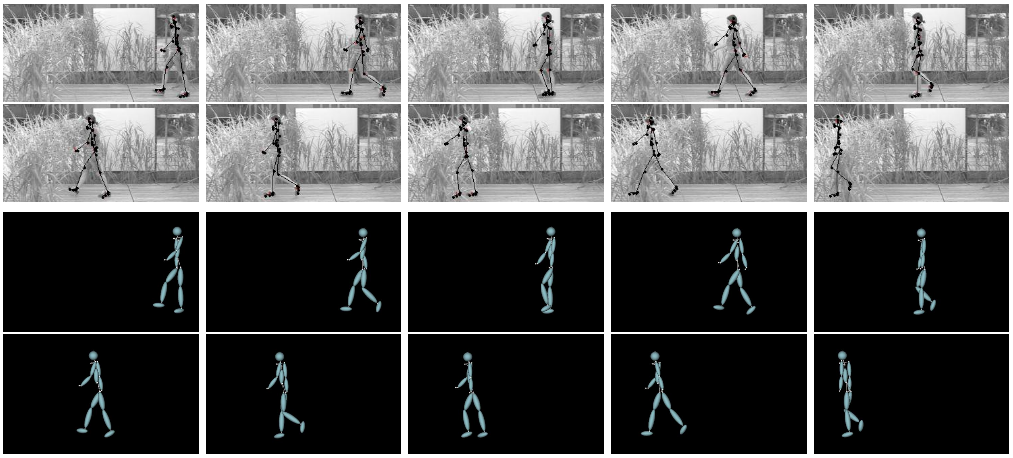


Figure 6.7: Tracking 56 frames of a walking motion with an almost total occlusion (just the head is visible) in a very clutter and moving background. Note that the prediction when the occlusion happens is a realistic motion.

One can also significantly speed up the minimization when one knows that the motion of the tracked object is very similar to the training motions. In that case, one can assume that there is negligible uncertainty in the reconstruction mapping, and therefore a pose is directly given by $\mathbf{y} = \mu_Y(\mathbf{x})$. This reduces the reconstruction likelihood for every pose to $\frac{D}{2} \ln \sigma^2(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|^2$, and the state at time t to $\phi_t = (\mathbf{g}_t, \mathbf{x}_t)$, which can be optimized straightforwardly.

6.3 Tracking Results

Here we focus on tracking different styles and speeds for the same activity. We use the Balanced GPDM model shown in Fig. 6.3 for tracking all walking sequences below. In Fig. 6.6 we use a well-known sequence to demonstrate the robustness of our algorithm to data loss. In the first frame, we supply nine 2D points—the head, left shoulder, left hand, both knees and feet, and center of the spine (the root). They are then tracked automatically using WSL [67]. As shown in Fig. 6.5(d) the tracked points are very noisy; the right knee is lost early in the sequence and the left knee is extremely inaccurate. By the end of the sequence the right foot and left hand are also lost. Given such poor input, our algorithm can nevertheless recover the correct 3D motion, as shown by the projections of the skeleton onto the original images.

While better image measurements can be obtained for this sequence, this is not always an option when there are occlusions and image clutter. E.g., Fig. 6.7 depicts a cluttered scene in which the subject becomes hidden by a shrub; only the head remains tracked by the end of the sequence (see Fig. 6.5(e)). For these frames only the global translation is effectively constrained by the image data, so the GPDM plays a critical role. In Fig. 6.7, note how the

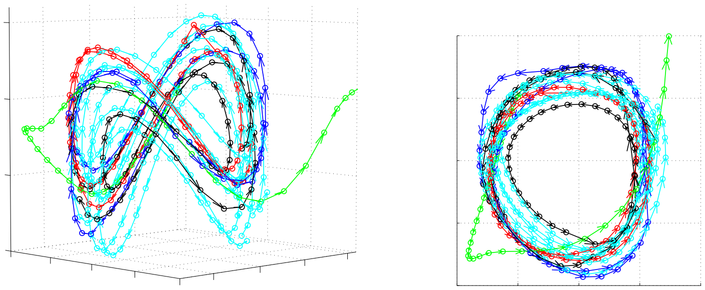


Figure 6.8: **Tracked Latent Positions:** Side and top views of the 3D latent space show the latent trajectories for the tracking results of Figs. 6.6, 6.7, 6.9, and 6.10 are shown in red, blue, black, and green. The learned model latent positions are cyan.

projected skeleton still appears to walk naturally behind the shrub.

Figure 6.9 shows a sequence in which the subject is completely occluded for a full gait cycle. When the occlusion begins, the tracking is governed mainly by the prior.[3] The 3D tracker is then switched back on and the global motion during the occlusion is refined by linear interpolation between the 3D tracked poses before and after the occlusion. Before an occlusion, it is very important to have a good estimation of $\mathbf{x}$, as subsequent predictions depend significantly on the latent position. To reduce the computational cost of estimating the latent positions with great accuracy, we assume perfect reconstruction, i.e., $\mathbf{y} = \mu_Y(\mathbf{x})$, and use the second algorithm described in Section 6.2.

The latent coordinates obtained by the tracker for all of the above sequences are shown in Fig 6.8. The trajectories are smooth and reasonably close to the training data. Further, while the training gait period was 32 frames, this three sequences involve gait periods ranging from 22 to 40 frames (by comparison, natural walking gaits span about 1.5 octaves). Thus the prior generalizes well to different speeds.

To demonstrate the ability of the model to generalize to different walking styles, we also track the exaggerated walk shown in Fig. 6.10. Here, the subject's motion is exaggerated and stylistically unlike the training motions; this includes the stride length, the lack of bending of the limbs, and the rotation of the shoulders and hips. Despite this the 3D tracker does an excellent job. The last two rows of Fig. 6.10 show the inferred poses with a simple character, shown from two viewpoints, one of which is quite different from that of the camera. The latent coordinates obtained by the tracker are shown in Fig. 6.8; the distance of the trajectory to the training data is a result of the unusual walking style.

---

[3] We manually specify the beginning and end of the occlusion. We use a template matching 2D detector to automatically re-initialize WSL after the occlusion, as shown in Fig 6.5(c).

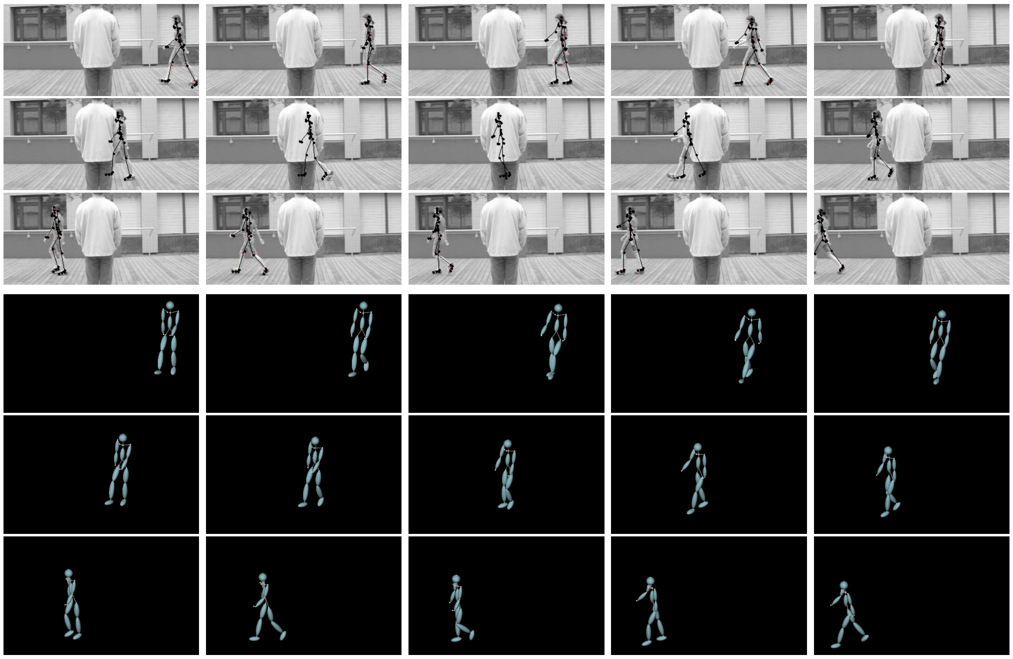


Figure 6.9: Tracking 72 frames of a walking motion with a total occlusion. During the occlusion the tracker is switch off and the mean prediction is used. Note the accuracy of the tracking before and after the occlusion and the plausible motion during it.

6.4 Conclusions and Future Work

We have introduced the GPDM for learning smooth prior models of human pose and motion for 3D people tracking. We showed that GPDM priors, unlike SGPLVM (chapter 5) ones, can be learned from modest amounts of training motions including stylistic diversity. Further, they are shown to be effective for tracking a range of human walking styles, despite weak and noisy image measurements and significant occlusions. The quality of the results, in light of such a simple measurement model attest to the utility of the GPDM priors.

As will be shown in chapter 7, the pose dynamical priors introduced here result in better tracking results than the motion and pose priors proposed in chapters 4 and 5 respectively, when working under challenging conditions such as occlusions or big stylistic deviation from the training data.

Future work will focus on building better appearance models, as this is a principal weakness of the tracker. One problem with GPs concerns the computation that grows quickly with the number of training samples. So, we plan to investigate sparsification methods so that larger training sets can be used. Another open question concerns how one should learn models with many different activities.

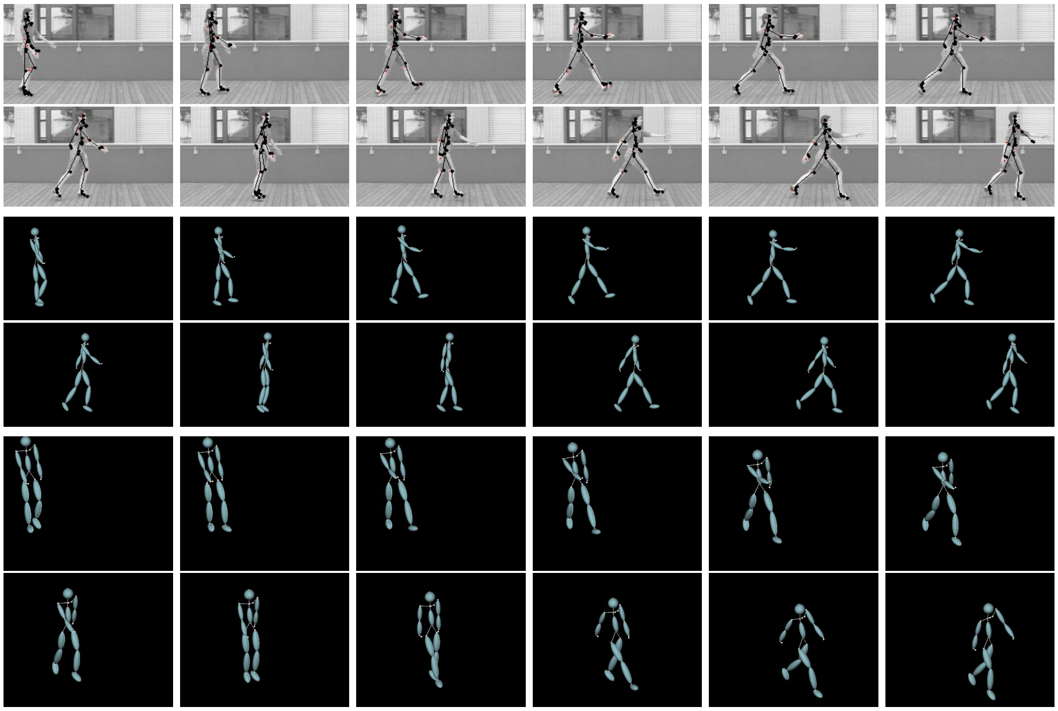


Figure 6.10: Tracking 37 frames of an exaggerated gait. Note that the results are very accurate even though the style is very different from any of the training motions. The last two rows depict two different views of the 3D inferred poses of the second row.

7 Comparative Results

In this work we have made a number of choices when learning GP models. In this chapter we examine their influence, more specifically we focus on different possible parameterizations that could have been used to parameterize a pose, the amount of training data necessary to obtain a good generalization, and the influence of the dimensionality of the input and latent spaces. The goal is to obtain non-sparse clustered spaces, where the poses performed at the same phase of the motion are aligned. A sparse model is not good for tracking since it does not generalize well; only poses close to the training data are possible. On the contrary, it is useful for recognition. We want a clustered model, so that the training data form a single manifold. This will result in functions easier to minimize when tracking. When dealing with multiple manifolds it is difficult to perform a transition between them, for example when the subject changes the style of the walking.

We then compare tracking results of the different motion and pose models for synthetic and real data. The evaluation criterions were the estimated 3D joint locations, the 2D projections, and the obtained Euler angles. Finally, we study the robustness of the models to the sparsity of the image data.

7.1 Learning GP

In this section we discuss the issues related to learning pose and motion models with Gaussian Processes. First, we discuss the influence of the different positional and angular parameterizations, and the amount of training data necessary to have a good generalization. We explore the consequences of varying the input space dimensionality, learning 80D and 20D models. We then study different dimensionalities of the latent space, showing 2D and 3D models.

7.1.1 Is parameterization an issue?

A pose can be parameterized in many different ways that can be classified as either positional or angular, which can result in very different models. This is due to the fact that each parameterization implies a very different metric and a different input dimension. Note that learning different parameterizations is equivalent to learning different Warped GP [134], where the warping functions are the conversion between the different parameterizations.

Positional parameterizations: Joints are parametrized in terms of their 3D spatial locations. They represent reality in the most natural way, since they do not have any of

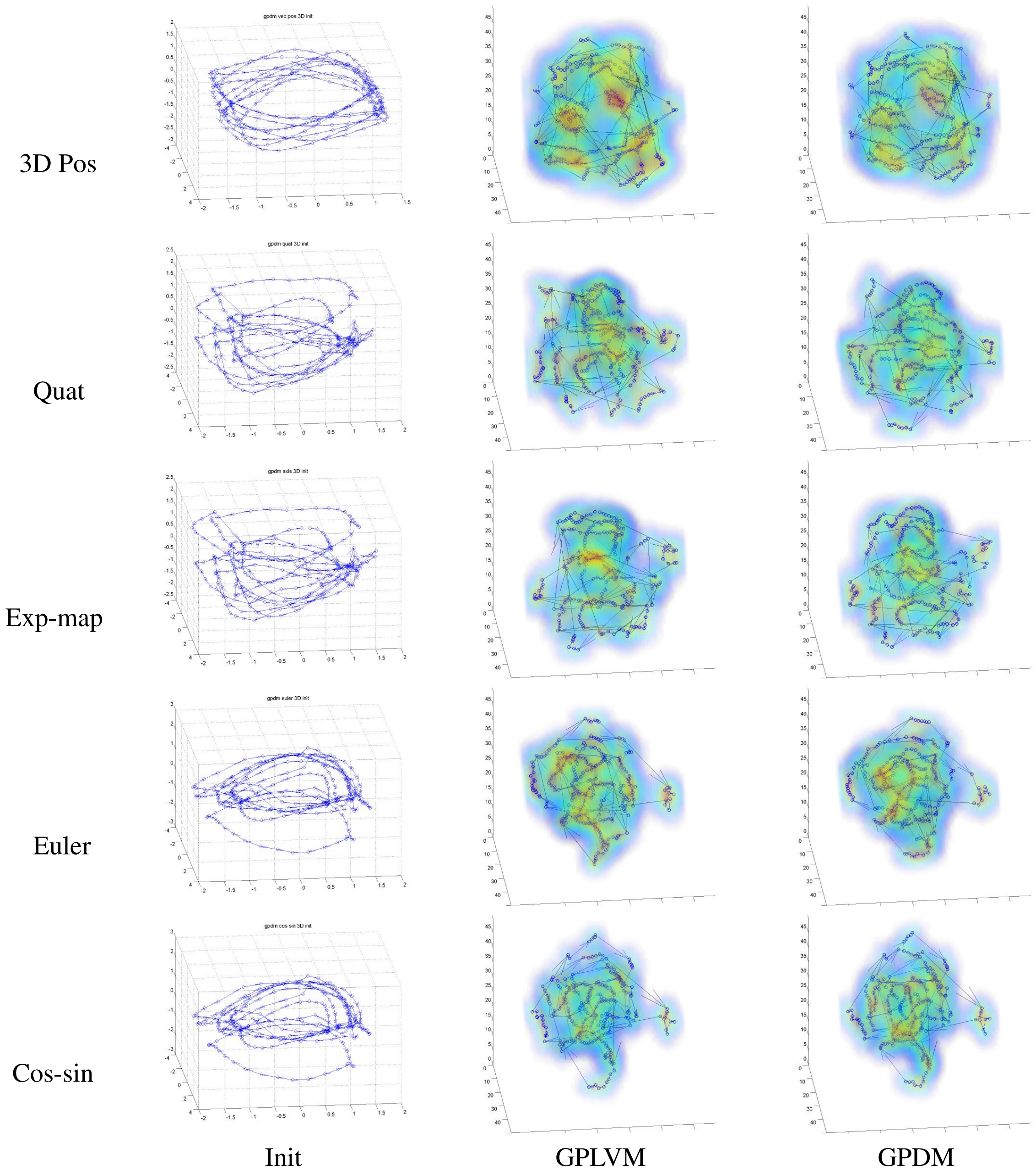


Figure 7.1: **GPLVM and GPDM for different parametrizations** seen from the same viewpoint. They were learned from a database compose of 9 walking cycles of 9 different persons (6 male and 3 female) walking at 4 km/h in a treadmill. The latent positions are initialized using PCA (first column). Due to the high dimensionality of the data, the learned GPLVM (second column) and GPDM (last column) models have non smooth latent trajectories. Because the input space is very high dimensional (80D), the reconstruction term of GPDM is overfitting, the dynamical one has almost no effect. This results in similar models to the GPLVMs.

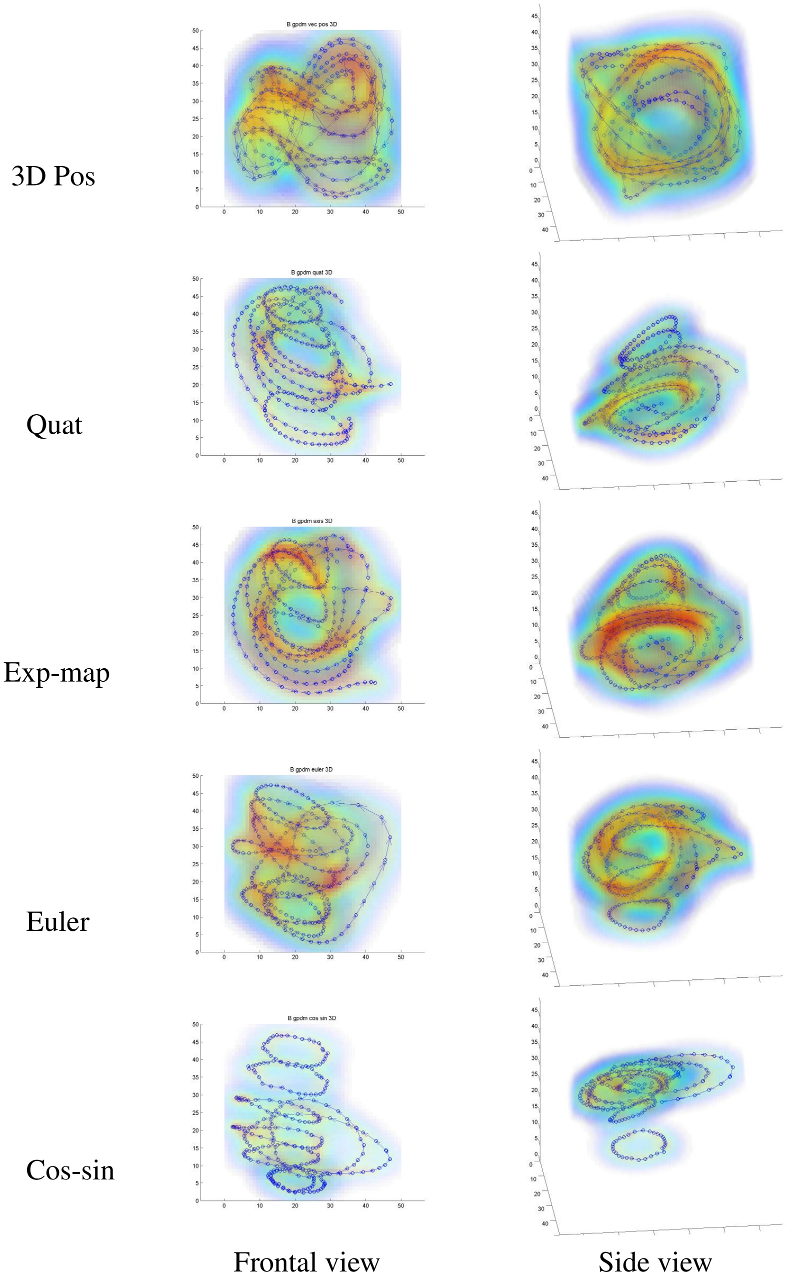


Figure 7.2: **Balanced GPDM for different parameterizations** learned from the 9 people database, seen from two viewpoints. The latent coordinates were initialized with the ones depicted by the left column of Fig. 7.1. Note the clustering corresponding to the positional parametrization of the top row. The exponential map and the quaternion models are very similar. The Euler angle is the simplest parametrization, but also the one with the largest artifacts. The cos-sin is the sparsest one since it has double the degrees of freedom of the exponential map or Euler-angle parameterizations.

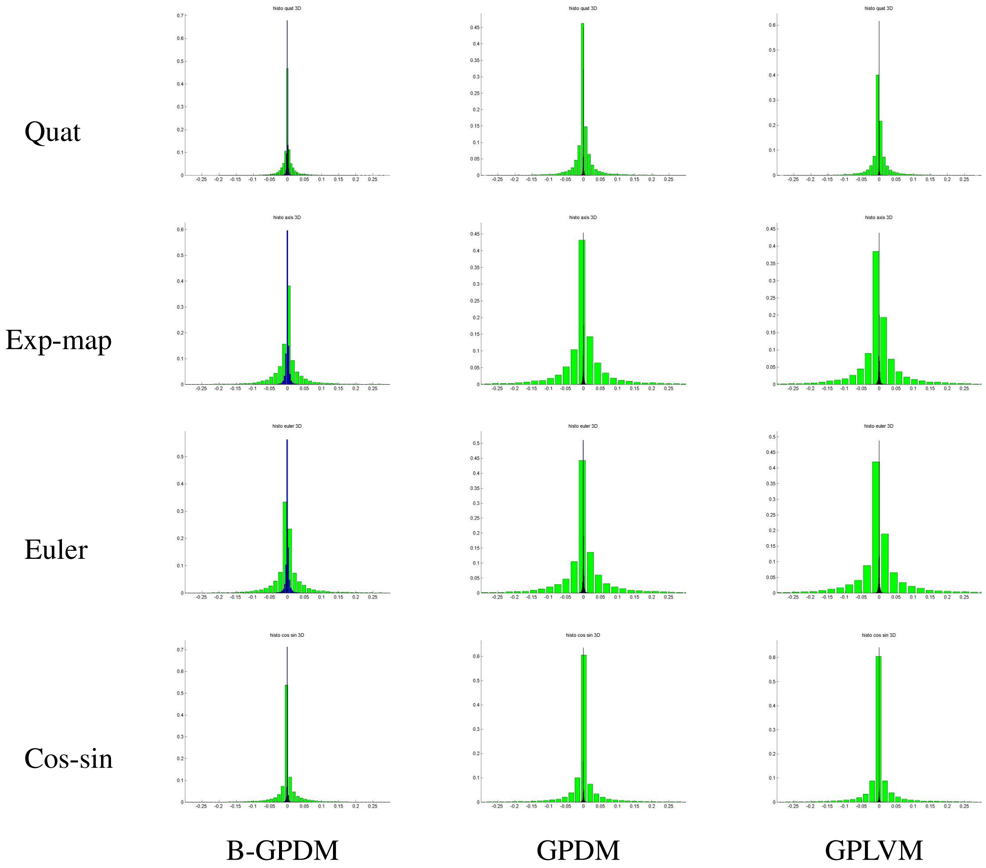
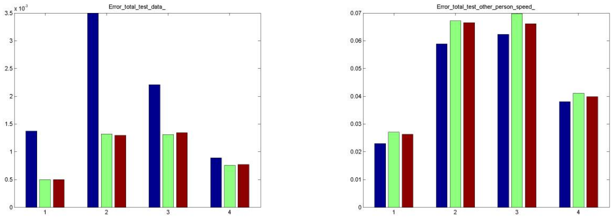


Figure 7.3: **Error histograms for the Balanced GPDM, GPDM and GPLVM** of Figs. 7.1 and 7.2. The errors, defined as $||\mathbf{y} - \mu_Y(\mathbf{x}))||$, are depicted in blue for the training data and in green for the test data. The test sequences are walkings performed by the same 9 subjects as before, but at a different speed (5km/h) than the training ones (4km/h). Note that the B-GPDM generalized better; The error histograms for the test data are narrower than the ones of GPDM and GPLVM.

train data	quat	exp-map	euler	cos_sin
B_gpdm	0.00121	0.00320	0.00355	0.00096
gpdm	0.00050	0.00089	0.00116	0.00059
gplvm	0.00044	0.00116	0.00117	0.00049

same person	quat	exp-map	euler	cos_sin
B_gpdm	0.01049	0.02672	0.02801	0.01925
gpdm	0.01664	0.04120	0.04192	0.02463
gplvm	0.01656	0.04170	0.04320	0.02480



Training

Other speed

Figure 7.4: **Mean Errors** for the 9 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponential map, Euler angle and cos-sin parameterizations. For each parametrization, three error bars are depicted, one for each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** Training data. **Right:** Test sequences are walkings performed by the same 9 subjects, but at a different speed (5km/h) than the training ones (4km/h). Note that the GPDM and GPLVM are overfitting (bigger errors for testing), and the training errors of the B-GPDM are much bigger since the dynamics is minimized assuming a narrower prior than the GPDM. The mean errors for training and testing are very small.

the artifacts that the angular parameterizations have, such as singularities, periodicity and non-euclidean distance. The resulting Balanced GPDM models are the most clustered[1], as shown in Figs. 7.2, 7.6 and 7.10. However, in practice, this parametrization is not very useful since a different model should be learned for each specific skeleton size. Moreover, it does not constrain the length of the limbs to remain constant for a given subject when doing regression, and in particular when fitting to image data. This happens even for the training data due to noise. To use the resulting models as a prior for tracking, additional hard constraints should be added to force the length of the limbs to remain constant.

Angular parameterizations: Quaternions, Euler angles, and exponential maps are examples of this type of parametrization. By definition, they guarantee that limb lengths remain constant when doing regression. Although some retargetting would be needed to express them in a common skeleton [154], they can be considered as independent of the limb sizes, and one can learn a single model common to all possible sizes of a given skeleton. But these parameterizations suffer from a variety of problems. They are hierarchical, since the position of a joint depends on the specific rotations of all its parents. As a consequence two vectors $\mathbf{y}_i$, $\mathbf{y}_j$ can be very different but represent very similar poses. Their distance metric is not euclidean, for example the distance between two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ is defined as

$$dist(\mathbf{q}_1, \mathbf{q}_2) = 2\cos(|\mathbf{q}_1 \mathbf{q}_2|) \ . \tag{7.1}$$

They present singularities, especially the Euler angles that suffer from the Gimbal lock[2] problem [161]. Moreover, they are not unique; Two sets of 3 different Euler angles may produce the same rotation matrix. Euler angles are also periodic, with period 2π. This introduces artifacts since this periodicity is not modeled with the RBF kernel used in our GP models. Agarwal and Triggs [4] suggested to replace each angle θ_i, by $(\cos(\theta_i), \sin(\theta_i))$ to get rid of the periodicity problem. This increases by a factor of 2 the dimensionality of the input data, which is already very high ($\approx$ 80D). The rotation matrix parametrization was not considered here since it increases by a factor of 3 the dimensionality of the input space, making learning even more difficult. Exponential maps are not periodic but they suffer a discontinuity at $2\pi n$, with $n > 0$. Quaternions are also not periodic and they do not have discontinuities, but their dimensionality is higher than Euler angles and exponential maps.

Fig. 7.1 depicts GPLVM and GPDM models learned from a database of 9 people (6 male and 3 female) walking on a treadmill at 4 km/h for the different parameterizations. The latent positions are initialized using PCA, and are depicted by Fig. 7.1 (left). Since the poses are very high dimensional (more than 80D), the reconstruction term in the GPDM (right) is clearly overfitting, resulting in a model very similar to the GPLVM (center). Note that both contain non smooth latent trajectories. By contrast, the Balanced GPDMs of Fig.

---

[1] *Clustered* in this context means forming in a common manifold

[2] *Gimbal lock* occurs when the second angle is +90 or -90 degrees, causing the first and third axis of the linkage to become aligned. The linkage then has two rotational degrees of freedom rather than three.

7.2, that were learned from the same database, contain very smooth latent trajectories for every parametrization.

The 3D positional parametrization depicted by the first row of Fig.7.2 yields the most clustered model. As could be expected, the exponential map and the quaternion models have very similar latent positions, but the quaternion one is sparser since the dimensionality of the input space is higher. A sparse model is a model that do not generalize well, since the pose and motions far from the training data have a very low probability.

The Euler angle is the simplest parametrization that one can use, but the corresponding model exhibits the largest artifacts, resulting in poses performed at the same phase of the motion by the different subjects being far apart. The cos-sin parametrization yields the sparser model since the input space has double the number of degrees of freedom of the exponentila map or Euler-angle ones.

One can conclude that the exponential map parametrization is the best parametrization since, unlike the positional ones, it does not require additional constraints, it is not too sparse, and the poses performed at the same phase of the motion are aligned for most of the subjects. Three of the motions that compose the database were noisy and relatively different from the other six. As depicted by Fig. 7.6, learning a database containing only clean motions produces models with aligned latent trajectories that are more suitable for tracking.

7.1.2 Generalization of the different GPs

To study the generalization of the different GPs, we compute the *reconstruction errors* for the training and test sequences. For each pose $\mathbf{y}$, first its corresponding latent position is initialized to $\mathbf{x}_i$, where $\mathbf{y}_i$ is the closest pose to $\mathbf{y}$. Then, $\hat{\mathbf{x}}$ is obtained by minimizing Eq. 6.26 with respect to $\mathbf{x}$. The error for each pose is computed as $||\mathbf{y} - \mu_Y(\hat{\mathbf{x}})||$, with $\mu_Y(\mathbf{x})$ defined as in Eq. 6.24.

Fig. 7.3 depicts the normalized error histograms for the different models of Figs. 7.1 and 7.2. They are depicted in blue for the training data and in green for the test data. The test sequences are walkings performed by the same 9 subjects, but at a different speed (5km/h) than the training ones (4km/h). Fig. 7.4 depicts the mean error for each type of parametrization and model. The errors for the training are very small, and the Balanced GPDM has the largest ones, while the GPDM and GPLVM performed similarly. The GPDMs and GPLVMs are overfitting resulting in bigger errors than the Balanced GPDMs for the test sequence. The parameterizations are not directly comparable with each other since they have different scales [3]. One can conclude that the Balanced GPDM generalizes the best, and this will result in better tracking performance as will be seen in section 7.2.

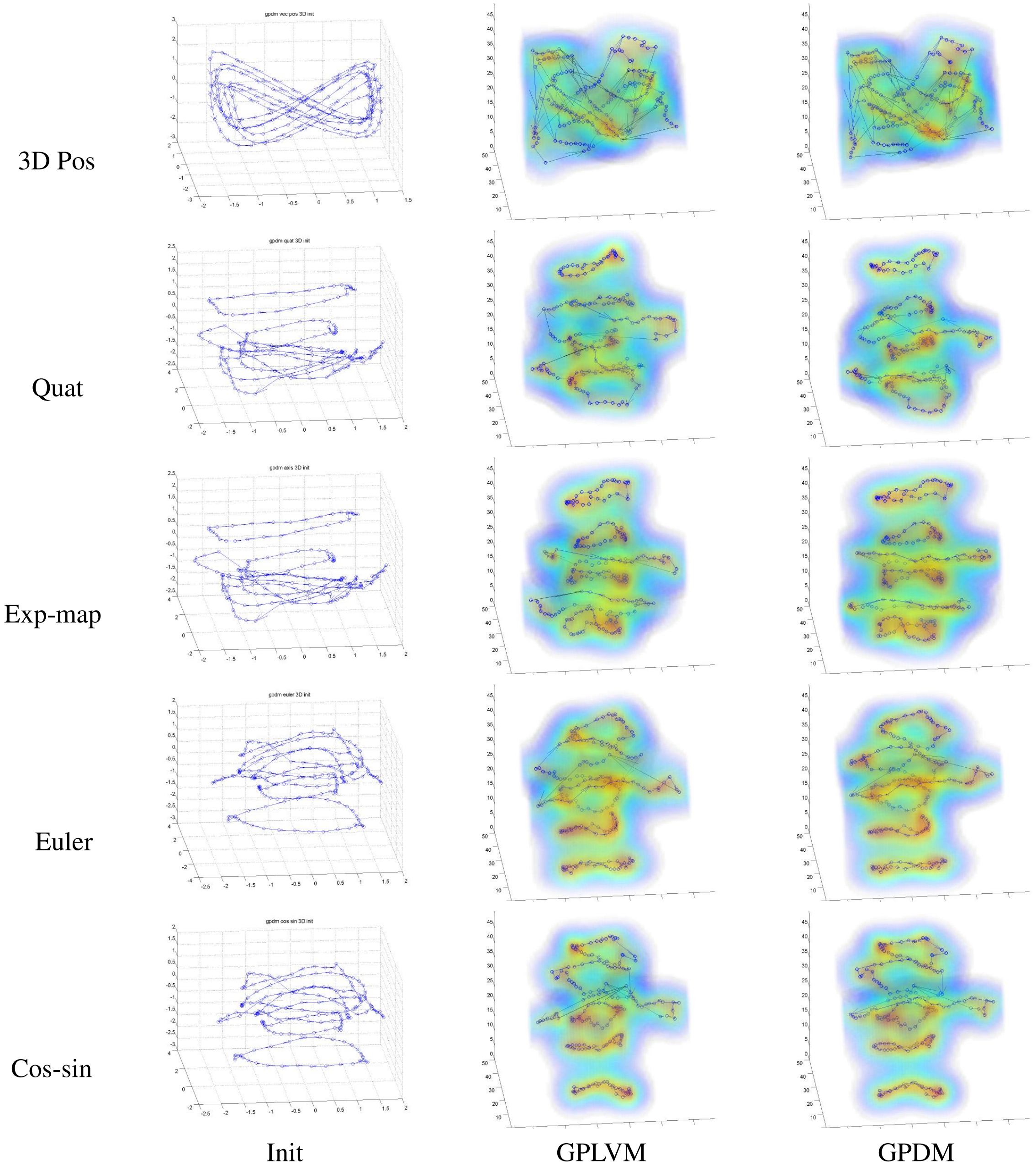


Figure 7.5: **GPLVM and GPDM for different parametrizations** seen from the same viewpoint. PCA was used for initialization (left). As for the 9 people walking database of Fig. 7.1, due to the high dimensionality of the data, the learned GPLVM (center) and GPDM (right) models have non smooth latent trajectories, resulting in models less suitable for tracking than the B-GPDM of Fig. 7.6

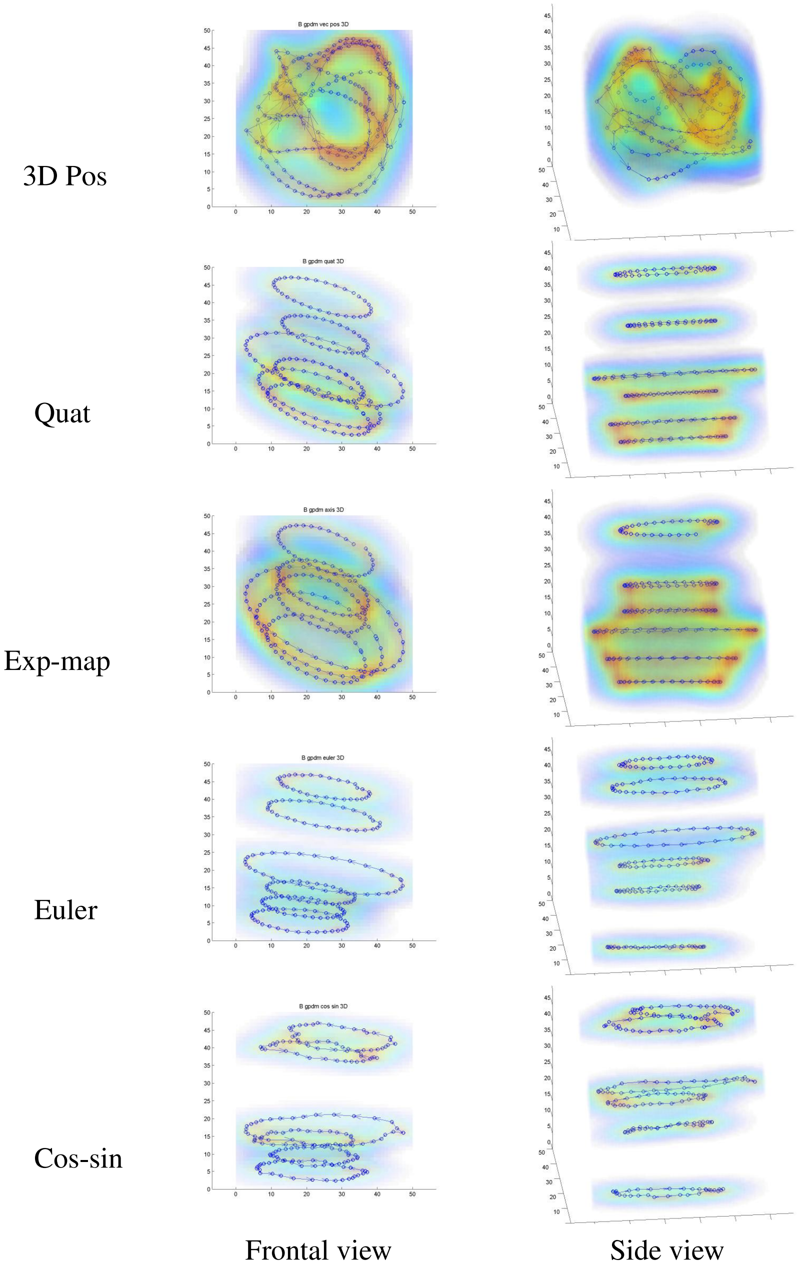


Figure 7.6: **Balanced GPDM for different parameterizations**, learned from the 6 people database, view from two different viewpoints. The PCA coordinates depicted by Fig. 7.5 (left) were used as initialization. As for the 9 people database, the positional parametrization is the most clustered, the exponential map model is similar to the quaternion one and the cos-sin yields the most sparse model. Note that the artifacts of the Euler angle parametrization are less visible here, the cycles are well represented as ellipses within a common cylindrical structure. The content is represented as an ellipse an the style (subject identity) as the length axis of the cylinder. These models are more suitable for tracking since poses performed at the same phase of the motion are aligned.

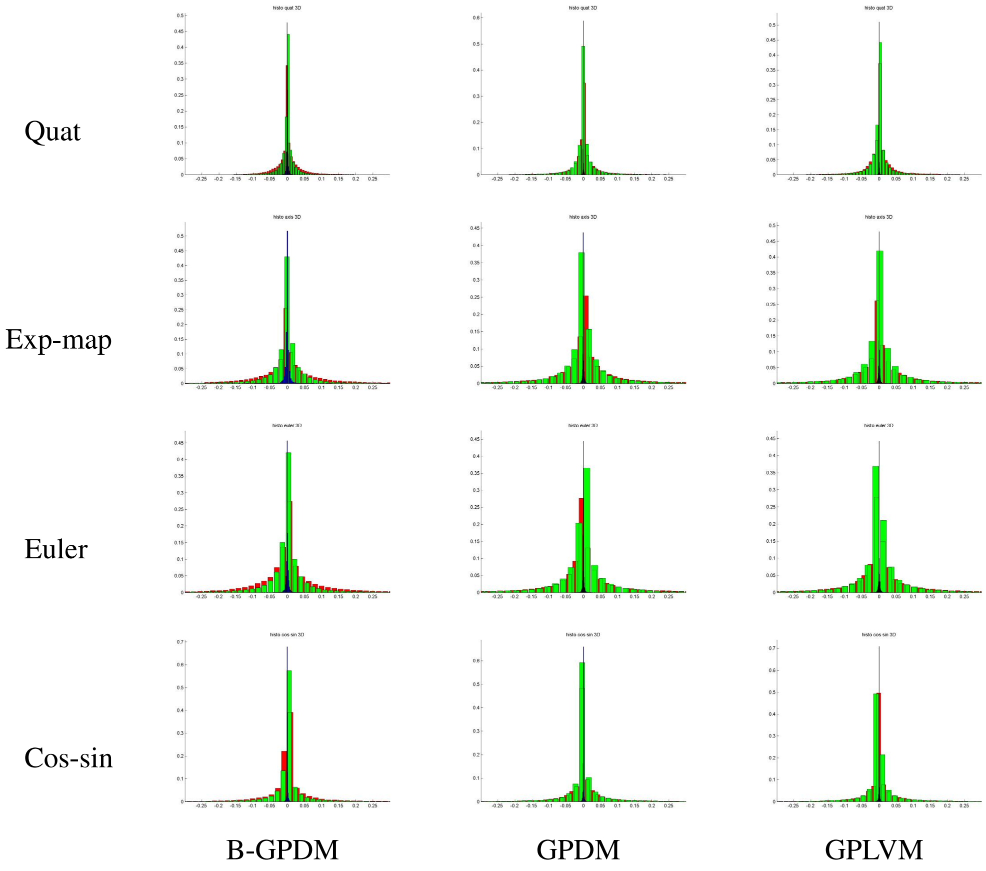
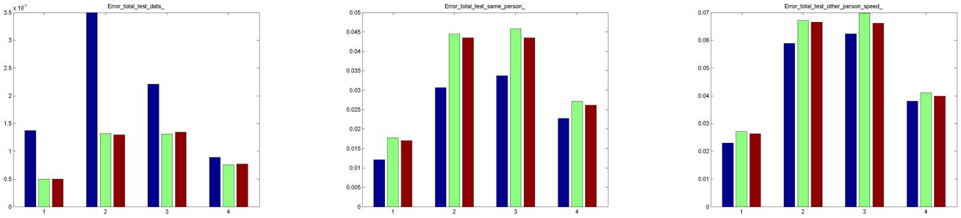


Figure 7.7: **Error histograms** for the Balanced GPDM, GPDM and GPLVM of Figs. 7.5 and 7.6. The errors are depicted in blue for the training data, in green for test data performed by the same subject at different speeds ranging from 3 to 7 km/h, and in red for test data performed by 3 different subjects (whose motions are not part of the database) at different speed ranging from 3 to 7 km/h. Note that the B-GPDM generalized better, the error histograms for the test data are narrower than the ones of GPDM and GPLVM.

train data	quat	exp-map	euler	cos_sin
B_gpdm	0.00137	0.00350	0.00221	0.00089
gpdm	0.00050	0.00132	0.00131	0.00076
gplvm	0.00050	0.00130	0.00134	0.00077

same person	quat	exp-map	euler	cos_sin
B_gpdm	0.01210	0.03065	0.03371	0.02278
gpdm	0.01775	0.04444	0.04581	0.02715
gplvm	0.01700	0.04350	0.04351	0.02617

other person	quat	exp-map	euler	cos_sin
B_gpdm	0.02301	0.05883	0.06228	0.03809
gpdm	0.02706	0.06716	0.06973	0.04109
gplvm	0.02633	0.06654	0.06613	0.03991



Training other speed other person

Figure 7.8: **Mean Errors** for the 6 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponential map, Euler angle and cos-sin parameterizations. For each of parametrization, three error bars are depicted, one of each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** training data, **Center:** walking cycles performed by the same subjects but with different speeds ranging from 3 to 7km/h. **Right:** walking cycles of 3 subjects whose motions are not part of the database, walking with speeds ranging 3 to 7km/h. Note that as for the 9 people database of Fig. 7.8 the GPDM and GPLVM are overfitting, and the B-GPDM has the worst training errors. The mean errors for training and testing are very small.

7.1.3 Generalization vs amount of training data

To study the influence of varying the amount of training data for generalization, we learned different models from a subset of the 9 people database. Figs. 7.5 and 7.6 depict GPLVM, GPDM and Balanced-GPDM models for the different parameterizations when learning 6 people (3 male and 3 female) walking on a treadmill at 4km/h. Note that the artifacts of the Euler angle parametrization are less visible here, the cycles are well represented as ellipses within a common cylindrical structure, where the latent coordinates for the poses performed by all the subjects at the same phase of the motion are aligned. This is not surprising since this was in part the behavior (Fig. 6.4) for databases composed of walking cycles at incremental speeds for the same subject, where the speed was encoded by the radius of the ellipses. One can expect that the other hidden variable, that is, the style of the motion, is encoded by a third dimension. This makes our approach comparable to methods that try to separate style an content [42]. Here, the content is represented as an ellipse and the style as the longitudinal axis of the cylinder. The models are sparse, and present a desirable property for classification, one dimension encodes the subject identity. This is not the case anymore when learning a database composed of 9 subjects (Fig. 7.2). Part of the problem is due to the fact that the motions of those 3 new subjects are noisy and are quite different from the other 6. Moreover, as will be described below, this results in the models with the better tracking performances.

Figs. 7.7 and 7.8 depicts error histograms and mean errors for the training data, walking cycles performed by the same 6 subjects but at different speeds ranging from 3 to 7km/h, and walking cycles performed by the other 3 persons that compose the 9 subject database and whose motions are not included in the training set. Note that as expected the Balanced GPDM has the biggest reconstruction errors for the training data, but the GPDM and GPLVM are overfitting and their errors are bigger than the Balanced GPDM for the test data, even in the case of the walking cycles performed by the same training subjects (but at different speeds).

When learning a database containing only two different walking cycles performed by two different subjects, the behavior is different, the three different GP models overfit, as shown by the error histograms and the mean errors of Figs. 7.11 and 7.12. Note that the Euler angle Balanced GPDM pull apart the different walkings (Fig. 7.10), resulting in a model suitable for classification. The GPDM and GPLVM models of Fig. 7.9 are very similar.

7.1.4 Dimensionality of the input space

Figs. 7.2–7.9 depict results when learning 80D spaces that are very high dimensional. Note that only the positional parameterization results in clustered and non-sparse models. The models learned from the 6 subject database are the best ones, since they are clustered and poses that are produced in similar phases of the different motions are aligned. But these models are sparse.

---

[3] The norm of a rotational quaternion is always 1, the value of each Euler angle is between 0 and 2π, the norm of an exponential map is between 0 and 1.

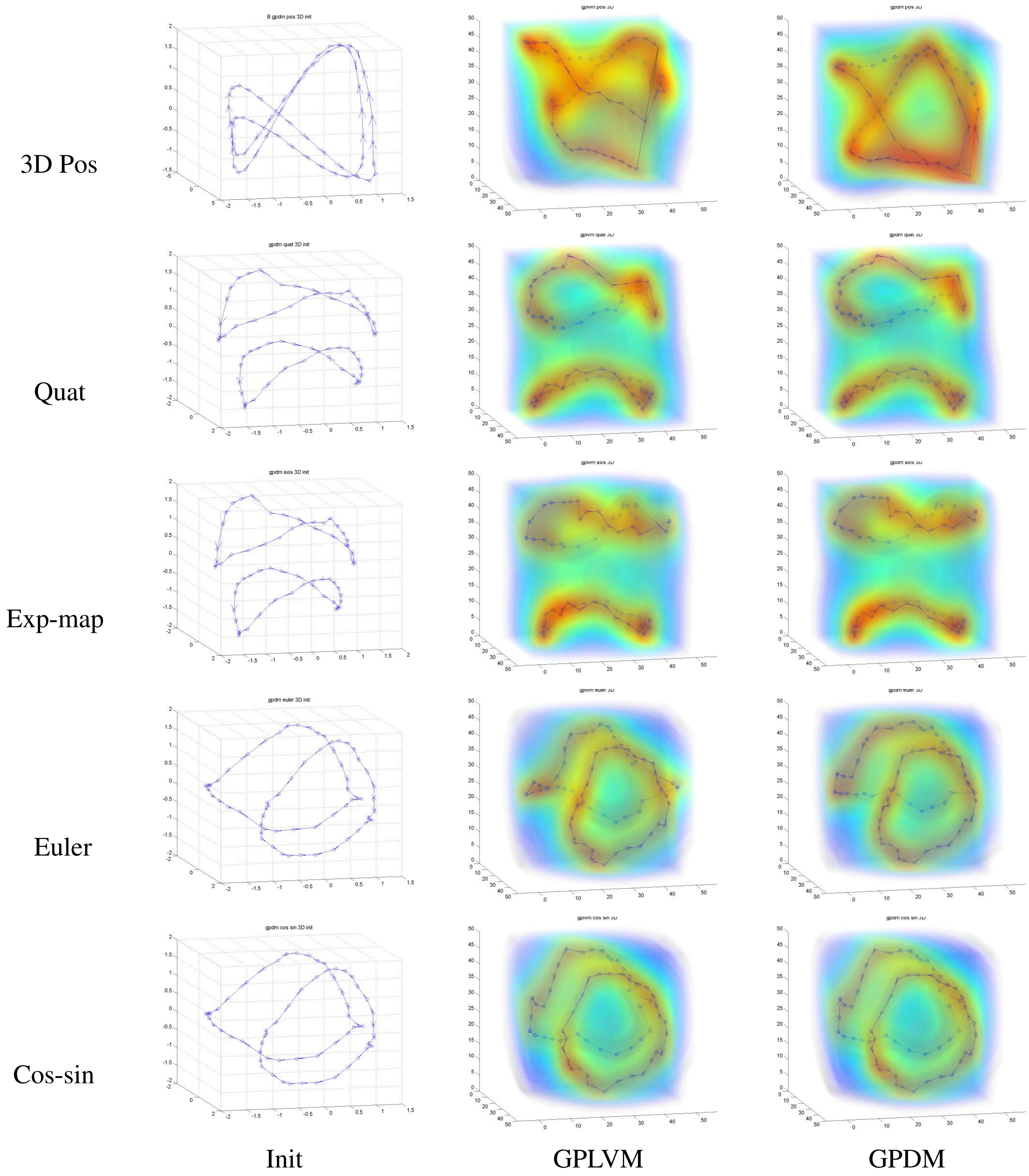


Figure 7.9: **GPLVM and GPDM for different parametrizations**, learned from a database composed of 2 subjects (1 male, 1 female) walking on a treadmill at 4km/h. Because of the small amount of training data, the GPLVM (center) and (GPDM) models have smooth latent trajectories.

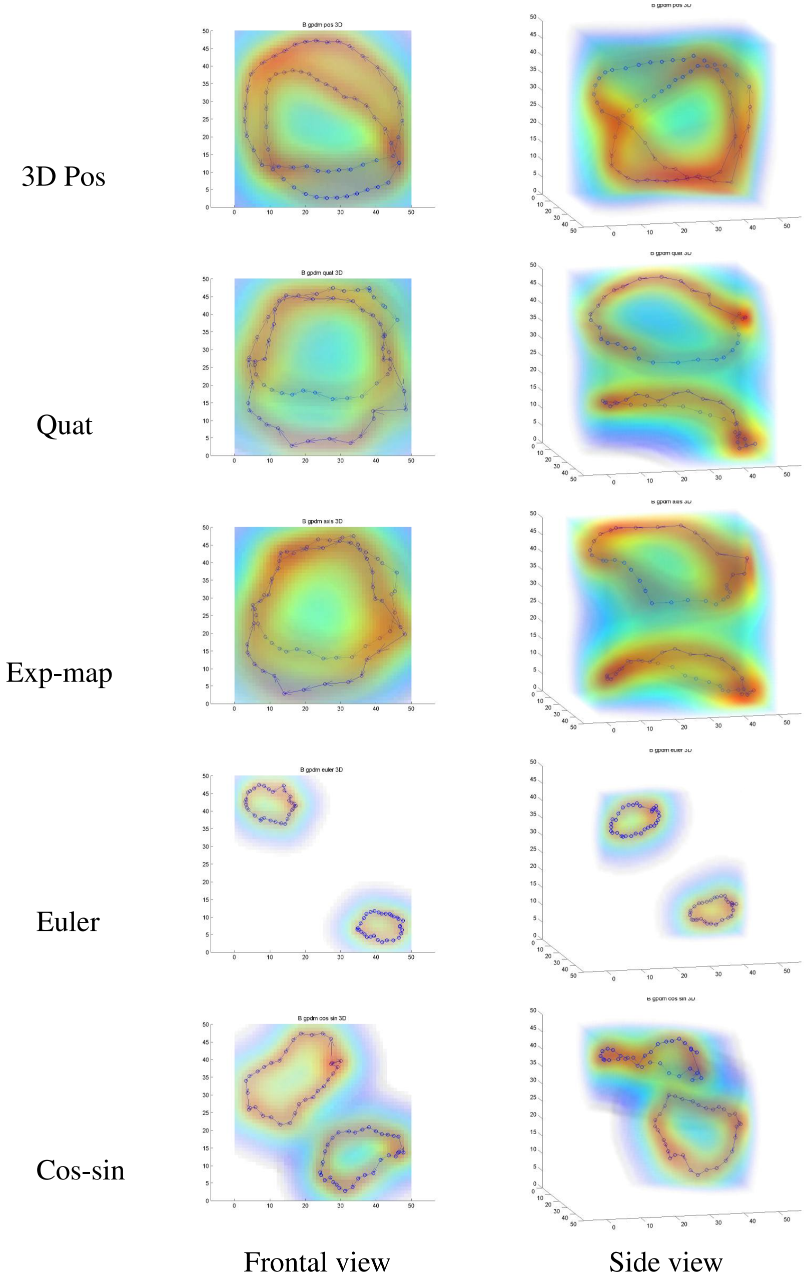


Figure 7.10: **B-GPDM for different parameterizations for** the 2 subjects database, view from two different viewpoints. Note that the learned models vary a lot depending on the type of parametrization used. The positional one yields once more the most clustered model, and the exponential map model is quite similar to the quaternion one. Because the PCA initialization (Fig. 7.9) has a small loop, the B-GPDM pull apart the two walkings. This model could be useful for recognition, since the poses from different subjects are clearly separated.

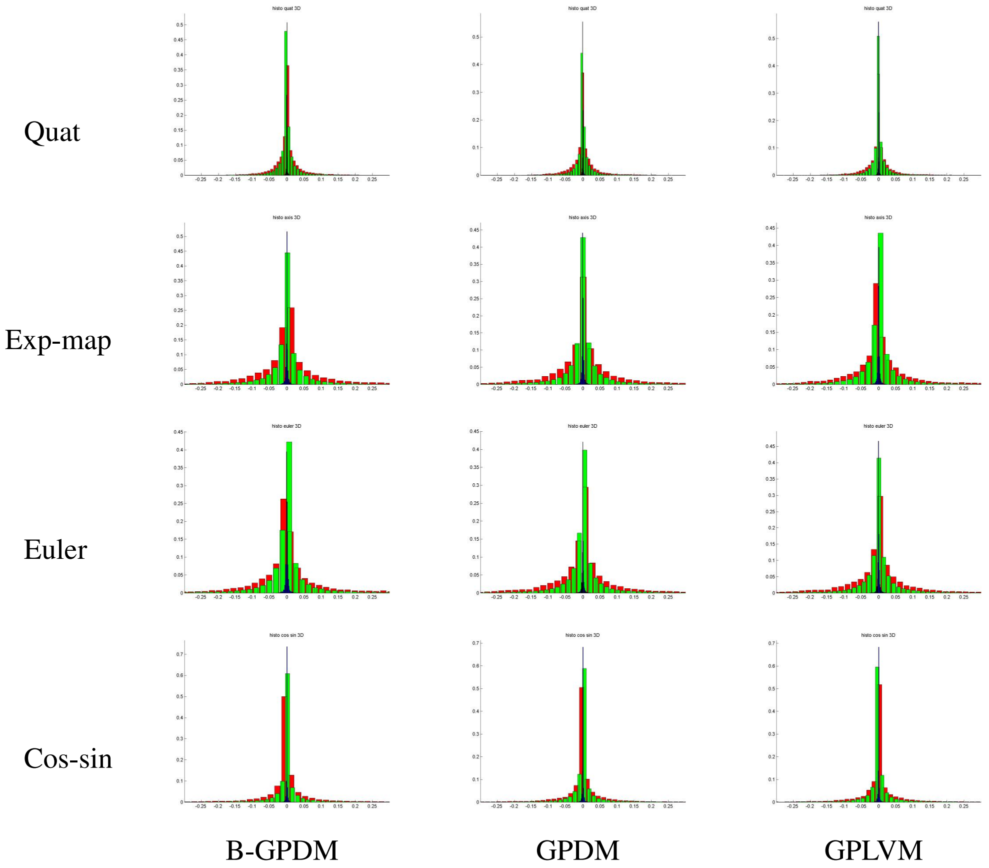
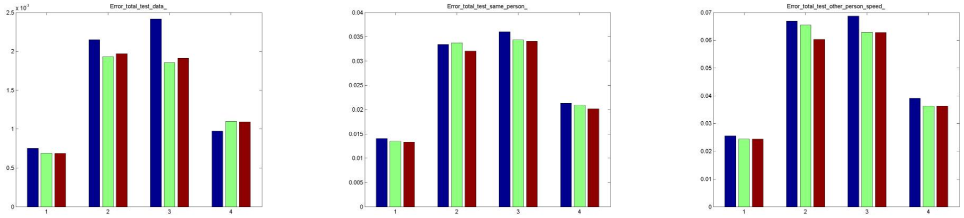


Figure 7.11: **Error histograms** for the Balanced GPDM, GPDM and GPLVM of Figs. 7.9 and 7.10. The errors are depicted in blue for the training data, in green for test data performed by the same subject at different speeds ranging from 3 to 7km/h, and in red for test data performed by 3 different subjects (whose motions are not part of the database) at 4km/h. Due to the small amount of training data, the errors of the B-GPDM, GPDM and GPLVM are similar.

train data	quat	exp-map	euler	cos_sin
B_gpdm	0.00075	0.00215	0.00242	0.00097
gpdm	0.00069	0.00193	0.00185	0.00110
gplvm	0.00069	0.00197	0.00191	0.00109

same person	quat	exp-map	euler	cos_sin
B_gpdm	0.01405	0.03345	0.03607	0.02129
gpdm	0.01350	0.03375	0.03434	0.02095
gplvm	0.01335	0.03209	0.03405	0.02015

other person	quat	exp-map	euler	cos_sin
B_gpdm	0.02555	0.06687	0.06867	0.03911
gpdm	0.02443	0.06546	0.06289	0.03633
gplvm	0.02443	0.06035	0.06277	0.03634



Training other speed other person

Figure 7.12: **Mean Errors** for the 2 people database. Each plot is divided in 4 groups, representing the errors for the quaternion, exponetial map, Euler angle and cos-sin parameterizations. For each parametrization, three error bars are depicted, one of each model: B-GPDM (blue), GPDM (green) and GPLVM (red). **Left:** training data, **Center:** walking cycles performed by the same subjects but with different speeds ranging from 3 to 7km/h. **Right:** walking cycles of 3 subjects whose motions are not part of the database, walking at a speed of 4km/h. Note that all the models are overfitting and have similar reconstruction errors for the test sequence.

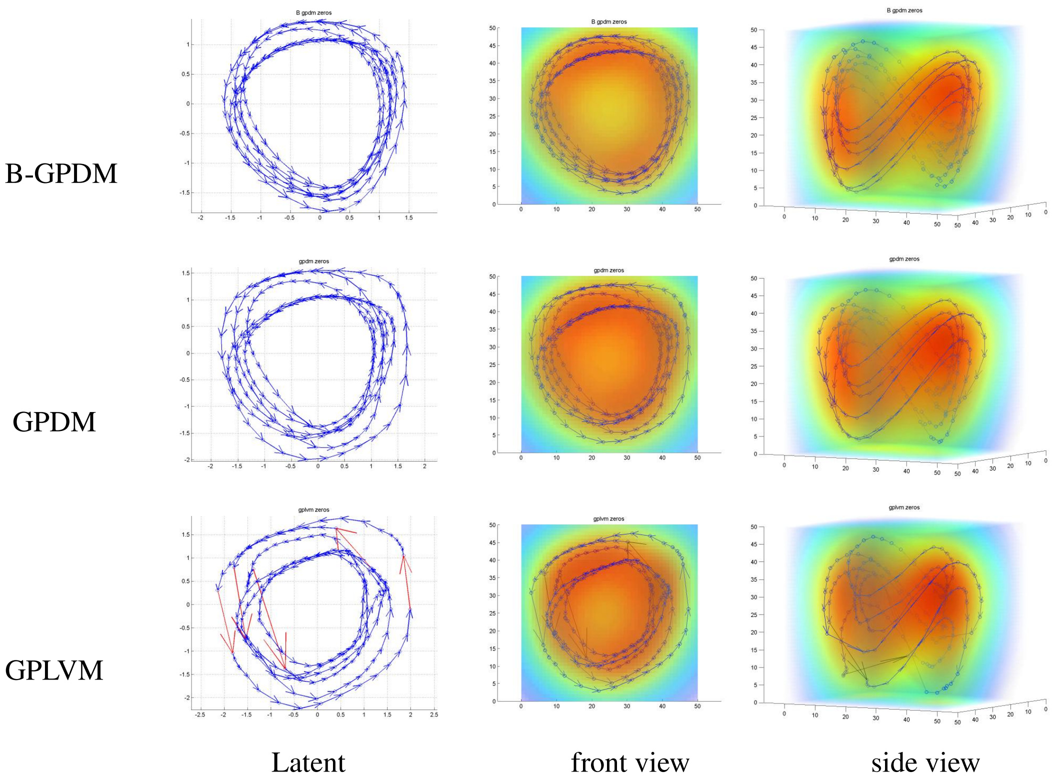


Figure 7.13: **GP Models for a 20D input space**. The training data is composed of 6 subjects walking at 4km/h on a treadmill. Note that the models clustered, and the poses that correspond to the same phase of the motion for the 6 subjects are aligned.

To learn more clustered aligned non-sparse models with angular parameterizations, one can decrease the dimensionality of the input space mitigating some artifacts. 20D Balanced GPDM, GPDM and GPLVM for the 6 subject database are shown in Fig. 7.13. Note that the model is non sparse, clustered, and the poses from different subjects at similar motion phases are aligned. These are desirable properties for tracking since with a small change in the latent position one can change the style of a pose without changing the phase of the motion. Only the GPLVM results in non smooth latent trajectories, since the model is 20D, the fraction D/d is small, and the reconstruction overfitting of the GPDM is much smaller than in the 80D case. Note that the 20D Balanced GPDM was used to produce the tracking results shown in chapter 6.

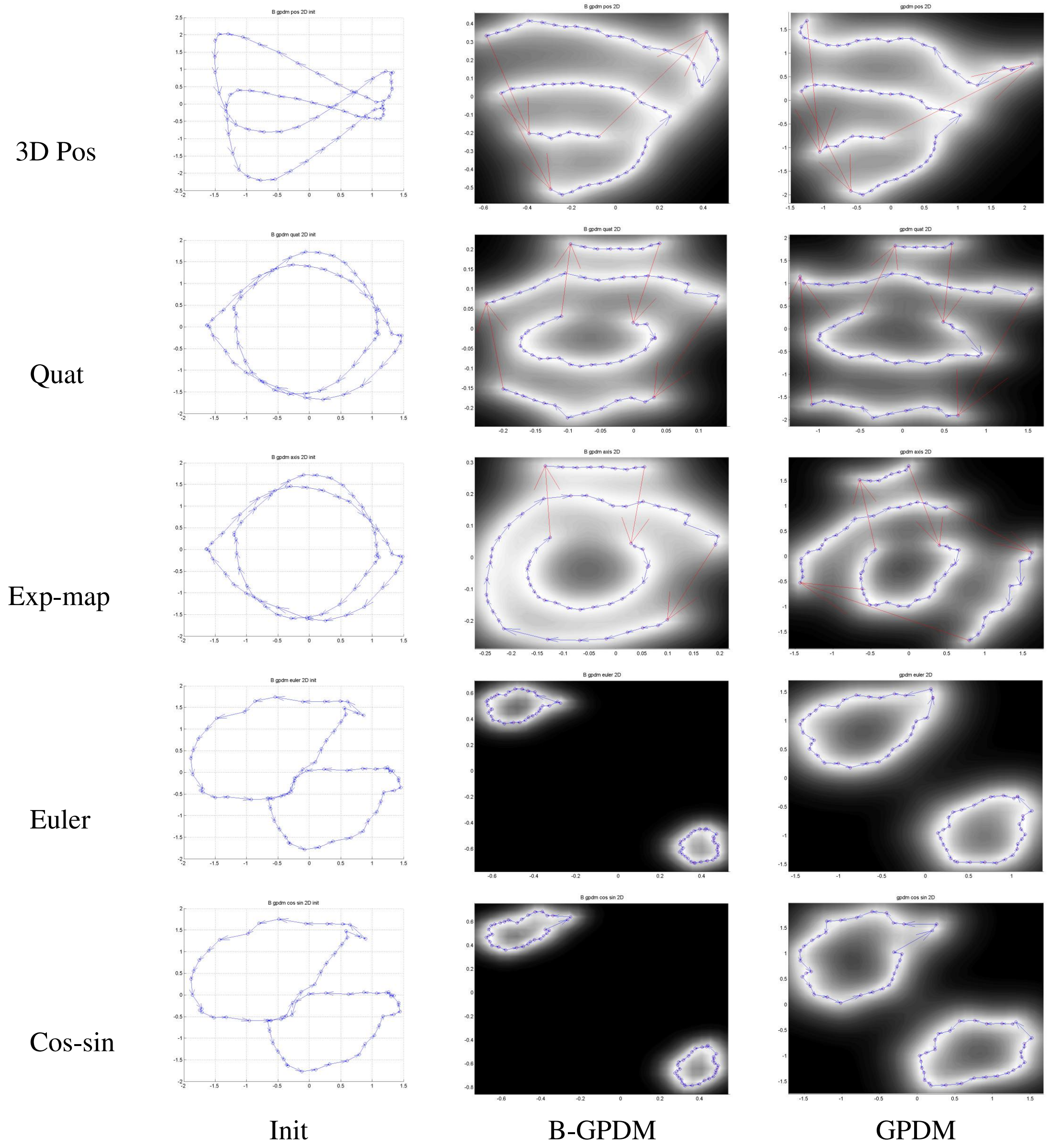


Figure 7.14: **2D Balanced GPDM and GPDM for different parametrization**. When the initialization contains loops the B-GPDM and GPDM result in non smooth latent trajectories. The jumps in the latent space are depicted in red. To avoid these discontinuities, one should change the initial conditions, raise the dynamics likelihood prior to a much stronger power than D/d, include back-constraints, model suitable priors for $\mathbf{X}$, or increase the dimensionality of the latent space. The 3D models are depicted in Figs. 7.9 and 7.10.

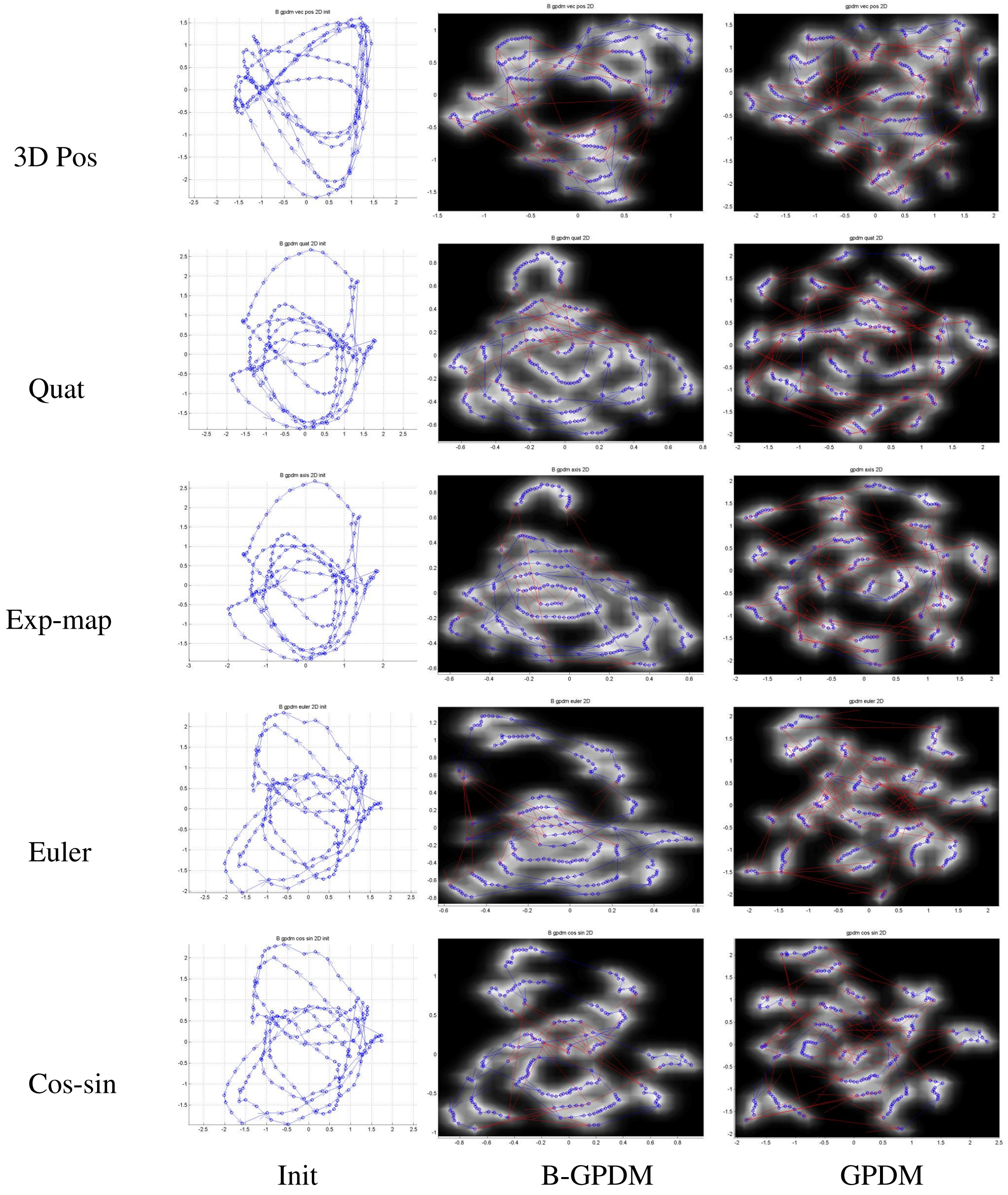


Figure 7.15: **2D Balanced GPDM and GPDM for different parameterizations**. When the initialization contains loops the B-GPDM and GPDM result in non smooth latent trajectories. The jumps in the latent space are depicted in red. None of the models result in smooth latent trajectories. The dimensionality used is not high enough and one should learn 3D models (Fig. 7.6).

7.1.5 Dimensionality of the latent space

When the number of training data is small enough a 2D model can be learned. Depending on the parameterization, the initialization contains loops, and the resulting models have non smooth latent trajectories, even when learning a Balanced-GPDM. Fig. 7.14 shows Balance GPDM and GPDM models learned from a database composed of 2 people (one male and one female) walking on a treadmill at 4km/h. To make the latent trajectories smoother, as seen in Chapter 5, one can change the initial conditions and use for example non-linear techniques, such as Isomap [141], LLE [121] or Laplacian Eigenmaps [10]. As suggested by Lawrence [77], one can incorporate back-constraints. One can also think of including proper priors over the $\mathbf{X}$, that will mitigate such problems.

Fig. 7.15 depicts 2D models for the database composed of 6 subjects. Note that for all the parameterizations, the latent trajectories are non smooth. When the amount of training data is high the easiest way to avoid these discontinuities is to increase the dimensionality, as shown in Fig. 7.6, where 3D models are depicted.

7.2 Tracking

In this section we compare tracking results using the different pose, motion and pose dynamical models proposed. We use synthetic and real data to study the robustness of the different models with respect to the decrement of the number of image cues.

7.2.1 Synthetic data

Two different sequences were used to test the performance of the methods proposed in chapters 4, 5 and 6. The first sequence is a training sequence, and the walking was performed by a female subject walking at 4km/h on a treadmill. The second one was performed by none of the subjects of the 6 and 2 people databases. Although it was performed by one of the subjects that compose the 9 people database, the test sequence was not used for learning, and its speed (3km/h) was different from the one used for training (4km/h).

We used an image size of 640x480, and a fixed camera with a calibration matrix whose values were set so that all the joints projected into the image, with the root, defined at the level of the sacroiliac, projecting at the center of the image. The 2D locations of the projected joints were used as input for the different tracking algorithms. The synthetic sequences did not contain any global motion. The reason is that we wanted to analyze the pose recovery of the different models, without any artifacts that a bad global motion estimation will produce. The real sequence that is analyzed below contained global rotation and position changes to be estimated. We used the same number of minimization steps for all the results shown here.

7.2.1.1 GP models

A temporal window of size 3 was used for all the experiments, and the models were initialized to the correct initial poses, $\mathbf{y}_{1:3}$, but to bad initial latent positions $\mathbf{x}_{1:3}$. We choose this configuration to test the performance of the different models against bad initialization.

Two set of experiments were performed. The first one minimized Eq. 6.26, and the second one minimized the same equation but assuming perfect reconstruction. It used the mean prediction (MP) as the pose estimate, so that one minimized Eq. 6.26 with respect to $\mathbf{x}$ with $\mathbf{y} = \mu_Y(\mathbf{x})$. We tested two different variances of the image likelihood error σ_e, the one used for tracking the sequences in chapter 6 ($\sigma_e = 10$), and a more realistic estimate for the noiseless synthetic image data ($\sigma_e = 1$).

Training sequence: Fig. 7.16 shows the mean tracking errors for the different models, when tracking the training sequence: The 2D projections errors (pixels), the error in the 3D location of the joints (mm), and the error in the estimated Euler angles (radians). We measure three types of error since they are not always correlated. They are depicted by Fig. 7.16. Note that in the first row the 2D projections errors for the 20D model (blue) are very different from the Euler angle ones. The 2D projections and 3D locations errors are small while the Euler angles are big. This may be caused by the artifacts of the Euler angle parametrization, such as the fact that they are not unique. Sometimes the 2D projection error and the 3D location one are different. This is due to the depth ambiguities in monocular tracking, i.e., two different 3D locations project to the same 2D point. Each plot is composed of 3 groups, representing Balanced GPDM, GPDM and GPLVM results. For each GP type, 8 bars are shown in 4 different colors. Each color corresponds to a different model: 80D (9, 6, and 2 subjects) and 20D (6 subjects) respectively. For each model, two bars are depicted in the same color, the first one represents the error when using all the 2D joint projections as input, the second one uses only a subset, the same that was used for real sequences.

For the two optimization strategies, the Balanced GPDM errors are bigger than the GPDM and GPLVM ones. This is not surprising since the data is generated from a training sequence. This behavior was already observed in section 7.1 when we study the generalization properties of the different models. The reconstruction errors for the training sequences were bigger when learning a Balanced GPDM. The tracking mean errors are small as depicted by Fig. 7.30 ranging 2 to 0.08 cm for the different models.

Latent trajectories: Figs. 7.17–7.20 depict the latent trajectories estimated when tracking the training sequence with the different models. The training data is depicted in cyan. In the top row of each figure the training sequence to track is depicted in black for the Balanced GPDM (left), GPDM (center) and GPLVM (right). In the bottom three rows, the latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$ are depicted. Four trajectories are shown for each plot. The result of minimizing Eq. 6.26 is depicted in red when using all the data and in

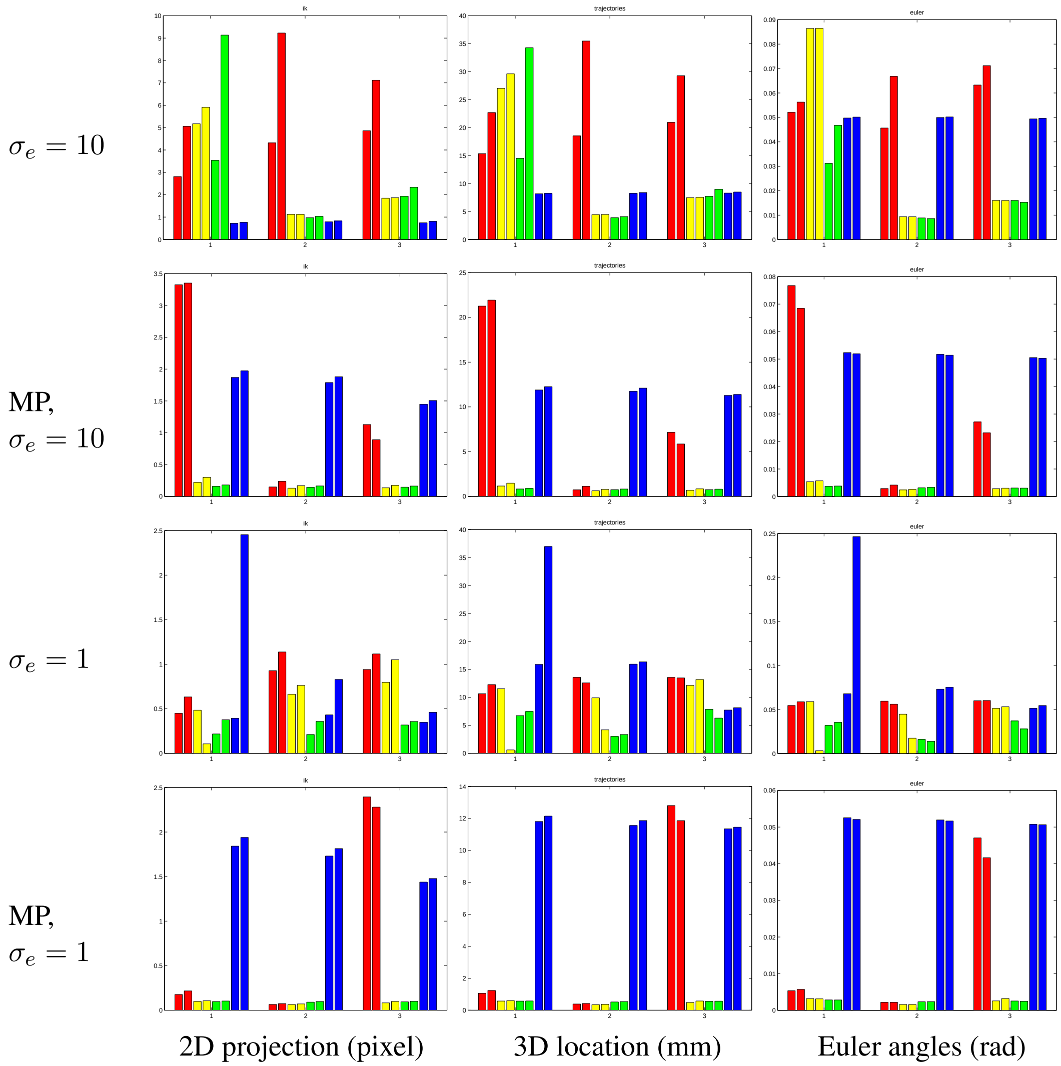
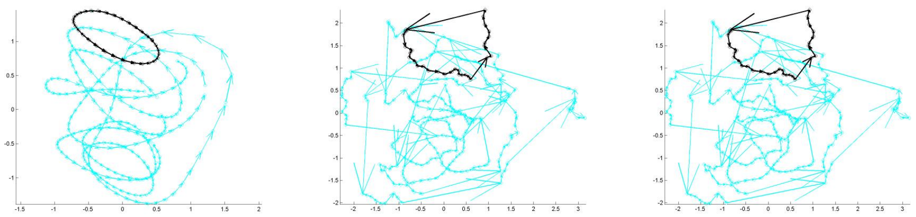
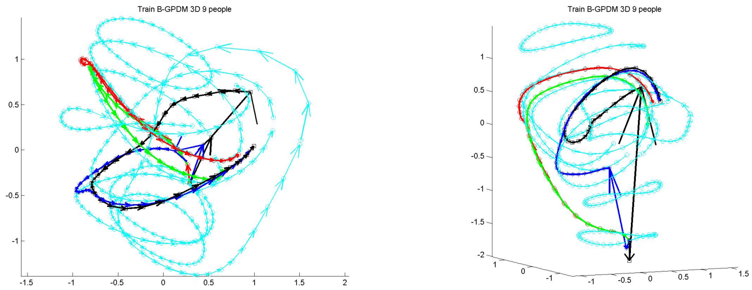


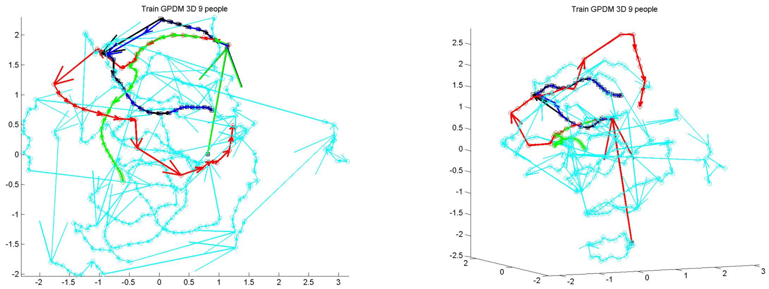
Figure 7.16: **Mean errors when tracking a training sequence with GPs**. Each plot is composed of 3 groups, representing Balanced GPDM, GPDM and GPLVM results. For each GP type, 8 bars are shown in 4 different colors. Each color corresponds to a different model: 80D (9, 6, and 2 subjects) and 20D (6 subjects) respectively. For each model, two bars are depicted in the same color, the first one represents the error when using all the 2D projection as input, the second one uses only a subset, the same that was used for real sequences. Three types of errors are depicted: The 2D projections errors (pixels), the error in the 3D location of the joints (mm), and the error in the estimated Euler angles (radians). They are not always correlated, as shown in the first row for the 20D model (blue).



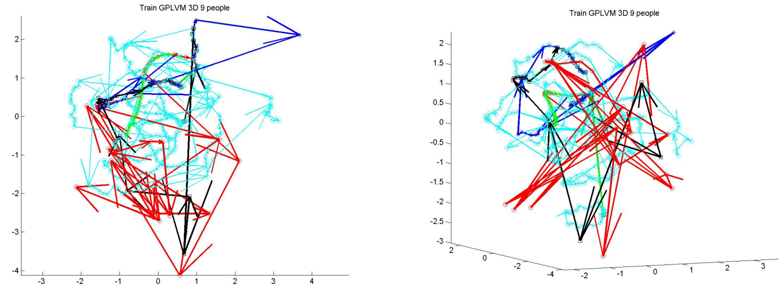
Training motion used as ground truth



Balanced GPDM

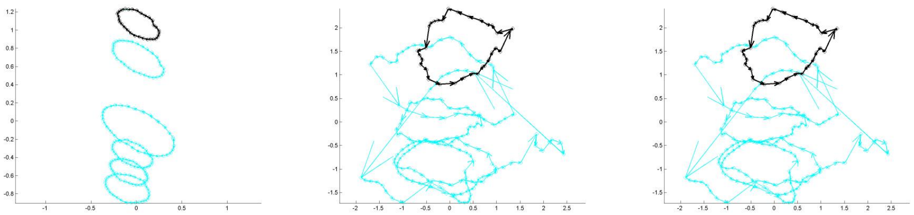


GPDM

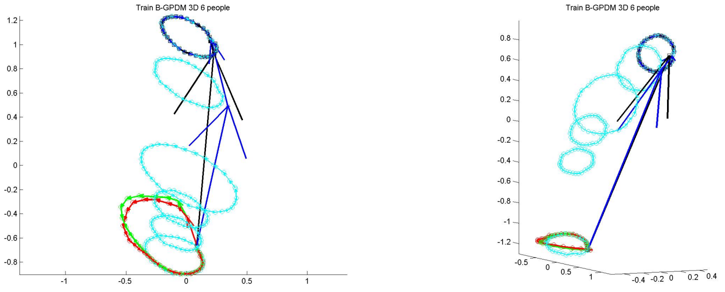


GPLVM

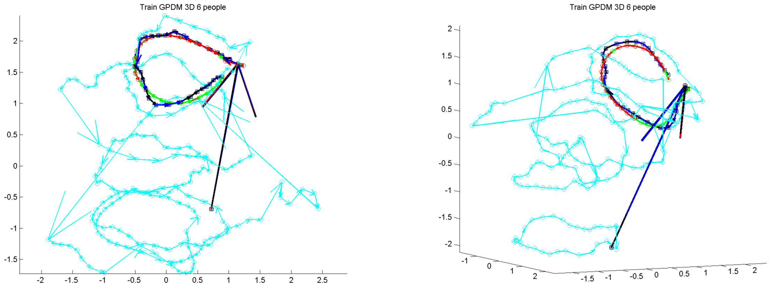
Figure 7.17: **Estimated latent trajectories for the 9 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Four trajectories are depicted for each plot. Minimizing Eq. 6.26 whole set (red) and subset (green), when MP and complete set (black) and MP and subset (blue). The latent trajectories are not well recovered for the Balanced GPDM, and non smooth for the GPLVM.



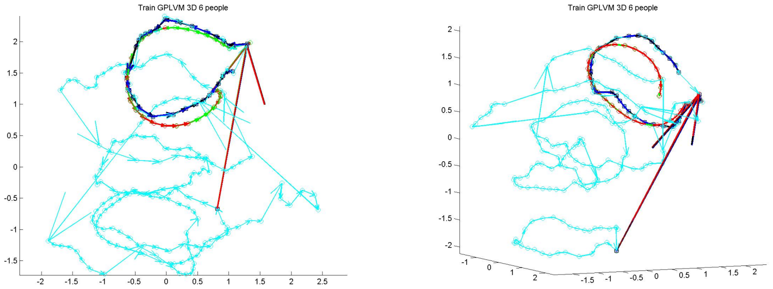
Training motion used as ground truth



Balanced GPDM

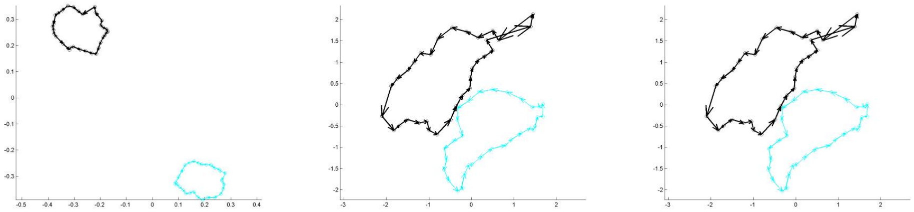


GPDM

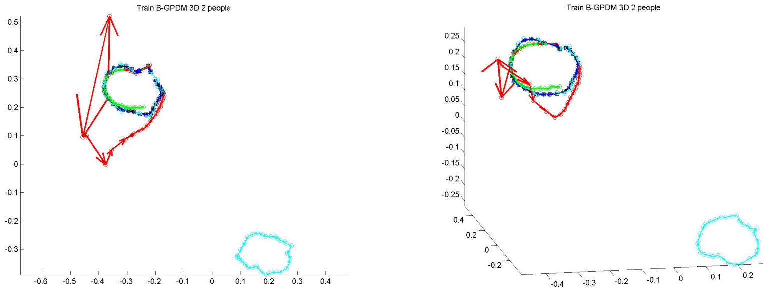


GPLVM

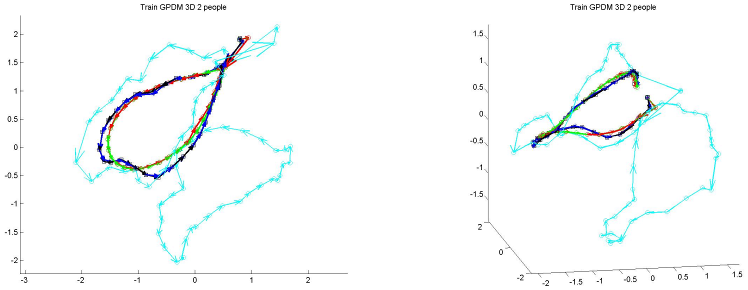
Figure 7.18: **Estimated latent trajectories for the 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories estimated when tracking with the Balanced GPDM and non assuming perfect reconstruction are not well recovered, resulting in big errors, as depicted by Fig. 7.16.



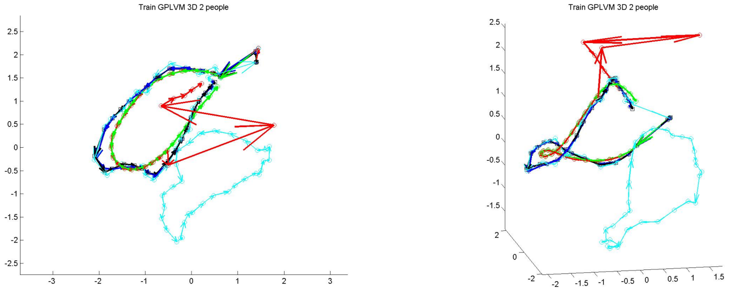
Training motion used as ground truth



Balanced GPDM

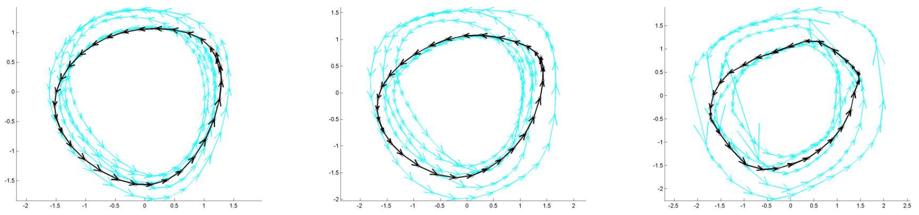


GPDM

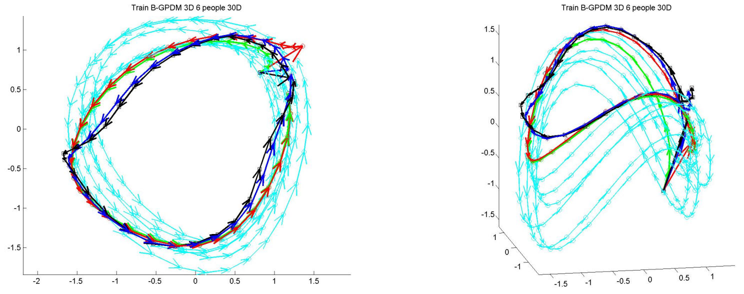


GPLVM

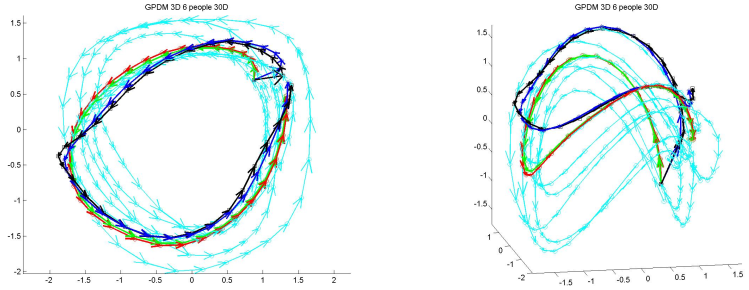
Figure 7.19: **Estimated latent trajectories for the 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories are well recovered, resulting in small errors, as depicted by Fig. 7.16.



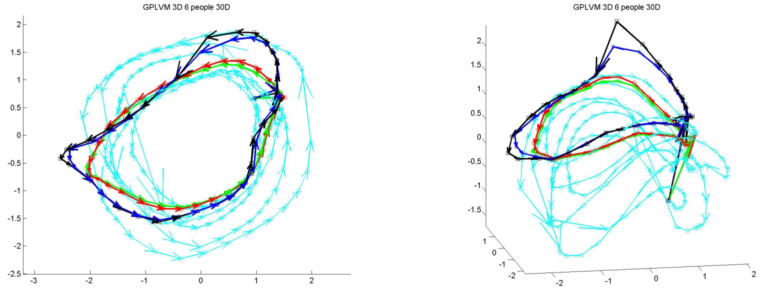
Training motion used as ground truth



Balanced GPDM



GPDM



GPLVM

Figure 7.20: **Estimated latent trajectories for the 20D 6 people database** when tracking the training sequence. The training data is depicted in cyan. **Top Row:** training sequence to track (black) for the Balanced GPDM (left), GPDM (center) and GPLVM (right). **Bottom three rows:** Latent trajectories tracked with Balanced GPDM (second row), GPDM (third row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the trajectories are well recovered, resulting in relatively small errors, as depicted by Fig. 7.16.

green when using only a subset of the 2D joints. The resulting trajectories when assuming perfect reconstruction are depicted in black when using the whole set of 2D joint projections and in blue when using only a subset. The 9 subject database has the worst results, especially in the case of GPLVM, since the poses are not aligned by the phase of the walkings, and the minimizer has to jump too many local minima to get to the correct solution. The latent trajectories when tracking with the GPLVM are not smooth, and this results in large mean errors, as depicted by Fig. 7.16. In general the trajectories are better recovered when assuming perfect reconstruction.

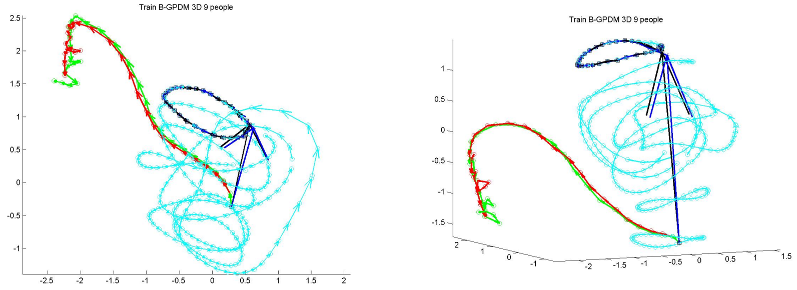
Initialization: is an important issue in tracking, especially when using a hill-climbing strategy. For the 9 subject database models the initial conditions are very important, since their motions are not aligned. Due to the bad initialization of the latent positions the tracking results are bad. This can be seen in Fig. 7.17, where the latent trajectories are depicted. They are not well recovered, making the errors to be higher than those of the other models that properly recovered them. Another example is when tracking with a Balanced GPDM learned from the 6 subject database (second row of Fig. 7.18), the latent trajectories estimated when minimizing Eq. 6.26 are not well recovered. This is once more due to the bad initialization. The dynamical prior penalizes too much the big jump necessary to get to the correct latent positions. This is not the case when using GPDM and GPLVM since the dynamical prior is not as important. Note that when tracking, one assumes an initial position close to the solution. This is clearly not the case here. Using a multi-hypothesis tracker, or a better initialization, would result in a better estimation.

Assuming perfect reconstruction: results in worst results, since without this assumption the model is free to adapt to a wider range of styles. In such case the latent positions are less accurately recovered. This is shown in Fig. 7.18 where we recovered the correct latent positions only when assuming perfect reconstruction.

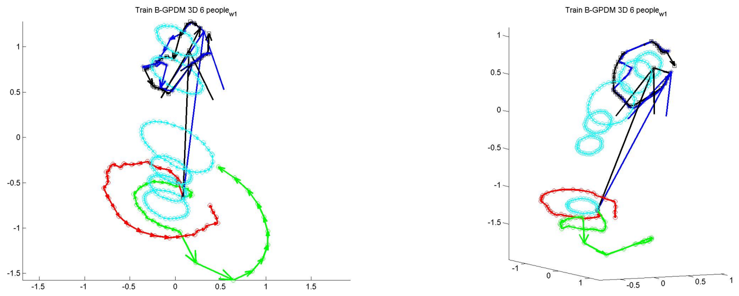
Lower dimensional input space: The 20D database was learned from an 80D input space, where 60D were constants and independent of the pose. The learning results in $w = 1$ for such dimensions, and a much small value (at least an order of magnitude) for the other 20. This results in a much narrower prior for the 60D, with the mean reconstruction for these dimensions being a constant (the same that was fixed for learning), independent of the latent position. The reason why we did not use a 20D model for tracking is that it does not have enough flexibility to fit the image data. The 20D model has more errors when assuming perfect reconstruction, since the 60D have a constant value for all the poses in the video sequence. In general the 20D database results in more errors than the 80D ones when the data is noiseless. On the contrary this model performs better under noisy and missing data. This is also the case when tracking motions very different from the training ones, since the 80D databases are more difficult to control, and may diverge. Note that when the solution has the 60D very far from the training one, because their prior is narrower, the estimation may result in latent trajectories very far from the training ones, as depicted in

Fig. 7.26. Note that although the latent trajectories are far, the error is small (see Fig. 7.22). When decreasing σ_e, this results in divergence, as depicted in the third row of Fig. 7.22.

Decreasing the image error: When σ_e decreases (last two rows of Fig. 7.16), the results are more accurate, the errors are much smaller and in the range of 0.04 to 1 cm in mean. Decreasing the variance is not a good strategy in general when dealing with real data, since, due for example to noise, the tracking may diverge and result in unfeasible positions. One of the reasons is that the recovered latent positions are not smooth anymore, as depicted by Fig. 7.21, and worst estimated when we do not assume perfect reconstruction.



(a) Training sequence



(a) Test sequence sequence

Figure 7.21: **Estimated latent trajectories when reducing** σ_e. **Top row** represents the results when using B-GPDM and the 9 people database, with a value of $\sigma_e = 1$ for the training sequence of Fig. 7.18. **Bottom row** represents the latent trajectories when using B-GPDM and the 6 people database, with a value of $\sigma_e = 1$ for the test sequence. The corresponding $\sigma_e = 10$ tracked values are shown in Fig. 7.23.

Test sequence: When tracking the testing sequence instead of the training one, the errors augment, specially in the case of using the pose mean prediction (i.e. assuming perfect reconstruction). The errors are smaller for the Balanced GPDM since it generalizes better than GPDM and GPLVM. This was also the behavior observed when studying the

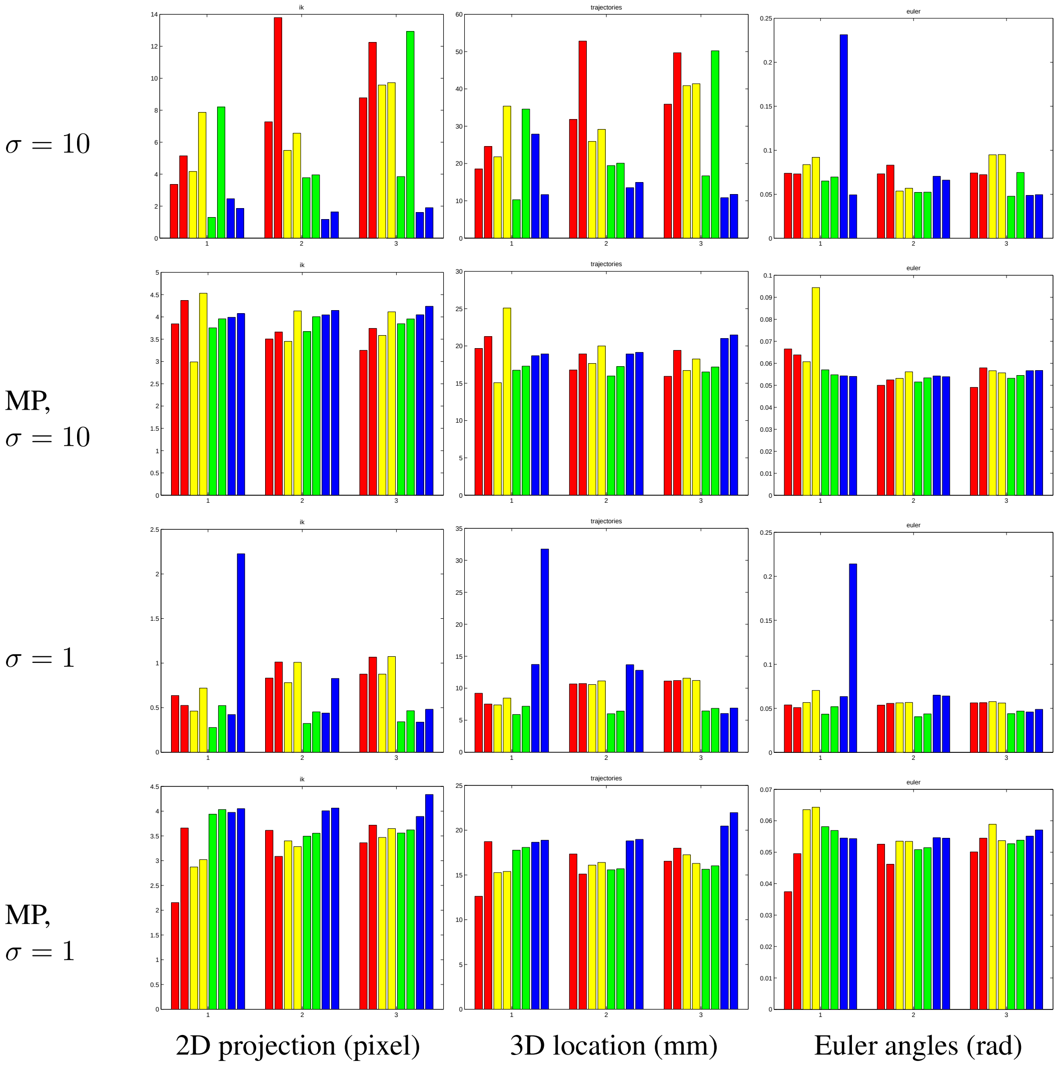
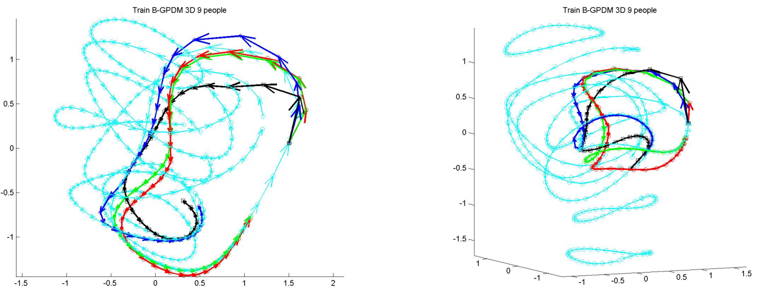
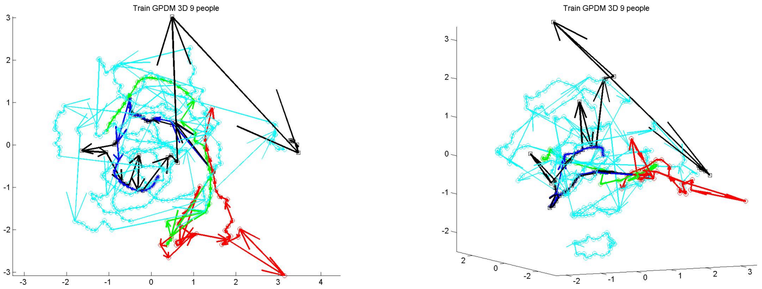


Figure 7.22: **Mean errors when tracking with GPs a test sequence** performed by a subjects whose motion is not part of the 6 and 9 people database, performed at 3km/h, while the training motions are performed at 4km/h. Each plot is composed of 3 groups, representing Balanced GPDM, GPDM and GPLVM results. For each GP type, 8 bars are shown in 4 different colors. Each color corresponds to a different model: 80D (9, 6, and 2 subjects) and 20D (6 subjects) respectively. For each model, two bars are depicted in the same color, the first one represents the error when using all the 2D projection as input, the second one uses only a subset, the same that was used for real sequences. Three types of errors are depicted: The 2D projections errors (pixels), the error in the 3D location of the joints (mm), and the error in the estimated Euler angles (radians).


Train B-GPDM 3D 9 people

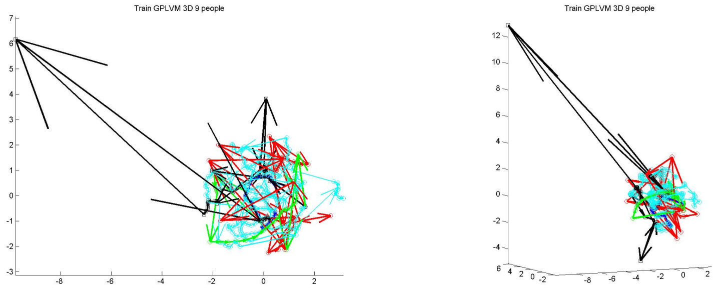
Train B-GPDM 3D 9 people

Balanced GPDM


Train GPDM 3D 9 people

Train GPDM 3D 9 people

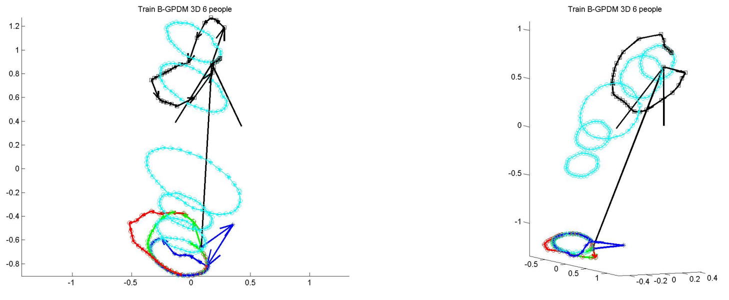
GPDM


Train GPLVM 3D 9 people

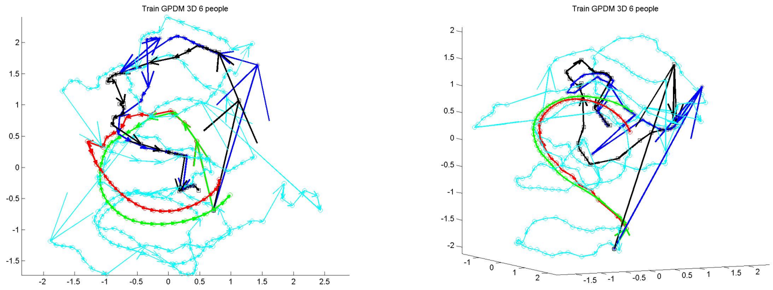
Train GPLVM 3D 9 people

GPLVM

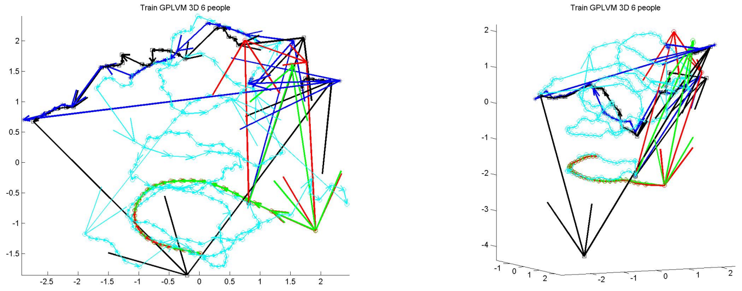
Figure 7.23: **Estimated latent trajectories for the 9 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the Balanced GPDM are the only ones that are smooth, resulting in large tracking errors for the GPDM and GPLVM when assuming perfect reconstruction (Fig. 7.22), since the pose is directly computed from the $\mathbf{x}$ estimated as the mean reconstruction prediction.



Balanced GPDM

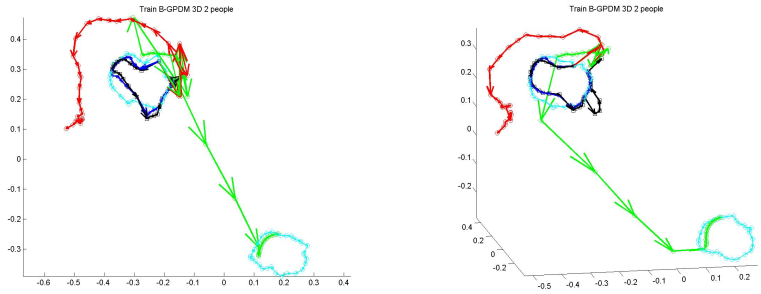


GPDM

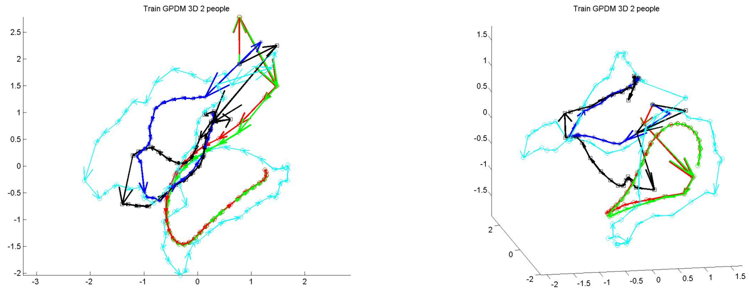


GPLVM

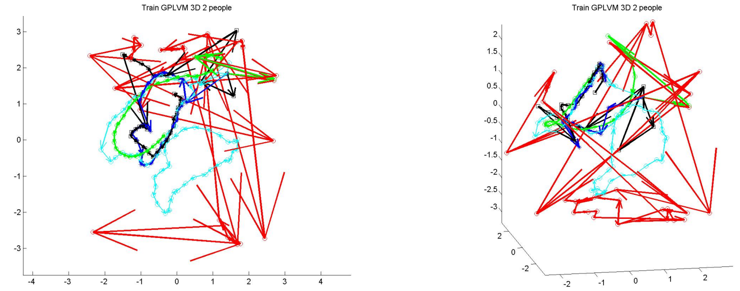
Figure 7.24: **Estimated latent trajectories for the 6 people database** when tracking the test sequence. The training data is shown in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the GPLVM latent trajectories are non smooth.



Balanced GPDM

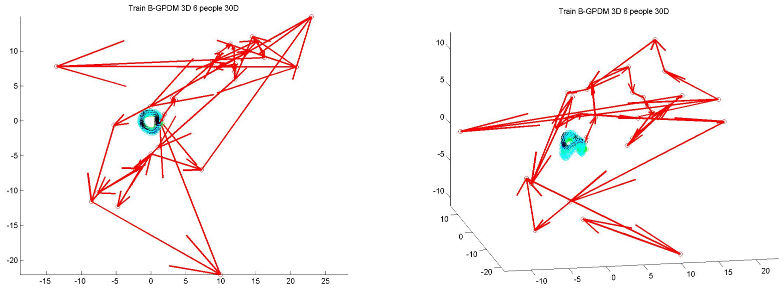


GPDM

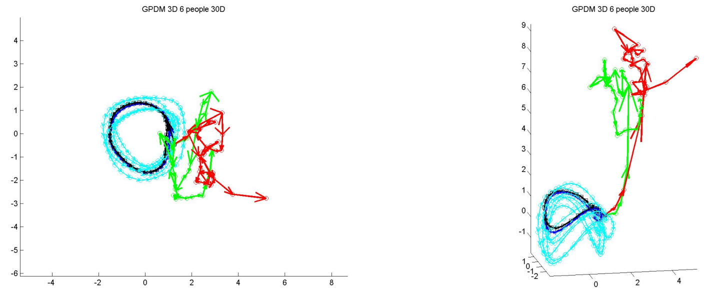


GPLVM

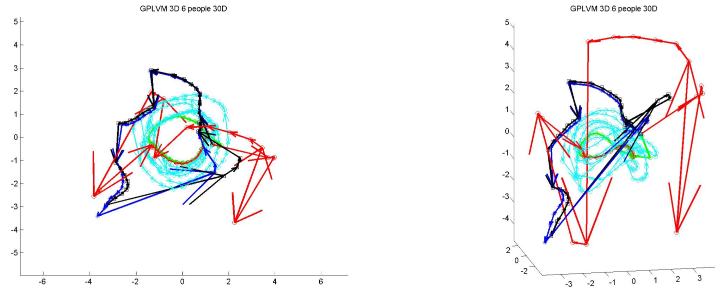
Figure 7.25: **Estimated latent trajectories for the 2 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the GPLVM are not smooth.



Balanced GPDM



GPDM



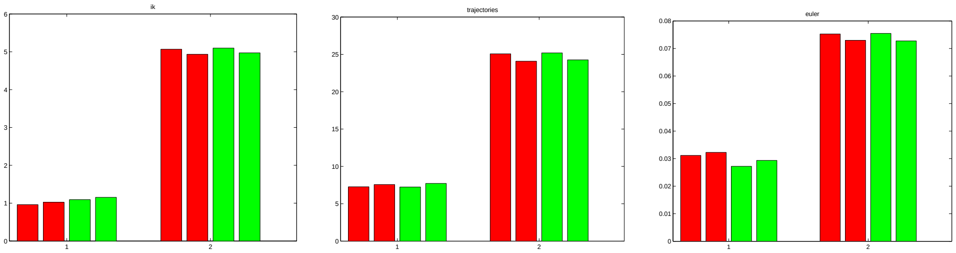
GPLVM

Figure 7.26: **Estimated latent trajectories for the 20D 6 people database** when tracking the test sequence. The training data is depicted in cyan. The latent trajectories tracked with Balanced GPDM (first row), GPDM (second row) and GPLVM (bottom row), with $\sigma_e = 10$. Note that the latent trajectories for the Balanced GPDM have diverged, but that the tracking errors (Fig. 7.22) are not big (in terms of 2D projection, and 3D position). The tracking is doing the correct thing, but the system does not have the freedom to evolve, resulting in large tracking errors.

generalization properties of the different models in section 7.1. As in the training sequence, when σ_e decreases, the tracking performed better, since the data is noiseless. Note the accuracy of the results, the mean errors are less than 1 cm, as depicted in Fig. 7.30. When using perfect reconstruction the 20D model performs the worst and the 9 subjects the best, since it has more training data to which the new sequence can be similar. The difference between the models are also smaller in that case.

Number of image cues: In general all the trackers performed better when using the complete set of joints that when using the subset used in the real sequences. But for most of the cases, the performance is similar, and such a subset provide enough information to obtain accurate results.

7.2.1.2 Motion models



2D projection 3D location Euler angles

Figure 7.27: **Mean errors when tracking with the PCA motion models.** Three types of errors (2D projection, 3D location, Euler angles) are depicted. Each plot is split in two groups, the left one represents the training sequence errors and the right one the test sequence ones. For each group 4 error bars of 2 different colors are depicted, each color represents a different window size ($\tau + 1 = 3$ on red and $\tau + 1 = 5$ on green). For each color two bars show the errors first for the complete set of joints and then for the subset of joints, with similar results. The errors for the sequence performed by a subject whose motion was not recorded when building the database are much bigger than the ones for the training sequence.

We test the linear motion models of chapter 4 using the two synthetic sequences. We initialize the normalized time μ_t to a linear function, 0 at the beginning and 1 at the end of the sequence. The style of the walking was initialized to the mean motion (i.e., $x_i = 0$), and estimated during the tracking.

Two sizes of the temporal window were used, 3 and 5 respectively, with very similar results as shown in Fig. 7.27. The three types of errors (2D projection, 3D location, Euler angles) are depicted. Each plot is split in two groups, the left one represents the training

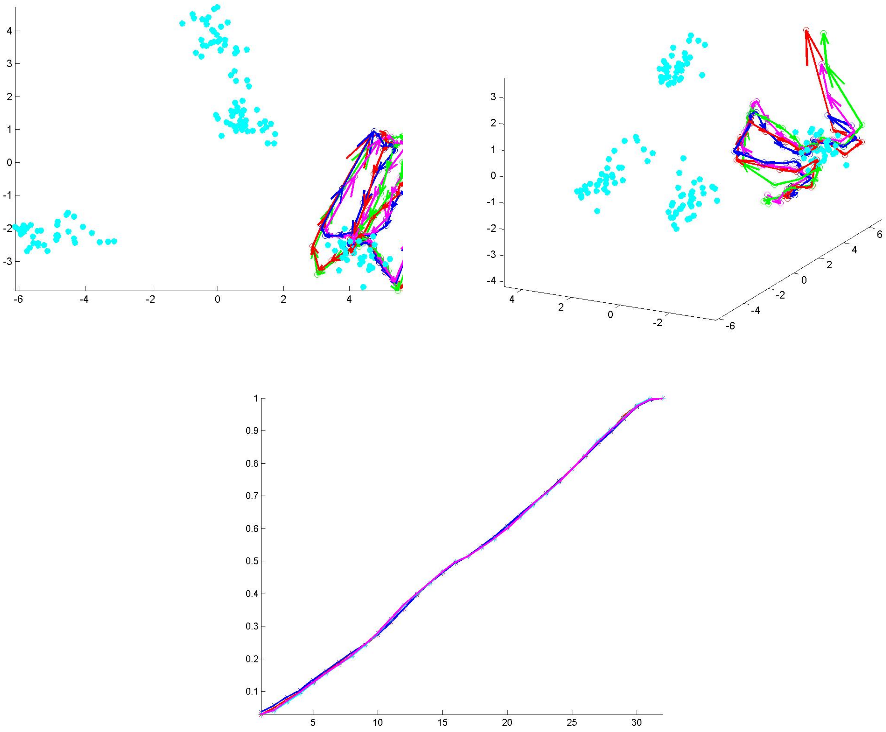


Figure 7.28: **PCA motion models** tracking results when tracking the training sequence. The tracking errors are depicted in the left side of each plot of Fig. 7.27. **Top row:** Latent coordinates recovered when tracking a training sequence. Note that the trajectories are less smooth than the ones obtained with the GP, and that they are in the neighborhood of the training data of the same subject, so that the tracker recognizes the subject. **Bottom row:** Estimated phase of the motion as a function of the frame index. Note that the phase is smoothly monotonically increasing.

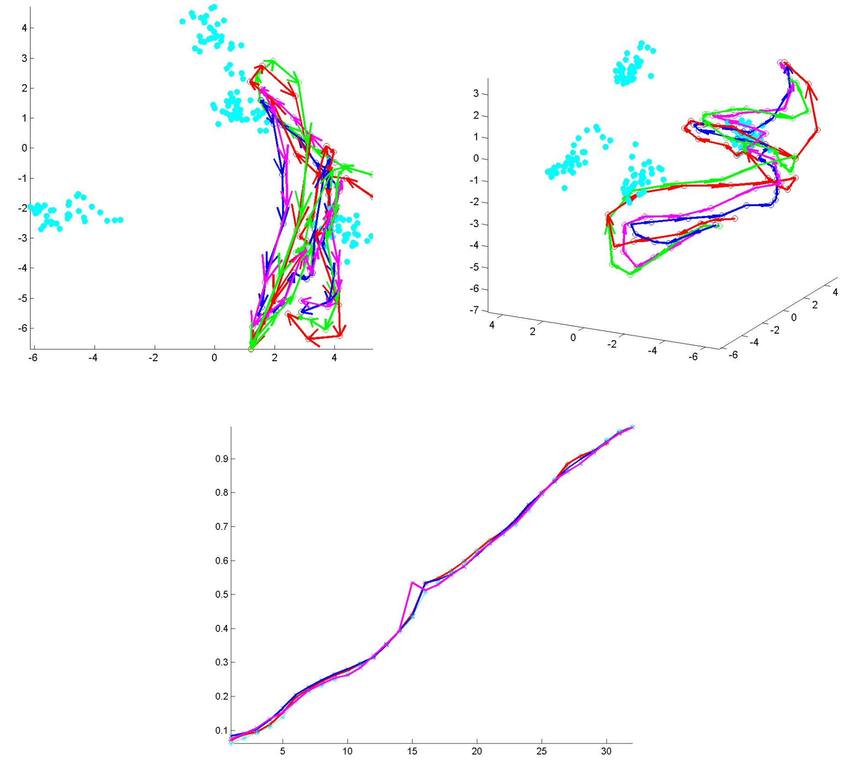
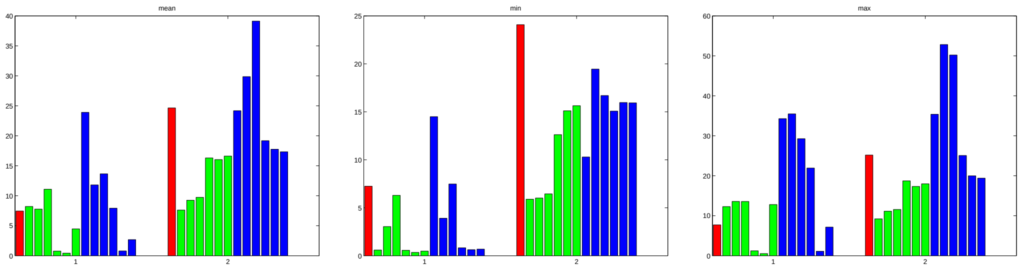


Figure 7.29: **PCA motion models** tracking results when tracking the test sequence that was performed at 3km/h by a subject whose motions were not used for learning. The tracking errors are depicted in the right side of each plot of Fig. 7.27. **Top row:** Latent coordinates recovered when tracking a training sequence. Note that the trajectories are less smooth than the ones obtained with the GP, and that they do not cluster in any of the subjects. The tracker does not recognize the subject. **Bottom row:** Estimated phase of the motion as a function of the frame index. Note that the phase is less smooth that for the training sequence depicted in Fig. 7.28.

	Training			Test		
	mean	min	max	mean	min	max
PCA	7.45	7.25	7.72	24.65	24.08	25.19
B-GPDM $\sigma_e = 1$	8.22	0.60	12.29	7.61	5.89	9.22
GPDM $\sigma_e = 1$	7.78	3.04	13.58	9.26	6.01	11.14
GPLVM $\sigma_e = 1$	11.09	6.29	13.57	9.74	6.45	11.56
B-GPDM MP $\sigma_e = 1$	0.77	0.57	1.23	16.31	12.62	18.72
GPDM MP $\sigma_e = 1$	0.43	0.35	0.54	16.03	15.10	17.34
GPLVM MP $\sigma_e = 1$	4.47	0.48	12.81	16.62	15.64	17.99
B-GPDM $\sigma_e = 10$	23.91	14.49	34.29	24.19	10.29	35.39
GPDM $\sigma_e = 10$	11.81	3.91	35.48	29.87	19.45	52.82
GPLVM $\sigma_e = 10$	13.65	7.48	29.28	39.13	16.69	50.21
B-GPDM MP $\sigma_e = 10$	7.92	0.83	21.93	19.19	15.08	25.08
GPDM MP $\sigma_e = 10$	0.80	0.63	1.12	17.76	15.96	19.99
GPLVM MP $\sigma_e = 10$	2.68	0.69	7.16	17.33	15.93	19.41



Training Testing
Mean

Training Testing
Min

Training Testing
Max

Figure 7.30: **Error comparison (mm).** Table representing the mean, minimal and maximal errors in the 3D joint estimation for the different 80D models tested in the training and test sequences. The second row represent the mean, minimal and maximal errors in the same order as the table. Each plot is divided in two, the training and testing results. The errors are expressed in mm. For the GPs models the errors are a mean, (minimal, maximal) of the 9, 6 and 2 people database with all the joints and with a subset. For the PCA, the results are obtained with $\tau = 3$ and $\tau = 5$, for all the joints and a subset. Note that the B-GPDM is the one with the best results when $\sigma_e = 1$. It generalizes much better to the test sequence.

sequence errors and the right one the test sequence ones. For each group 4 error bars of 2 different colors are depicted, each color represents a different window size ($\tau + 1 = 3$ on red and $\tau + 1 = 5$ on green). For each color two bars show the errors first for the complete set of joints and then for the subset of joints, with similar results. The errors for the sequence performed by a subject whose motion was not recorded when building the database are much bigger than the ones for the training sequence.

The errors obtained with the linear motion models are bigger than the ones obtained with the GP when using $\sigma_e = 1$, as depicted by Fig. 7.30. In general the results are also less smooth than the GP ones. The first rows of Figs. 7.28 and 7.29 depict the first three dimensions of the latent trajectories recovered for the training and test sequence respectively. The trajectories recovered from the training sequence (7.28) are in the neighborhood of the training data of the same subject. The tracker recognize the subject. The ones estimated from the test sequence are not in the neighborhood of a single subject. The subject is not recognized. The estimated phase of the motion for the training and test sequences are depicted in the last row of Figs. 7.28 and 7.29 respectively. Note that they are both monotonically increasing, but the test one is noisier, since it is sometimes difficult for the tracking to estimate when there is a phase or style change. There is not always a unique solution.

7.2.2 Caricatural walking

We choose a caricatural walking to test the generalization capabilities of the different methods, since it is very different from the training ones. We use the same initial poses $\mathbf{y}_{1:3}$ for all the methods. All the results were obtained using the same number of minimization steps.

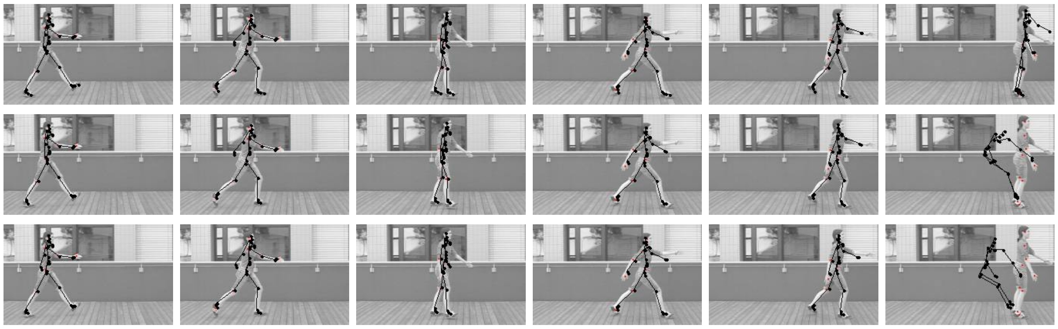
7.2.2.1 GP models

The first three rows of Fig. 7.31 depict the projected pose estimated when tracking using Balanced GPDM, GPDM and GPLVM respectively learned using the 20D, 6 people database. Note that the Balanced GPDM tracks perfectly the motion, but the GPDM and GPLVM diverge at the end of the sequence. The dynamical term is more important when dealing with motions very far from the training ones in order to not diverge.

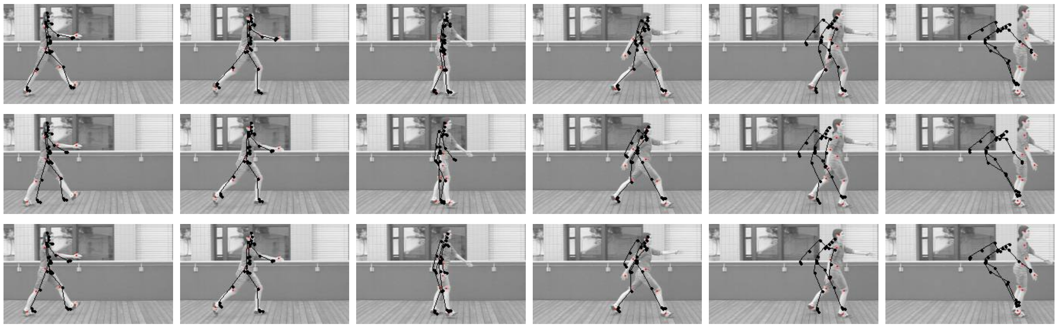
When using the 80D, 9 people database, the B-GPDM, GPDM and GPLVM diverge (three middle rows of Fig. 7.31), since the latent trajectories are not aligned by the phase of the motion, and 80D is too high dimensional to be estimated from a sequence that is so far from the training ones. The 3D volumetric visualization is depicted in Fig. 7.32.

7.2.2.2 Motion models

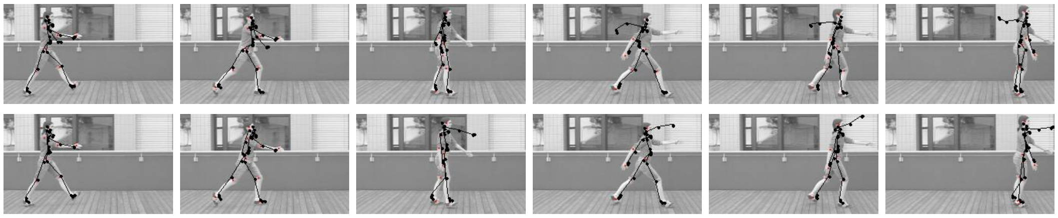
The PCA tracker produces impossible poses, when using 3 or 5 frames as temporal window. This is due to the fact that the latent positions are forced to be very far from the training



Balanced GPDM, GPDM and GPLVM learned from the 20D 6 subjects database.

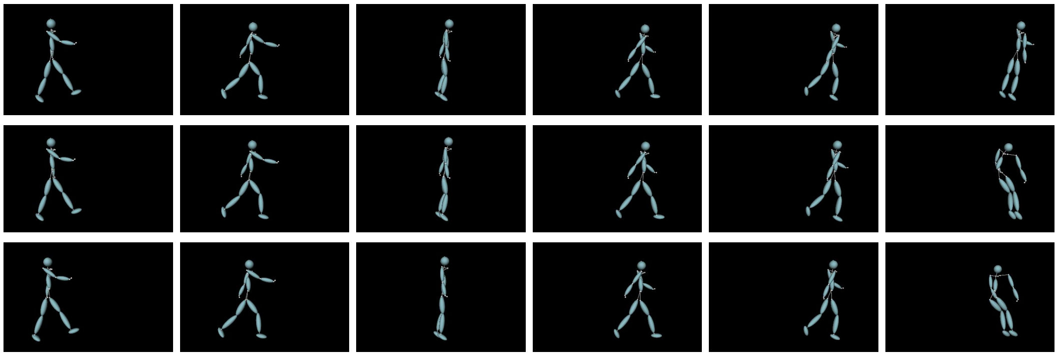


Balanced GPDM, GPDM and GPLVM learned from the 80D 9 subjects database.

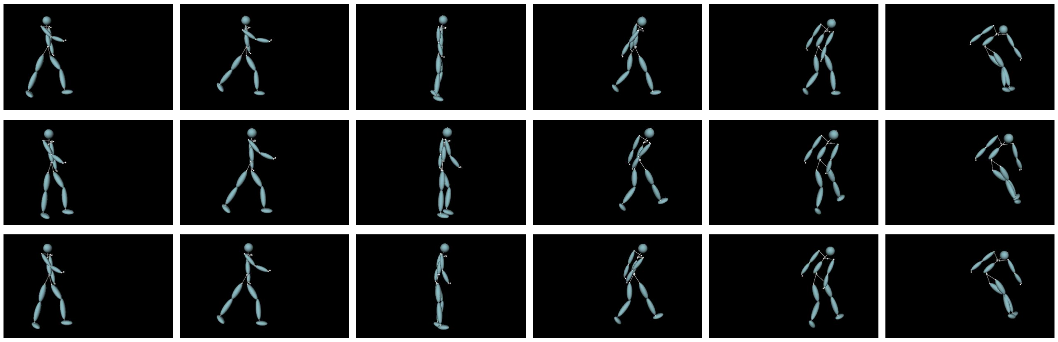


PCA with 3 and 5 frames window.

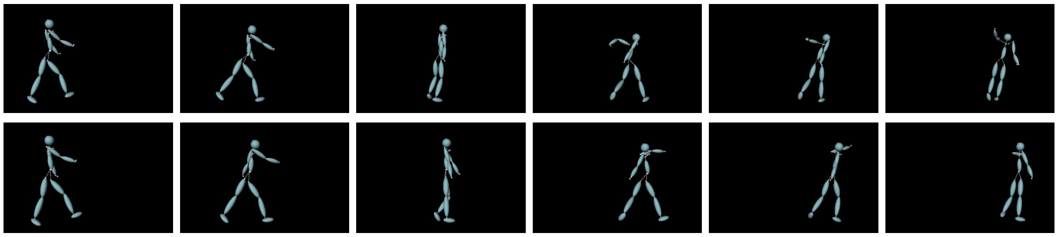
Figure 7.31: Tracking 40 frames of an exaggerated gait with the different GP and PCA models. Note that the 20D B-GPDM is the one that performed the best. The PCA results in impossible positions. The GPLVM and GPDM trackers are lost suddenly at the end of the sequence because of divergence. The 9 subject database too, since the latent trajectories are not aligned by phase of the motion, and 80D are too much to estimate for a sequence that is so far from the training ones. The 3D representation is depicted in Fig. 7.32.



Balanced GPDM, GPDM and GPLVM learned from the 20D 6 subjects database.



Balanced GPDM, GPDM and GPLVM learned from the 80D 9 subjects database.



PCA with 3 and 5 frames window.

Figure 7.32: Tracking 40 frames of an exaggerated gait with the different GP and PCA models. The 3D visualization is depicted by Fig. 7.32.

ones (6 standard deviations). Moreover the tracking results are not smooth. The PCA motion model does not generalize well to such motions.

7.3 Conclusion

We have discussed the influence of the different positional and angular parameterizations, and the amount of training data require to produce a good generalization when learning GPs. We have explored the consequences of varying the input space dimensionality from 80 and 20 dimensions, and of changing the dimensionality of the latent space.

We conclude that when learning high dimensional models, the positional parametrization is the only one that results in clustered and non-sparse models. The models learned from the 6 subject database are the best high dimensional ones, since they are clustered and poses that are produced in similar phases of the different motions are aligned. But these models are sparse. These models have a desirable property for classification, one dimension encodes the subject identity. We have showed that the 20D model result in the most clustered and non sparse model. Moreover the latent positions of the poses that are produced in similar phases of the different motions are aligned.

We have tested the accuracy of the different pose, motion and pose dynamical models in tracking synthetic and real data, and their ability to generalize to motions very far from the training ones. The Balanced GPDMs generalize better than GPDMs, SGPLVMs, and the PCA-based motion models, because they are smoother and provide better dynamical models that prevent divergence. This results in better estimations when tracking motions that are different from the training ones. This is specially true, when the differences become very large. The linear motion models result in impossible positions, and the SGPDM and GPDM diverge due to the non smoothness of their learned latent trajectories.

8 Conclusions and Future Work

In this work, we have proposed new ways to learn pose and motion prior models. We have demonstrated that they can be used to increase the performance of 3D body tracking algorithms, resulting in very realistic motions under very challenging conditions.

These models have proved very robust, allowing us to track with a single-hypothesis hill-climbing approach. This results in much lower computational complexity than the current multi-hypothesis techniques. We have demonstrated the effectiveness of our approaches for monocular tracking of cyclic motions such as walking and acyclic motions as golf swinging.

In practice we have traded the complexity of tracking for the complexity of knowing which model to apply. This might mean keeping several models active at any one time and selecting the one that fits best. This brings us back to multiple hypotheses tracking, but the multiple hypotheses are over models and not states. This might be much more effective than what many particle filters do because it ensures that the multiple hypotheses are sufficiently different to be worth exploring.

8.1 Contributions

We have shown that when representing whole motions, simple linear models can be used both to achieve good tracking performance, and to perform activity and subject recognition. We have introduced a new method for motion extrapolation, where these motion models are used to infer new motions of a subject that is observed once performing a given activity, while respecting his/her particular style. The major limitation of these linear motion models is that they required many examples to create a complete database with good generalization properties. Moreover, the training data need to be noiseless, segmented, and time warped. When learning multi-activity databases the convexity assumption of such models is violated, and better probabilistic models are needed in order to allow the model to produce only physically possible motions.

We have therefore investigated more complex non-linear statistical techniques. We have showed that Gaussian Process can be used to learn pose models from much smaller amounts of training data than competing techniques. Furthermore, they provide more realistic probabilistic models than the simplistic underlying Gaussian model of our motion models.

We have developed a SGPLVM-based method to learn prior models of 3D human pose and proved its effectiveness for monocular 3D tracking. In the case of both walking and golfing, we have been able to recover the motion from video sequences given a single exemplar of each motion to train the model. Furthermore, these priors sufficiently constrain the problem so that this could be accomplished with straightforward deterministic optimization. This is in sharp contrast to competing techniques that either involve large amounts of training data or computationally expensive multi-hypothesis tracking algorithms.

The tracking was accomplished with particularly simple second-order dynamics and an observation model based on a very small number of tracked features. The quality of our results with such simple dynamics and appearance models clearly demonstrates the power of the SGPLVM prior models.

In the presence of very noisy or missing data, for example due to occlusions, the simplistic second order Markov model is not realistic enough to sufficiently constrain the algorithm. Moreover, when learning models that contain stylistic diversity, from different people or from the same person performing an activity multiple times, the SGPLVM results in models whose latent trajectories are not smooth, and are therefore not suited for hill climbing tracking.

We have therefore developed a more sophisticated approach based on the more powerful GPDMs. We have introduced the Balanced GPDMs that learn smooth pose dynamical models with stylistic diversity. They can be used to reconstruct very realistic motions despite weak and noisy image measurements and significant occlusions. The quality of the results, given such a simple measurement model attest the utility of the GPDM priors.

The Balanced GPDMs have better generalization properties than GPDM, SGPLVM and PCA-based motion models, resulting in better estimations when tracking motions that are different than the training ones. This is especially true, when tracking motions very different, for example a caricatured walking. In such cases the motion models result in impossible positions, and the SGPDM and GPDM diverge due to the non-smoothness of their learned latent trajectories.

8.2 Future Work

Future work will focus on building better appearance models, as this is a principal weakness of our trackers. Furthermore, in all the models presented here, the global motion was assumed to be independent of the other joints and treated separately. Of course, this is not the case in reality. The coupling between the global translational velocity and the joint angle time series should be investigated.

Another area that requires further investigation is the combination of our tracker with inverse kinematics techniques to clean up the artifacts, such as foot sliding, which can be observed in some of our results. The simplest approach would be to use Inverse Kinematics as a post-processing step. A more ambitious approach would be to detect foot support phases in real-time and enforce them with an IK solver.

In this work we have restricted ourselves to model a small set of activities. Further work should involve incorporating multiple motion classes and transitions between them. Three different solutions exist, one can learn multiple models and use the one that works best, use a mixture of experts (Gaussian Process) where all the models are working together for a common goal, or learn different motion types within a common manifold, for example by using Gaussian Processes.

Although in this work we used Gaussian Process to modeling poses for tracking, they could be used to model motion as well, and solve some of the problems of the PCA-based motion models. They will provide a better probabilistic model forcing the motion to produce physically possible motions, even when learning different activities within the same model. The simplicity in learning motion models should let us to use linear kernels, resulting in much faster learning, and fewer local minima.

Finally, the main problem with GPs concerns the computation, which grows quickly with the number of training samples. We plan to investigate sparsification methods so that larger training sets can be used. Further investigation will focus in the use of annealing to reduce the overfitting of the GPs, when learning them from small number of examples.

Bibliography

[1] AGARWAL, A., AND TRIGGS, B. 3d human pose from silhouettes by relevance vector regression. In *Conference on Computer Vision and Pattern Recognition* (2004), vol. 2, pp. 882–888.

[2] AGARWAL, A., AND TRIGGS, B. Learning to Track 3D Human Motion from Silhouettes. In *International Conference in Machine Learning* (Banff, Alberta, Canada, July 2004).

[3] AGARWAL, A., AND TRIGGS, B. Tracking articulated motion with piecewise learned dynamical models. In *European Conference on Computer Vision* (Prague, May 2004), vol. 3, pp. 54–65.

[4] AGARWAL, A., AND TRIGGS, B. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 1 (January 2006), 44–58.

[5] ALEXA, M., AND MUELLER, W. Representing animations by principal components. In *Eurographics* (2000), vol. 19, pp. 411–418.

[6] ARIKAN, O., AND FORSYTH, D. Interactive motion generation from examples. *Computer Graphics, SIGGRAPH Proceedings 21*, 3 (2002).

[7] Ascension Electro-Magnetic Motion Catpure system. Ascension Technology CorporationV, Burlington, USA. http://www.ascension-tech.com/.

[8] BAERLOCHER, P., AND BOULIC, R. Parameterization and range of motion of the ball-and-socket joint. In *Avatars* (2000).

[9] BAERLOCHER, P., AND BOULIC, R. An Inverse Kinematics Architecture for Enforcing an Arbitrary Number of Strict Priority Levels. *The Visual Computer* (2004).

[10] BELKIN, M., AND NIYOGI, P. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2001, pp. 585–591.

[11] BISHOP, C. M. *Neural Networks for pattern recognition*. Oxford, 1995.

[12] BISHOP, C. M., AND TIPPING, M. E. A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 3 (1998), 281–293.

[13] BISSACCO, A. Modeling and Learning Contact Dynamics in Human Motion. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005), vol. 1, pp. 421–428.

[14] BLANZ, V., BASSO, C., POGGIO, T., AND VETTER, T. Reanimating Faces in Images and Video. In *Eurographics* (Granada, Spain, September 2003), vol. 22.

[15] BLANZ, V., AND VETTER, T. A Morphable Model for The Synthesis of 3–D Faces. In *Computer Graphics, SIGGRAPH Proceedings* (Los Angeles, CA, August 1999), pp. 187–194.

[16] BORGA, M., AND KNUTSSON, H. A Canonical Correlation Approach to Blind Source Separation. Technical Report LiU-IMT-EX-0062, Department of Biomedical Engineering, Link"ø"ping University, 2001.

[17] BOTTINO, A., AND LAURENTINNI, A. A silhouette based technique for the reconstruction of human movement. *Computer Vision and Image Understanding 83* (2001), 79–95.

[18] BRAND, M., AND HERTZMANN, A. Style Machines. *Computer Graphics, SIGGRAPH Proceedings* (July 2000), 183–192.

[19] BROSTOW, G. J., ESSA, I., STEEDLY, D., AND KWATRA, V. Novel Skeletal Representation For Articulated Creatures. In *European Conference on Computer Vision* (Prague, Czech Republic, May 2004), vol. 3, pp. 66–78.

[20] BRUDERLIN, A., AND WILLIAMS, L. Motion Signal Processing. *Computer Graphics, SIGGRAPH Proceedings* (Aug. 1995), 97–104.

[21] CALINON, S., AND BILLARD, A. Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM. In *International Conference in Machine Learning* (Bonn, Germany, 2005).

[22] CALINON, S., GUENTER, F., AND BILLARD, A. On learning the statistical representation of a task and generalizing it to various contexts. In *International Conference on Robotics and Automation* (2006).

[23] CARRANZA, J., THEOBALD, C., MAGNOR, M., AND SEIDEL, H. P. Free-viewpoint video of human actors. In *Computer Graphics, SIGGRAPH Proceedings* (2003), pp. 565–577.

[24] CHAI, J., AND HODGINS, J. K. Performance animation from low-dimensional control signals. In *Computer Graphics, SIGGRAPH Proceedings* (August 2005), vol. 24, pp. 686–696.

[25] CHALLIS, J. H. A procedure for determining rigid body transformation parameters. *Journal of Biomechanics 28*, 6 (1995), 733–737.

[26] CHEUNG, G., S.BAKER, AND KANADE, T. Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematis Estimation and Motion Capture. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, July 2003), pp. 569–577.

[27] CHOO, K., AND FLEET, D. People tracking using hybrid monte carlo filtering. In *International Conference on Computer Vision* (Vancouver, Canada, July 2001), vol. 2, pp. 321–328.

[28] CHU, C. W., JENKINS, O. C., AND MATARIC, M. J. Markerless kinematic model capture from volume sequences. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, July 2003), pp. 475–482.

[29] CMU database. http://mocap.cs.cmu.edu/.

[30] DAVISON, A. J., DEUTSCHER, J., AND REID, I. D. Markerless motion capture of complex full-body movement for character animation. In *Eurographics Workshop on Computer Animation and Simulation* (2001), Springer-Verlag LNCS, pp. 3–14.

[31] DE SILVA, V., AND TENENBAUM, J. Global versus local methods in nonlinear dimensionality reduction. In *Neural Information Processing Systems* (Cambridge, MA, 2003), vol. 15, MIT Press, pp. 721–728.

[32] DELAMARRE, Q., AND FAUGERAS, O. 3D Articulated Models and Multi-View Tracking with Silhouettes. In *International Conference on Computer Vision* (Corfu, Greece, September 1999), vol. 2, pp. 716–721.

[33] DEUTSCHER, J., BLAKE, A., AND REID, I. Articulated Body Motion Capture by Annealed Particle Filtering. In *Conference on Computer Vision and Pattern Recognition* (Hilton Head Island, SC, 2000), pp. 2126–2133.

[34] DEWAELE, G., DEVERNAY, F., AND HORAUD, R. Hand motion from 3d point trajectories and a smooth surface model. In *European Conference on Computer Vision* (May 2004), pp. 495–507.

[35] Digiclops Stereo Vision camera system. Point Grey Research Inc., Vancouver, Canada. http://www.ptgrey.com/products/digiclops/.

[36] DIMITRIJEVIĆ, M., ILIĆ, S., AND FUA, P. Accurate Face Models from Uncalibrated and Ill-Lit Video Sequences. In *Conference on Computer Vision and Pattern Recognition* (Washington, DC, June 2004).

[37] DONTCHEVA, M., YNGVE, G., AND POPOVIC, Z. Layered acting for character animation. *Computer Graphics, SIGGRAPH Proceedings* (2003).

[38] DORETTO, G., CHIUSO, A., WU, Y. N., AND SOATTO, S. Dynamic textures. *International Journal of Computer Vision 51*, 2 (2003), 91–109.

[39] DRUMMOND, T., AND CIPOLLA, R. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *International Conference on Computer Vision* (Vancouver, Canada, July 2001), vol. 2, pp. 315–320.

[40] D'SOUZA, A., VIJAYAKUMAR, S., AND SCHAA, S. Learning Inverse Kinematics . In *International Conference on Intelligent Robots and Systems* (Hawaii, 2001), pp. 298–303.

[41] ELGAMMAL, A., AND LEE, C. Inferring 3D Body Pose from Silhouettes using Activity Manifold Learning. In *CVPR* (Washington, DC, June 2004), vol. 2, pp. 681–688.

[42] ELGAMMAL, A., AND LEE, C. Separating Style and Content on a Nonlinear Manifold. In *CVPR* (Washington, DC, June 2004), vol. 1, pp. 478–485.

[43] ENGIN, A. E., AND TUMER, S. T. Three-dimensional kinematic modeling of the human shoulder complex-part i: Physical model and determination of joint sinus cones. *Journal of Biomechanical Engineering* (1989), 113–121.

[44] FORSYTH, D., AND PONCE, J. *Computer Vision. A modern approach.* Prenticd Hall, 2003.

[45] FRANCO, J. S., AND BOYER, E. Fusion of multi-view silhouette cues using a space occupancy grid. In *International Conference on Computer Vision* (Beijing, China, October 2005), pp. 1747–1753.

[46] GHAHRAMANI, Z., AND HINTON, G. The em algorithm for mixtures of factor analyzers. Tech. Rep. CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

[47] GIRARD, M. Interactive design of 3-d computer-animated legged animal motion. In *Proceedings of the 1986 workshop on Interactive 3D graphics* (1987), pp. 131–150.

[48] GLARDON, P., BOULIC, R., AND THALMANN, D. A coherent locomotion engine extrapolating beyond experimental data. In *Proceedings of CASA* (2004).

[49] GLARDON, P., BOULIC, R., AND THALMANN, D. Robust on-line adaptive footplant detection and enforcement for locomotion. *The Visual Computer 22*, 3 (2006), 194–209.

[50] GLEICHER, M. Motion editing with spacetime constraints. In *ACM Symposium on Interactive 3D Graphics* (April 1997), pp. 139–148.

[51] GLEICHER, M. Comparing constraint-based motion editing methods. *Graphical Models 63*, 2 (2001), 107–134.

[52] GOLAM, A., AND WONG, K. C. Dynamic time warp based framespace interpolation for motion editing. *Graphics Interface* (2000), 45–52.

[53] GRAUMAN, K., SHAKHNAROVICH, G., AND DARRELL, T. Inferring 3D structure with a statistical image-based shape model. In *International Conference on Computer Vision* (Nice, France, 2003), pp. 641–648.

[54] GROCHOW, K., MARTIN, S., HERTZMANN, A., AND POPOVIC, Z. Style-based inverse kinematics. In *Computer Graphics, SIGGRAPH Proceedings* (2004), pp. 522–531.

[55] HERDA, L., FUA, P., PLÄNKERS, R., BOULIC, R., AND THALMANN, D. Using skeleton-based tracking to increase the reliability of optical motion capture. *Human Movement Science Journal 20*, 3 (June 2001), 313–341.

[56] HERDA, L., URTASUN, R., AND FUA, P. Hierarchical Implicit Surface Joint Limits to Constrain Video-Based Motion Capture. In *European Conference on Computer Vision* (Prague, Czech Republic, May 2004), vol. 2, pp. 405–418.

[57] HERDA, L., URTASUN, R., AND FUA, P. Hierarchical Implicit Surface Joint Limits for Human Body Tracking. *Computer Vision and Image Understanding 99*, 2 (2005), 189–209.

[58] HERDA, L., URTASUN, R., HANSON, A., AND FUA, P. An automatic method for determining quaternion field boundaries for ball-and-socket joint limits. *International Journal of Robotics Research 22*, 6 (2003), 419–436.

[59] HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. Animating human athletics. *Computer Graphics, SIGGRAPH Proceedings* (1995), 71–78.

[60] HOWE, N. R., LEVENTON, M. E., AND FREEMAN, W. T. Bayesian reconstructions of 3D human motion from single-camera video. In *Neural Information Processing Systems* (Cambridge, MA, 1999), MIT Press.

[61] HU, M. K. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory 8* (1962), 179–187.

[62] HYVÁRINEN, A. Fast and robust fixed-point algorithm for independent component analysis. In *IEEE Transactions on Neural Networks* (1999), pp. 626–634.

[63] ISARD, M., AND BLAKE, A. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision 29*, 1 (August 1998), 5–28.

[64] ISARD, M., AND BLAKE, A. A mixed-state condensation tracker with automatic model-switching. In *International Conference on Computer Vision* (Mumbai, 1998).

[65] JACOBS, R. A., JORDON, M. I., NOWLAN, S. J., AND HINTON, G. E. Adaptive mixtures of local experts. *Neural Computation 3* (1991), 79–87.

[66] JENKINS, O. C., AND MATARIĆ', M. J. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *International Conference in Machine Learning* (Banff, Alberta, Canada, July 2004), vol. 69.

[67] JEPSON, A., FLEET, D. J., AND EL-MARAGHI, T. Robust on-line appearance models for vision tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 10 (2003), 1296–1311.

[68] JOLLIFFE, I. T. *Principal Component Analysis.* Springer series in statistics, Springer-Verlag, 1986.

[69] JULIER, S. J., AND UHLMANN, J. K. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of AeroSense. The 11th International Symposium on Aerospace/Defense Sensing, Simulatin and Control* (1997).

[70] KAMBHATLA, N., AND LEEN, T. K. Dimension Reduction by Local Principal Component Analysis. *Neural Computation 19* (1997).

[71] KARAN, B., AND VUKOBRATOVIĆ, M. Calibration and accuracy of manipulation robot models – an overview. *Mechanism and Machine Theory 29*, 3 (1992), 479–500.

[72] KIRK, A., O'BRIEN, J., AND FORSYTH, D. Skeletal Parameter Estimation from Optical Motion Capture Data. In *CVPR* (San Diego, CA, 2005).

[73] KOREIN, J. U. *A Geometric Investigation of Reach.* MIT Press, Cambridge, MA, 1985.

[74] KOVAR, L., AND GLEICHER, M. Flexible automatic motion blending with registration curves. In *ACM Symposium on Computer Animation* (July 2003), pp. 214–224.

[75] KOVAR, L., GLEICHER, M., AND PIGHIN, F. Motion Graphs. In *Computer Graphics, SIGGRAPH Proceedings* (July 2002), pp. 473 – 482.

[76] LAWRENCE, N. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research 6* (2005), 1783–1816.

[77] LAWRENCE, N. The Gaussian process latent variable model. Technical report, University of Sheffield, 2006.

[78] LAWRENCE, N. D. Gaussian Process Models for Visualisation of High Dimensional Data. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

[79] LAWRENCE, N. D., SEEGER, M., AND HERBRICH, R. Fast sparse Gaussian process methods: The informative vector machine. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2003, pp. 609–616.

[80] LEE, J., CHAI, J., REITSMA, P., HODGINS, J., AND POLLARD, N. Interactive control of avatars animated with human motion data. In *Computer Graphics, SIGGRAPH Proceedings* (2002), pp. 491–500.

[81] LEE, J., AND SHIN, S. Y. A hierarchical approach to interactive motion editing for human-like figures. In *Computer Graphics, SIGGRAPH Proceedings* (aug 1999), pp. 39–48.

[82] LEPETIT, V., SHAHROKNI, A., AND FUA, P. Robust Data Association For Online Applications. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, June 2003), vol. 1, pp. 281–288.

[83] LI, R., YANG, M. H., SCLAROFF, S., AND TIAN, T. Monocular Tracking of 3D Human Motion with a Coordinated Mixture of Factor Analyzers . In *European Conference on Computer Vision* (Graz, Austria, May 2006).

[84] LIM, I. S., AND THALMANN, D. Construction of animation models out of captured data. In *Proc. IEEE Conf. Multimedia and Expo* (2002).

[85] LIU, C., HERTZMANN, A., AND POPOVIC, Z. Learning Physics-Based Motion Style with Nonlinear Inverse Optimization. In *Computer Graphics, SIGGRAPH Proceedings* (2005), pp. 1071–1081.

[86] LIU, C., AND POPOVIC, Z. Synthesis of Complex Dynamic Character Motion from Simple Animations. In *Computer Graphics, SIGGRAPH Proceedings* (2002).

[87] LIU, J. S., AND CHEN, R. Sequential monte carlo methods for dynamic systems. *Journal American Statistical Association 93* (1998), 1032–1044.

[88] MACCORMICK, J., AND ISARD, M. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *European Conference on Computer Vision* (2000), vol. 2, pp. 3–19.

[89] MACKAY, D. J. C. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, Section A 354*, 1 (1995), 73–80.

[90] MACKAY, D. J. C. Introduction to Gaussian processes. In *Neural Networks and Machine Learning*, C. M. Bishop, Ed., NATO ASI Series. Kluwer Academic Press, 1998, pp. 133–166.

[91] MACKAY, D. J. C. *Information Theory, Inference, and Learning Algorithms.* Cambridge University PressU, 2003.

[92] MAUREL, W., AND D.THALMANN. Human Shoulder Modeling Including Scapulo-Thoracic Constraint and Joint Sinus Cones. *Computers and Graphics 24* (2001), 203–218.

[93] MENACHE, A. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, 1999.

[94] Meta Motion electro-mechanical motion capture. Meta Motion, San Francisco, CA, USA. http://www.metamotion.com/.

[95] MIKIC, I., TRIVEDI, M., HUNTER, E., AND COSMAN, P. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision 53*, 3 (2003), 199–223.

[96] MOESLUND, T. *Computer Vision-Based Motion Capture of Body Language*. PhD thesis, Aalborg University, Aalborg, Denmark, June 2003.

[97] MOESLUND, T., AND GRANUM, E. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding 81*, 3 (March 2001), 231–268.

[98] MORI, G., AND MALIK, J. Estimating Human Body Configurations Using Shape Context Matching. In *European Conference on Computer Vision* (2002), vol. 3, pp. 666–680.

[99] MORI, G., REN, X., EFROS, A., AND MALIK, J. Recovering Human Body Configurations: Combining Segmentation and Recognition. In *Conference on Computer Vision and Pattern Recognition* (Washington, DC, 2004), vol. 2, pp. 326–333.

[100] Motion Analysis optical motion capture system. Motion Analysis Corporation, Santa Rosa, CA, USA. http://http://www.motionanalysis.com/www.ascension-tech.com/.

[101] MULTON, F., FRANCE, L., CANI-GASCUEL, M., AND DEBUNNE, G. Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation 10*, 1 (1999), 39–54.

[102] MURRAY, M. P. Gait as a total pattern of movement. *American Journal of Physical Medicine 46*, 1 (1967), 290–333.

[103] NEAL, R. M. Bayesian learning for neural networks. vol. 118 of *Lecture Notes in Statistics*, Springer-Verlag.

[104] NORTH, B., BLAKE, A., ISARD, A., AND RITTSCHER, J. Learning and classification of complex dynamics . *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 9 (2000), 1016–1034.

[105] O'Brien, J. F., Bodenheimer, R. E., Brostow, G. J., and Hodgins, J. K. Automatic joint parameter estimation from magnetic motion capture data. *Graphics Interface* (May 2000), 53–60.

[106] Ormoneit, D., Sidenbladh, H., Black, M., and Hastie, T. Learning and tracking cyclic human motion. In *Advances in Neural Information Processing Systems 13* (2001), pp. 894–900.

[107] Park, S., Shin, H. J., and Shin, S. On-line locomotion generation based on motion blending. In *ACM Symposium on Computer Animation* (2002).

[108] Pavolic, J. M., Rehg, J., and MacCormick, J. Learning switching linear models of human motion. In *Neural Information Processing Systems* (2000), pp. 981–987.

[109] Perlin, K. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1* (1995), 5–15.

[110] Physilog, a portable datalogger for long term recording. LMAM, EPFL, Switzerland. http://lmam.epfl.ch/page2842.html.

[111] Plänkers, R., and Fua, P. Articulated Soft Objects for Multi-View Shape and Motion Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 9 (2003), 1182–1187.

[112] Polhemus electro-magnetical motion capture system. Polhemus, Colchester, USA. http://www.polhemus.com/.

[113] Popovic, Z., and Witkin, A. Physically Based Motion Transformation. *Computer Graphics, SIGGRAPH Proceedings* (1999), 11–20.

[114] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. *Numerical Recipes, the Art of Scientific Computing.* Cambridge U. Press, Cambridge, MA, 1992.

[115] Quinonero-Candela, J., and Rasmussen, C. E. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research 6* (December 2005), 1939–1959.

[116] Rahimi, A., Recht, B., and Darrell, T. Learning appearance manifolds from video. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005), pp. 868–875.

[117] Ringer, M., and Lasenby, J. A procedure for automatically estimating model parameters in optical motion capture. In *British Machine Vision Conference* (Cardiff, UK, 2002), pp. 747–756.

[118] ROSALES, R., AND SCLAROFF, S. Infering Body Pose without Tracking Body Parts. In *Conference on Computer Vision and Pattern Recognition* (June 2000), vol. 2, pp. 506–511.

[119] ROSE, C., COHEN, M., AND BODENHEIMER, B. Verbs and adverbs: Multidimensional motion interpolation. *Computer Graphics and Applications 18*, 5 (1998), 32–41.

[120] ROWEIS, S. Em algorithms for pca and spca. In *Neural Information Processing Systems* (Cambridge, MA, 1997), MIT Press.

[121] ROWEIS, S., AND SAUL, L. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[122] ROWEIS, S., SAUL, R., AND HINTON, G. E. Global coordination of local linear models. In *Neural Information Processing Systems* (Cambridge, MA, 2001), MIT Press, pp. 889–896.

[123] SHAKHNAROVICH, G., VIOLA, P., AND DARRELL, T. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision* (Nice, France, 2003).

[124] SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94.

[125] SHOEMAKE, K. Animating Rotation with Quaternion Curves. *Computer Graphics, SIGGRAPH Proceedings 19* (1985), 245–254.

[126] SIDENBLADH, H., BLACK, M. J., AND FLEET, D. J. Stochastic Tracking of 3D human Figures using 2D Image Motion. In *European Conference on Computer Vision* (June 2000), vol. 2, pp. 702–718.

[127] SIDENBLADH, H., BLACK, M. J., AND SIGAL, L. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conference on Computer Vision* (Copenhagen, Denmark, May 2002), vol. 1, pp. 784–800.

[128] SILAGHI, M.-C., PLÄNKERS, R., BOULIC, R., FUA, P., AND THALMANN, D. Local and global skeleton fitting techniques for optical motion capture. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments* (Geneva, Switzerland, November 1998).

[129] SLOAN, P.-P. J., ROSE, C. F., AND COHEN, M. F. Shape by example. In *Symposium on Interactive 3D Graphics* (2001), pp. 135–144.

[130] SMINCHISESCU, C., AND JEPSON, A. Generative Modeling for Continuous Non-Linearly Embedded Visual Inference. In *International Conference in Machine Learning* (Banff, Alberta, Canada, July 2004), vol. 69, pp. 96–103.

[131] SMINCHISESCU, C., KANAUJIA, A., LI, Z., AND METAXAS, D. Discriminative Density Propagation for 3D Human Motion Estimation. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005), vol. 1, pp. 390–397.

[132] SMINCHISESCU, C., AND TRIGGS, B. Covariance Scaled Sampling for Monocular 3D Body Tracking. In *Conference on Computer Vision and Pattern Recognition* (Hawaii, 2001), vol. 1, pp. 447–454.

[133] SMINCHISESCU, C., AND TRIGGS, B. Kinematic Jump Processes for Monocular 3D Human Tracking. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, June 2003), vol. I, pp. 69–76.

[134] SNELSON, E., RASMUSSEN, C. E., AND GHAHRAMANI, Z. Warped Gaussian Processes. In *Neural Information Processing Systems* (2004), pp. 337–344.

[135] STARCK, J., AND HILTON, A. Model-Based Multiple View Reconstruction of People. In *International Conference on Computer Vision* (Nice, France, 2003), pp. 915–922.

[136] STARCK, J., AND HILTON, A. Spherical Matching for Temporal Correspondence of Non-Rigid Surfaces. In *International Conference on Computer Vision* (Beijing, China, 2005), pp. 1387–1394.

[137] SULLIVAN, J., AND CARLSSON, S. Recognizing and tracking human action. In *European Conference on Computer Vision* (2002), vol. 1, pp. 629–644.

[138] SUNG, H. *Gaussian mixture regression and classification*. PhD thesis, Rice University, 2004.

[139] SWITZER, P., AND GREEN, A. A. Min/max autocorrelation factors for multivariate spatial imagery. Technical Report 6, Department of Statistics, Standford University, 1984.

[140] TAYLOR, C. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding 80*, 3 (December 2000), 349–363.

[141] TENENBAUM, J., DE SILVA, V., AND LANGFORD, J. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[142] TIAN, T., LI, R., AND SCLAROFF, S. Articulated Pose Estimation in a Learned Smooth Space of Feasible Solutions. In *CVPR Learning Workshop* (San Diego, CA, 2005), vol. 3.

[143] TIPPING, M. The relevance vector machine. In *Neural Information Processing Systems* (Cambridge, MA, 2000), MIT Press.

[144] TIPPING, M. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research 1* (2001), 211–244.

[145] TIPPING, M., AND BISHOP, C. Probabilistic principal component anlaysis. *Journal of the Royal Statistical Society, B 61*, 3 (1999), 611–622.

[146] TOLANI, D., BADLER, N., AND TALLIER, J. A kinematic model of the human arm using triangular bézier spline surfaces. *Graphical Models* (2000).

[147] TROJE, N. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision 2*, 5 (2002), 371–387.

[148] UNUMA, M., ANJYO, K., AND TAKEUCHI, R. Fourier Principles for Emotion-based Human Figure. *Computer Graphics, SIGGRAPH Proceedings* (Aug. 1995), 91–96.

[149] URTASUN, R., FLEET, D. J., AND FUA, P. Monocular 3–d tracking of the golf swing. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005), vol. 2, pp. 932–938.

[150] URTASUN, R., FLEET, D. J., AND FUA, P. 3d people tracking with gaussian process dynamical models. In *Conference on Computer Vision and Pattern Recognition* (New York, June 2006).

[151] URTASUN, R., FLEET, D. J., HERTZMAN, A., AND FUA, P. Priors for people tracking from small training sets. In *International Conference on Computer Vision* (Beijing, China, October 2005), pp. 403–410.

[152] URTASUN, R., AND FUA, P. 3D Human Body Tracking using Deterministic Temporal Motion Models. In *European Conference on Computer Vision* (Prague, Czech Republic, May 2004), vol. 3, pp. 92–106.

[153] URTASUN, R., AND FUA, P. Human Motion Models for Characterization and Recognition. In *Automated Face and Gesture Recognition* (Seoul, Korea, May 2004), pp. 17–22.

[154] URTASUN, R., GLARDON, P., BOULIC, R., THALMANN, D., AND FUA, P. Style-based motion synthesis. *Computer Graphics Forum 23*, 4 (December 2004), 799–812.

[155] Vicon Optical Motion Catpure system. Vicon Motion System Ltd., Oxford, UK. http://www.vicon.com/.

[156] VIJAYAKUMA, S., D'SOUZA, A., AND SCHAAL, S. Incremental Online Learning in High Dimensions. *Neural Computation 17*, 12 (2005), 2602–2634.

[157] VIJAYAKUMAR, S., AND SCHAAL, S. LWPR : An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space. In *International Conference in Machine Learning* (Standford, CA, 2000).

[158] WANG, J. Gaussian process dynamical models for human motion. Master's thesis, University of Toronto, 2005.

[159] WANG, J., FLEET, D. J., AND HERTZMAN, A. Gaussian Process dynamical models. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.

[160] WANG, Q., XU, G., AND AI, H. Learning object intrinsic structure for robust visual tracking. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, June 2003).

[161] WATT, A., AND WATT, M. Advanced animation and rendering techniques, 1992.

[162] WEISS, Y. Motion segmentation using em – a short tutorial, 1997.

[163] WOOTEN, W. L., AND HODGINS, J. K. Simulating leaping, tumbling, landing and balancing humans. In *Proceedings of IEEE International Conference on Robotics and Automation* (2000).

[164] YACOOB, Y., AND BLACK, M. J. Parametric Modeling and Recognition of Activities. In *International Conference on Computer Vision* (Mumbai, India, 1998), pp. 120–127.

[165] YANG, M. H., AHUJA, N., AND KRIEGMAN, D. Face detection using a mixture of factor analyzers. In *International Conference on Image Processing* (1999).

Raquel URTASUN
Av. de Tivoli, 2C
1007 Lausanne
Switzerland

URL: http://cvlabwww.epfl.ch/∼rurtasun
e-mail: raquel.urtasun@epfl.ch
Date of Birth : January 30, 1976
Nationality : Spanish

Education

- *Ph.D. Candidate* (since 2001),
  Computer Vision Laboratory, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Doctoral School in Computer Science* (2000-2001),
  School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Master Thesis* (1999-2000),
  Corporate Communications Department, Institut EURECOM, Sophia-Antipolis, France.
- *B.S. and M.S. in Telecommunication Engineering* (1994-1999),
  Universidad Publica de Navarra (UPNA), Pamplona, Spain.

Professional Experience

- *Research and Teaching Assistant* (since 2002),
  Computer Vision Laboratory, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Research Assistant* (Summer 2000),
  TSI department, École Nationale Supérieure des Télécommunications, Paris, France.
- *Research Assistant* (1999),
  Corporate Communications Department, Institut EURECOM, Sophia-Antipolis, France.
- *Summer Internship* (Summer 1999),
  Electrical Engineering Department, University of Navarra, Pamplona, Spain.

Awards and Distinctions

- *PostGraduate Fellowship* (2000),
  Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Graduate Fellowship* (1999),
  Institut EURECOM, Sophia-Antipolis, France.
- *Reseach Fellowship* (1998),
  Spanish Ministry of Education and Culture Fellowship.

Publications

Journals

1. L. Herda, **R. Urtasun** and P. Fua, *Hierarchical Implicit Surface Joint Limits for Human Body Tracking*, In Computer Vision and Image Understanding (CVIU), 2005.
2. **R. Urtasun**, P. Glardon, R. Boulic, D. Thalmann and P. Fua, *Style-based Motion Synthesis*, In Computer Graphics Forum (CGF), Vol. 23, number 4 pp. 799-812. December 2004.
3. L.Herda, **R.Urtasun**, P. Fua, A.Hanson, *Automatic Determination of Shoulder Joint Limits using Quaternion Field Boundaries*, International Journal of Robotics Research (IJRR), 22(6): 419 - 436, 2003.
4. P. Dokladal, I. Bloch, M. Couprie, D. Ruijters, **R. Urtasun** and L. Garnero, *Topologically Controlled Segmentation of 3D Magnetic Resonance Images of the Head by using Morphological Operators*, Pattern Recognition, 36(10):2463 - 2478, 2003.

Conferences

1. **R. Urtasun**, D. J. Fleet and P. Fua, *Gaussian Process Dynamical Models for 3D people tracking*, In Conference on Computer Vision and Pattern Recognition (CVPR) New York, June 2006.
2. **R. Urtasun**, D. J. Fleet, A. Hertzmann and P. Fua, *Priors for People Tracking from Small Training Sets*, In International Conference in Computer Vision (ICCV) Beijing, China, October 2005.

3. **R. Urtasun**, D. J. Fleet and P. Fua, *Monocular 3D Tracking of the Golf Swing*, In Conference on Computer Vision and Pattern Recognition (CVPR) San Diego, CA, June 2005.
4. **R. Urtasun**, P. Fua, *3D Human Body Tracking using Deterministic Motion Models*, In European Conference on Computer Vision (ECCV), Prague, Czech Republic, May 2004.
5. L.Herda, **R. Urtasun**, P. Fua, *Hierarchical Implicit Surface Joint Limits to Constrain Video-Based Motion Capture*, In European Conference on Computer Vision (ECCV), Prague, Czech Republic, May 2004.
6. **R. Urtasun**, P. Fua, *3D Tracking for Gait Characterization and Recognition*, In Proceeding of the 6th International Conference on Automatic Face and Gesture Recognition (FGR), Seoul, Korea, May 2004. IEEE Computer Society.
7. L. Herda, **R. Urtasun**, P. Fua and A. Hanson, *An Automatic Method for Determining Quaternion Field Boundaries for Ball-and-Socket Joint Limits*, In Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition (FGR), pages 95 - 100, Washington DC, May 2002. IEEE Computer Society.
8. P. Dokladal, **R. Urtasun**, I. Bloch and L. Garnero, *Segmentation of 3D head MR images using Morphological reconstruction under constraints and automatic selection of markers*, In International Conference on Image Processing (ICIP), pages 1075-1078, Thessaloniki, Greece, October 2001.