

Continuous-state Graphical Models for Object Localization, Pose Estimation and Tracking

by

Leonid Sigal

B. A., Boston University, 1999

M. A., Boston University, 1999

Sc. M., Brown University, 2003

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in the
Department of Computer Science at Brown University

Providence, Rhode Island

May 2008

© Copyright 2003, 2004, 2006, 2008 by **Leonid Sigal**

This dissertation by **Leonid Sigal** is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____

Michael J. Black, Director

Recommended to the Graduate Council

Date _____

William T. Freeman, Reader
(Electrical Engineering and Computer Science)
Massachusetts Institute of Technology

Date _____

John F. Hughes, Reader
(Department of Computer Science)

Date _____

David Mumford, Reader
(Department of Applied Mathematics)

Approved by the Graduate Council

Date _____

Sheila Bonde
Dean of the Graduate School

VITA

Leonid Sigal was born on May 23, 1977 in Kiev, Ukraine. He is a Ph.D. candidate under the supervision of Michael J. Black at Brown University; he received his B.Sc. degrees in Computer Science and Mathematics from Boston University (1999), his M.A. from Boston University (1999), and his M.S. from Brown University (2003). From 1999 to 2001, he worked as a senior vision engineer at Cognex Corporation, where he developed industrial vision applications for pattern analysis and verification. In 2002, he spent a semester as a research intern at Siemens Corporate Research (SCR) working on autonomous obstacle detection and avoidance for vehicle navigation. During the summers of 2005 and 2006, he worked as a research intern at Intel Applications Research Lab (ARL) on human pose estimation and tracking. His work received the Best Paper Award at the Articulate Motion and Deformable Objects Conference in 2006 (with Michael J. Black). Leonid's research interests are primarily in computer vision and machine learning, including human motion analysis, graphical models, probabilistic and hierarchical inference.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor and mentor Michael J. Black for his attentive supervision during my six years at Brown University. Michael provided me with a wealth of expertise and knowledge. He taught me how to choose and approach challenging problems and how to formulate a clear scientific argument. His enthusiasm and ability to engage students in interesting problems has served as an inspiration to me. As an advisor, Michael struck a perfect balance of providing help to me where and when it was needed, yet allowing me the freedom to grow as an independent researcher.

I am also grateful to all members of Browns' computer vision group and graduate community for helpful discussions, collaborations and a friendly environment that allowed me to conduct this research. I would specifically like to thank Stefan Roth for being a congenial colleague and for the advice that he has given me over the years; Alexandru Balan for a variety of collaborations and for introducing me to the SCAPE model. I would also like to thank Alexandru for his entertaining personal stories and the companionship during numerous conference travels and internships. I would also like to mention other group members who have indirectly affected this thesis: Gregory Shakhnarovich, Payman Yadollahpur and Frank Wood. In addition, I want to thank Chad Jenkins and his students for introducing me to robotics and physics-based simulation, while this has no direct relation to this thesis, it has affected my overall research agenda.

The ideas developed in this thesis have also benefited from numerous external collaborations. I would like to thank Michael Isard for his early input and discussions on Particle Massage Passing (PAMPAS), which is at the core of this thesis. I would also like to thank Alex Ihler and Eric Sudderth for interesting discussions on the relations between Non-parametric Belief Propagation (NBP), developed by them, and PAMPAS. In addition, I would like to thank Dorin Comaniciu and Ying Zhu from Siemens Corporate Research (SCR) for the mentorship they provided me during my five month internship in Princeton, NJ. As part of my internship, I was able to extend my prior work on articulated pose estimation and tracking to the generic obstacle detection domain (for autonomous vehicle navigation); this gave rise to the results presented in Chapter 4. I would also like to thank, Horst Haussecker, who has hosted me as a Research Intern at Intels Applications Research Laboratory (ARL) in Santa Clara, CA for two summers in 2006 and 2007. Horst provided me with an invaluable environment and freedom to pursue research of my own personal interest. The collaborations with various people in the ARL group, in particular Nizhny Novgorod's team has benefited this thesis in many practical aspects. Much of the experiments presented in Chapter 5 would not be possible (or as easily attained) without their help. Hence, I would like to thank all members of the Intel Research group with whom I had a chance to interact and collaborate with during my visits: Adam Seeger, Oscar Nestares, Jean-Yves Bouguet, Konstantin Rodyushkin (Nizhny Novgorod, Russia) and Alexander Kuranov (Nizhny Novgorod, Russia).

Moreover, I am grateful to members of my thesis committee, William Freeman, John Hughes (Spike)

and David Mumford, for taking the time from their busy schedules to read and comment on my work. In addition, I would like to thank David Mumford whose class on statistical modeling of shape has introduced me to Belief Propagation and much of the basic statistical and mathematical formalism that I use throughout this thesis. Both his class and our outside discussions were always insightful. I would also like to thank Spike for his valuable input on modeling statistical distributions over angles.

Before coming to Brown University, my interest in the pursuit of an academic career and computer vision as a field was shaped by a number of key people. Most notably, Stan Sclaroff, my former advisor at Boston University. I had a chance to work with Stan both as an undergraduate and a master student. I am thankful to Stan, for sparking my interest in computer vision and for his continuing support and career advice over the years. I also would like to acknowledge people in IVC group with whom I had a chance to interact and collaborate at various points in my career: Vasillis Athitsos, Matheen Siddiqui, John Isodoro and Romer Rosales.

On a more personal level, I would like to thank my closest friends without whom this endeavor would not be half as much fun: Stan Rost (*a.k.a.* Progressor) for always making sure that I *do the right thing*, Max Frenkel and Arthur Furman for making sure that I get out of the house for beer once in a while; Natalya Ganchina, Filipp Rakevich, Irina Asipenko and Pasha Volkov for making their homes and refrigerators always open to me.

Lastly to a large extent for the success of this thesis I must thank my family. My parents, Yelena and Alexander Sigal, from an early age have taught me to value education and to pursue my dreams. While they may not have always agreed with my decisions or choices, they always stood by me and supported me along the way. My sister, Marina Sigal, is always someone I can talk to about my struggles and achievements. However, more importantly, my days would not be complete without her *funny stories*. Finally, this thesis would not be possible without the moral support and patience of my soon to be wife, Sofya Bubentsova; I want to thank her deeply for sharing with me all the successes and hardships along this lengthy but rewarding journey.

TABLE OF CONTENTS

List of Tables	xi
List of Illustrations	xiv
List of Algorithms	xv
1 Introduction	2
1.1 Object Localization and Tracking	3
1.2 Articulated Pose Estimation and Tracking	5
1.3 Challenges	6
1.4 Thesis Outline	9
1.5 List of Related Papers	11
2 State of the Art	12
2.1 Common Assumptions	12
2.2 Humans at Different Scales	15
2.3 Categorization of Approaches	15
2.4 Representing the Body	18
2.4.1 Kinematic Tree	19
2.4.2 Scale Prismatic Model	20
2.4.3 Part-based Representation	21
2.5 Image Features	22
2.5.1 Silhouettes	22
2.5.2 Color	23
2.5.3 Edges	23
2.5.4 Contours	24
2.5.5 Ridges	24
2.5.6 Image Flow	24
2.5.7 Voxels	25
2.5.8 Image Descriptors	26
2.6 Pose Estimation and Tracking	27
2.7 Discriminative and Generative Methods	27
2.8 Optimization Methods	28
2.9 Number of Views	32

2.9.1	Multiocular 3D Inference	33
2.9.2	Monocular 3D Inference	33
2.9.3	Sub-space Methods	35
2.10	Quantitative Evaluation	35
2.11	Generic Object Detection, Localization and Categorization	36
2.11.1	Sliding Window Classifiers	36
2.11.2	Part-based Models	37
2.11.3	Hierarchical Composition Models	38
3	Graphical Models and Inference	39
3.1	Graphical Model Building Blocks	40
3.1.1	Exponential Family	40
3.1.2	Gaussian Distribution and Properties	41
3.2	Bayesian Networks	42
3.2.1	Markov Chains	43
3.2.2	Hidden Markov Models	44
3.2.3	Generative and Discriminative Graphical Models	45
3.3	Undirected Graphical Models	46
3.3.1	Markov Random Fields	46
3.3.2	Pair-wise Markov Random Fields	48
3.3.3	Factor Graphs	49
3.4	Parameter Estimation	50
3.4.1	Maximum Likelihood	50
3.4.2	Expectation-Maximization	52
3.4.3	Parameter Estimation with Hyperpriors	54
3.5	Inference	55
3.5.1	Variable Elimination	56
3.5.2	Belief Propagation	58
3.6	Monte Carlo Methods	62
3.6.1	Importance Sampling	62
3.6.2	Kernel Density Estimation	63
3.6.3	Markov Chain Monte Carlo	64
3.6.4	Sequential Importance Sampling	68
3.7	Particle Message Passing	73
3.7.1	Sampling from a Product of Gaussian Mixtures	75
3.7.2	Sampling from More General Forms of Message Foundation	75
3.7.3	Choice of Importance Functions	77
3.7.4	Stratified Sampling	78
3.7.5	Differences between PAMPAS and NBP	79
3.7.6	Message Passing Scheduling	81
3.7.7	Simulated Annealing	81
3.7.8	Examples	81

3.8	Discriminative Models	82
3.8.1	Linear, Ridge and Locally Weighted Regression	88
3.8.2	Bayesian Mixture of Experts	89
3.8.3	Joint-based Learning for Mixture of Regressors	92
4	Graphical Object Models	97
4.1	AdaBoost	99
4.1.1	Bootstrapping	100
4.2	Graphical Object Models	102
4.2.1	Building the Graphical Model	103
4.2.2	Learning Spatial and Temporal Constraints	104
4.2.3	AdaBoost Image Likelihoods	104
4.2.4	Inference using Belief Propagation	111
4.2.5	Proposal Process	112
4.3	Experiments	112
4.3.1	Multi-frame Single Target Detection and Tracking	112
4.3.2	Single Frame Multi-target Detection	114
4.4	Conclusion and Discussion	115
5	Loose-limbed Body Model	117
5.1	Previous Work	118
5.2	Loose-limbed Body Model	121
5.3	Constraints	122
5.3.1	Kinematic Constraints	123
5.3.2	Penetration Constraints	126
5.4	Image Likelihoods	127
5.4.1	Foreground Likelihood	127
5.4.2	Edge Likelihood	129
5.4.3	Combining Features	130
5.5	Bottom-up Part Detectors	130
5.5.1	Head Detection	130
5.5.2	Limb Detection	131
5.6	Inference	133
5.6.1	Tracking	138
5.7	Experiments and Evaluation	139
5.7.1	HumanEva-I Dataset	139
5.7.2	Evaluation Metric	139
5.7.3	Pose Estimation	141
5.7.4	Tracking	142
5.7.5	Comparison with Annealed Particle Filter	148
5.7.6	Analysis of Failures	153
5.7.7	Discussion of Quantitative Performance	153

5.7.8	Analysis of Runtime Speed	156
5.8	Conclusion and Discussion	157
6	Hierarchical Approach for Monocular 3D Pose-Estimation and Tracking	158
6.1	Previous Work	161
6.2	Modeling a Person	163
6.3	Finding an Articulated Pose of a Person in 2D	164
6.3.1	Likelihood	164
6.3.2	Modeling Constraints	167
6.3.3	Inference	169
6.4	Proposing 3D Body Model from 2D	172
6.5	Tracking in 3D	174
6.6	Experiments	175
6.6.1	Monocular 2D Pose Estimation	176
6.6.2	Monocular 3D Pose Estimation	178
6.6.3	Monocular 3D Tracking	181
6.7	Conclusion and Discussion	181
7	Summary and Discussion	186
7.1	Future Work	187
7.1.1	Faster Inference Algorithms	187
7.1.2	Deeper Hierarchical Models	187
7.1.3	Learning of Model Structure	188
7.1.4	Scene Parsing	188
7.2	Conclusions	189
	Bibliography	190

LIST OF TABLES

2.1	Common assumptions made by articulated (human) pose estimation and tracking algorithms	13
2.2	Categorization and comparison of articulated human pose and motion estimation approaches	16
3.1	Inference using Belief Propagation	59
3.2	Kernel density estimators	64
5.1	Comparison of loose-limbed body model to other generative approaches.	119
5.2	Summary of pose estimation performance using loose-limbed body model	142
5.3	Summary of tracking performance using loose-limbed body model	148
5.4	Runtime speed of inference	156

LIST OF ILLUSTRATIONS

1.1	Localizing and tracking rigid objects in video	3
1.2	Articulated pose estimation	4
1.3	Hierarchical articulate 3D pose inference from monocular image(s)	6
1.4	Challenges in localizing and tracking objects in video	7
1.5	Challenging human motion	8
2.1	Representing the body	19
2.2	Common features used for articulated pose and motion estimation	22
2.3	Generative and discriminative pose estimation and tracking	27
3.1	Graphical model families	40
3.2	Baysian Networks	43
3.3	Markov Chain	43
3.4	Hidden Markov Models	44
3.5	Generative and discriminative graphical models	45
3.6	Markov Random Field	47
3.7	Pair-wise Markov Random Field	48
3.8	Gaussian Mixture Model Illustration	52
3.9	Graphical Models for Gaussian and Gaussian Mixture Model.	53
3.10	Gaussian Mixture Model with Hyperpriors	55
3.11	Belief Propagation	60
3.12	Kernel density bandwidth estimation	63
3.13	Non-parametric representation of distribution (coarse)	65
3.14	Non-parametric representation of distribution (fine)	66
3.15	Particle Message Passing in Hidden Markov Model	83
3.16	Particle Message Passing in pair-wise MRF	84
3.17	Particle Message Passing in pair-wise MRF with missing data	85
3.18	Particle Message Passing in Hidden Markov Model (with poor dynamical prior)	86
3.19	Particle Message Passing in pair-wise MRF (with poor dynamical prior)	87
3.20	Regression model	88
3.21	Mixture of experts model	90
3.22	Mixture of kernel regressors example	93
3.23	Mixture of kernel regressors example	94
3.24	Mixture of kernel regressors example	95

4.1	Variation within the class of vehicles	98
4.2	AdaBoost filters	100
4.3	Graphical models for the pedestrian and vehicle detection and tracking	105
4.4	Modeling spatial constraints	106
4.5	AdaBoost detector for the head	107
4.6	AdaBoost detector for the left side of an upper body	108
4.7	AdaBoost detector for the right side of an upper body	109
4.8	AdaBoost detector for the lower body	110
4.9	Vehicle component-based spatio-temporal object detection and tracking	113
4.10	Pedestrian component-based spatio-temporal object detection	114
4.11	Multiple target detection	115
5.1	Graphical model for a person	118
5.2	10-part and 15-part loose-limbed body models for a person	121
5.3	Parameterization of a 3D body part	122
5.4	Learned kinematic potentials	123
5.5	Backprojecting the 3D body model	128
5.6	Image likelihoods	129
5.7	Head detection	132
5.8	Limb detection	133
5.9	Message product for the torso	134
5.10	Illustration of convergence of loose-limbed body model during pose estimation	136
5.11	Virtual marker-based evaluation metric	140
5.12	Evaluation metric illustration	141
5.13	Pose estimation using 10-part loose-limbed body model	143
5.14	Pose estimation using 10-part loose-limbed body model	144
5.15	Pose estimation using 15-part loose-limbed body model	145
5.16	Pose estimation using 10-part loose-limbed body model	146
5.17	Pose estimation using 15-part loose-limbed body model	147
5.18	Tracking using 15-part loose-limbed body model	149
5.19	Tracking using 15-part loose-limbed body model	150
5.20	Tracking using 10-part loose-limbed body model	151
5.21	Tracking using 15-part loose-limbed body model	152
5.22	Comparison with annealed particle filter	154
5.23	Failure modes	155
6.1	Typical discriminative inference process	159
6.2	Example of the hierarchical inference process	159
6.3	Silly walks	160
6.4	Fighting the likelihood	161
6.5	Representing 2D body as a graph	165
6.6	Occlusion-sensitive likelihood	168

6.7	Occlusion-sensitive inference	170
6.8	Hierarchical inference	173
6.9	Proposed 3D pose	175
6.10	Quantitative performance evaluation of 2D pose estimation	176
6.11	Overall performance comparison of 2D pose estimation	177
6.12	Visual performance evaluation of 2D pose estimation	178
6.13	Occlusion-sensitive reasoning in movies	179
6.14	Learning parameters for conditional models	180
6.15	Quantitative evaluation of action-specific conditional model	182
6.16	Hierarchical 3D pose estimation	183
6.17	Monocular tracking in 3D	184
6.18	Comparison of monocular 3D pose estimation with tracking	184

LIST OF ALGORITHMS

1	Expectation-Maximization for Gaussian Mixture Model	54
2	Metropolis Sampler	67
3	Gibbs Sampler	69
4	Generic Particle Filter	72
5	Gibbs Sampler for a Product of Gaussian Mixtures	76
6	PAMPAS Stratified Message Update	80
7	AdaBoost Classifier Learning	101
8	Bootstrap Learning of AdaBoost Classifier	102

Abstract of “**Continuous-state Graphical Models for Object Localization, Pose Estimation and Tracking**” by **Leonid Sigal**, Ph.D., Brown University, May 2008.

Reasoning about pose and motion of objects, based on images or video, is an important task for many machine vision applications. Estimating the pose of articulated objects such as people and animals is particularly challenging due to the complexity of the possible poses yet has applications in computer vision, medicine, biology, animation, and entertainment. Realistic natural scenes, object motion, noise in the image observations, incomplete evidence that arises from occlusions, and high dimensionality of the pose itself are all challenges that need to be addressed. In this thesis we propose a class of approaches that model objects using continuous-state graphical models. We show that these approaches can be used to effectively model complex objects by allowing tractable and robust inference algorithms that are able to infer pose of these objects in the presence of realistic appearance variations and articulations.

We use continuous-state graphical models to model both rigid and articulated object structures; where nodes correspond to parts of objects and edges represent the constraints between parts encoded as statistical distributions. For rigid objects, these constraints can model spatial and temporal relationships between parts; for articulated objects kinematic, inter-penetration and occlusion relationships. Localization, pose estimation, and tracking can then be formulated as inference in these graphical models. This has a number of advantages over more traditional methods. First, these models allow inference algorithms that scale linearly with the number of body parts by breaking up the high-dimensional search for pose into a number of lower-dimensional collaborative searches. Secondly, partial occlusions can be dealt with robustly by propagating spatial information between parts. Thirdly, “bottom-up” information can be incorporated directly and effectively into the inference process, helping the algorithm to recover from transient tracking failures. We show that these hierarchical continuous-state graphical models can be used to solve the challenging problem of inferring the 3D pose of the person from a single monocular image.

CHAPTER 1

Introduction

Images and video provide rich low-level cues about the scenes and the objects in them. The goal of machine vision is to develop approaches for extracting meaningful semantic knowledge from these low-level cues; for example, in the case of robotics, allowing direct interaction of the computer with the real world. This is challenging because of the large variability that exists in imaging conditions and objects themselves. Objects that belong to same semantic classes can appear differently, image differently, and even act differently. Objects like cars vary in size, shape and color; people in weight, body shape and size/age. Motion of these objects is often complex and is governed by physical interactions with the environment (*e.g.* balance, gravity) and higher order cognition tasks like intent.

All these challenges make it impossible to determine the regions of the image that belong to a particular object, or part of the object, directly. Computer vision algorithms must propagate information both spatially and temporally, to effectively resolve ambiguities that arise, by inferring globally plausible and temporally persistent interpretations. Statistical methods are often used for these tasks, to allow reasoning in the presence of uncertainty. *Graphical models* provide a powerful paradigm for intuitively describing the statistical relationships precisely and in a modular fashion. These models effectively represent statistical and conditional independence relationships between variables, and allow tractable inference algorithms that make use of encoded conditional independence structure. In computer vision, inference algorithms for these graphical models need to be developed to handle the high-dimensionality of the parameter-space, complex statistical relationships between variables and the continuous nature of the variables themselves.

This thesis will concentrate on localizing, estimating the pose of and tracking rigid and articulated objects (most notably people) in images and video. Estimating the pose of people is particularly interesting because of a variety of applications in rehabilitation medicine, sports and the entertainment industry. Pose estimation and tracking can also serve as a front end for higher level cognitive reasoning in surveillance or image understanding. Localizing and tracking articulated structures like people, however, is challenging due to the additional degrees of freedom imposed by the articulations (compared with rigid objects). In general the search space grows exponentially with the number of parts and the degrees of freedom associated with each joint connecting these parts, making most straight forward search algorithms intractable. The recurring theme of this thesis will be the merge of Monte Carlo sampling and non-parametric inference methods with graphical models, resulting in tractable and distributed inference algorithms for localizing and tracking objects in 2D and 3D. We will also advocate the use of a hierarchical inference approach for mediating the

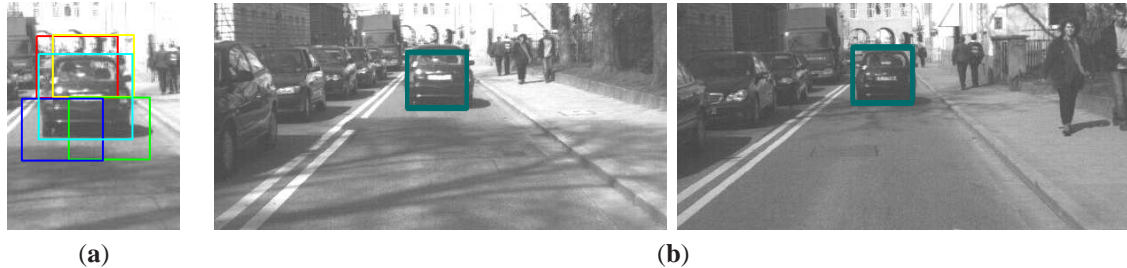


Figure 1.1: **Localizing and tracking rigid objects in video.** In (a) part-based representation of a vehicle class object is shown. Object itself is shown in cyan and 4 rigid image-based parts in terms of which it is modeled in red, yellow, green and blue. Results of localizing and subsequently tracking the object through a short sequence are shown in (b). Results on two representative frames, 50 frames apart, obtained from the car-mounted moving camera are shown. Notice the variation in lighting in the two video frames.

complexity of harder inference problems.

We will first describe the problem of pose estimation and tracking as it applies to rigid and articulated objects. We will then describe a kinematic model and the corresponding Monte Carlo sampling methods, which have successfully been applied to track articulated objects given an initial pose (often supplied manually at the first frame). We will then consider a more general problem of tracking people automatically, by first inferring the pose of the person and then incorporating temporal consistency constraints in a collaborative inference framework. We will show that we have made contributions in all aspects of this problem by addressing modeling choices, inference, likelihoods and priors.

1.1 Object Localization and Tracking

The most natural use of machine vision is to detect, recognize, localize and track objects in the scene. *Detection* deals with finding if objects are present, *recognition* with finding what objects are present, *localization* with finding where they are, and *tracking* with following them as they move in the scene. In this thesis we will concentrate on localization and tracking and to some extent detection¹. Recognition is an interesting problem in its own right and we refer the reader to [57, 60, 61, 224] for some of the latest work in this research area. In *localization*, the goal is to find the pose of the object. For example, the pose of rigid objects can often be described in terms of 3D position and orientation of the object in the scene, *i.e.* a vector $\in \mathbb{R}^6$. Depending on the task it may also be sufficient to describe the pose of the object in the image plane in which case only 4 parameters are needed: 2D position, orientation, and scale. The latter representation is more suited for presence/absence detection, whereas the former is more natural for spatial reasoning in the scene.

Tracking deals with finding the pose of an object at every frame in the image sequence. In tracking, models of motion/dynamics for objects are often used to robustly and efficiently localize them given the short history of estimates from previous frames. Tracking can be (and sometimes is [173]) replaced by localization at every frame. While this ensures that estimates are not subject to drift (accumulation of error resulting from propagating estimates from frame to frame), it often produces very noisy results. Incorporating temporal

¹Since most generative approaches tend to model the location of the object along with appearance of the object itself, detection and localization are often one and the same. Hence from now on we will tend to use these two terms interchangeably. There are some detection algorithms that are specifically designed to be invariant to the location of the object. In such cases a separate localization stage is needed to pinpoint where the object is in an image once its presence is established.

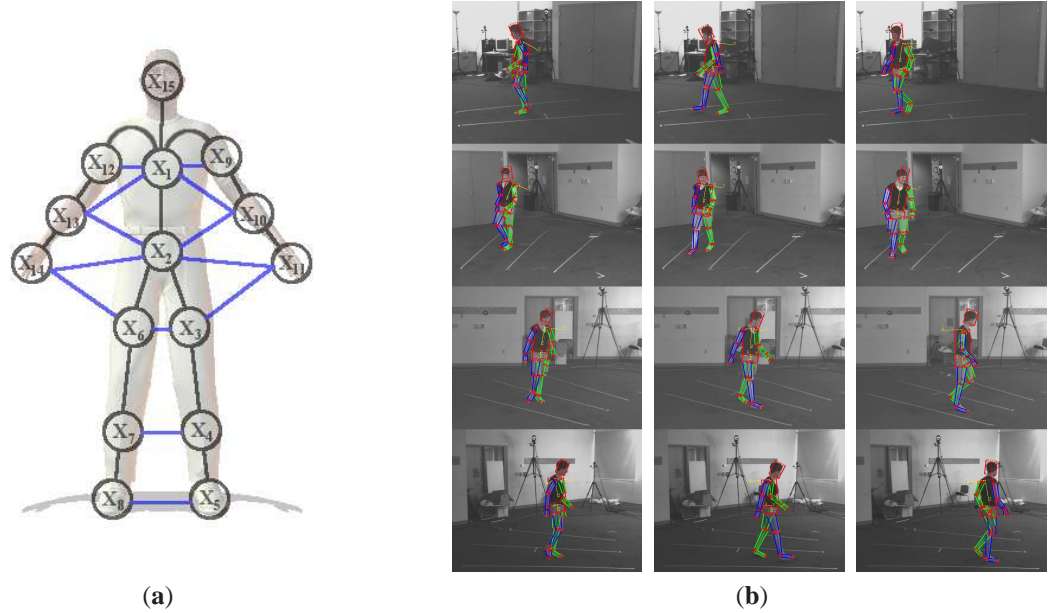


Figure 1.2: **Articulated pose estimation.** Figure (a) shows the loose-limbed body model used for inference of articulated pose and tracking; (b) shows the results of applying the model in (a) for inference of 3D pose from a single 4-view image. The 3 instances of the model being applied are shown from left to right in (b), with projections of the 3D model onto each of the 4-views from top to bottom.

consistency tends to smooth and regularize the results especially in the presence of inter-frame appearance variations.

One of the biggest challenges in object detection and localization is the variability in appearance, size and shape of objects. It has been shown however [59, 62, 144, 249] that for some classes of objects this variation is mostly due to the placement and not the appearance of individual parts. Lets take for example vehicles; while vehicles may look different (see Figure 1.4 (a)), they all have similar parts like the bumper, hood, and headlights, and differ mostly in the relative placement/arrangement of these parts.

Armed with this intuition we model objects using a part-based graphical model representation. We use two-layer graphical models to model two classes of objects, pedestrians and vehicles. In this framework we combine object detection/localization with tracking in a single unified framework, which allows us to achieve more robust solutions to both problems. Tracking can make use of object detection for initialization and re-initialization during transient failures, while object detection can benefit from the temporal consistency provided by the tracking over time. Modeling objects by arrangement of imaged-based parts that are spatially constrained (using learned statistical dependencies encoded by edges in our graphical model), facilitates detection, localization, and tracking of rigid objects under local deformations, partial occlusions and local lighting variations. This results in a tractable unified framework that shows promise for simultaneous object detection/localization and tracking. Examples of the results obtained using our model are shown in Figure 1.1.

1.2 Articulated Pose Estimation and Tracking

Articulated objects consist of a number of rigid parts connected by joints. Examples of such objects include people², animals² and man-made machines. In this thesis we will concentrate primarily on *people*, while similar approaches can be applied to other articulated objects (*e.g.* animals [170], hands [219], *etc.*). The pose of the articulated object refers not only to the position and orientation of the object in the scene but also to the configuration that it assumes. In the case of people this corresponds to posture, and is most often described by a set of parameters that encode the global 3D position and orientation of the torso in the scene, and 3D joint angles that account for 3D rotation of each limb relative to the torso. This results in a state-space vector representation of the pose $\in \mathbb{R}^d$, where $d \in \{30, \dots, 60\}$ depending on granularity of the model. A slightly more compact representation can be obtained by looking at the pose of the body in the image plane rather than the scene. In both cases, and even at coarse granularity, this leads to very high-dimensional continuous representation of the pose. Searching for the pose in this high-dimensional state-space using standard methods, which often scale exponentially with dimensionality, quickly becomes intractable.

One way of battling the high-dimensionality is using local search techniques [52] with good initialization; this is an approach most articulated tracking algorithms have taken in early years. This of course assumes that a good initialization is available or can be obtained from a cooperating subject via a predefined procedure. This is ineffective, however, if initialization is unavailable or the subject is unaware, which is often the case if our goal is to build autonomous machine vision systems. One alternative is to apply a dimensionality reduction technique and search for the pose in lower dimensional space. While there are clearly correlations between body parts that allow balance and coordination, the human pose manifold is complex and cannot effectively be modeled using linear low-dimensional embeddings like Principle Components Analysis (PCA) [228]. Even more sophisticated methods like Locally Linear Embedding (LLE) [55] or Gaussian Processes [226, 227] usually require motion to be constrained to a single relatively simple class of actions (*e.g.* walking [55, 227], running, golf swing [227], *etc.*) to learn a good low-dimensional representation. Video sequences provide additional temporal constraints that often help regularize single frame estimates, and can significantly reduce the search time by ruling out large portions of the search space.

Instead of attempting to battle the dimensionality of the state-space and complexity of motion directly, we formulate the problem of pose estimation and tracking as one of inference in a graphical model. The nodes in this graph correspond to parts (or limbs) of the body and edges to kinematic, inter-penetration and occlusion constraints imposed by the structure of the body and the imaging process. This model, which we call a *loose-limbed body model*, allows us to infer the 3D pose of the body effectively and efficiently from multiple synchronized views; or a 2D pose of the body from a single monocular image, in time linear in the number of articulated parts. Since discretization of rotation and position in 3D space is implausible³ we work directly with continuous variables, and use variants of Particle Message Passing (PAMPAS) [99] for inference.

Discretization in 2D is possible [59, 169], due to the lower-dimensionality and the more natural discrete representation of the pixel grid. However, to ensure that the inference is tractable, the structure of the discrete

²Actually people and animals have only approximately rigid parts. For the purposes of this thesis, however, we will assume rigidity and ignore non-rigid skin and muscle deformations.

³Discretizing moderate $5\text{ m} \times 5\text{ m} \times 2\text{ m}$ space even coarsely at granularity of 10 cm and 10 degrees , would require $36 \times 36 \times 36 \times 50 \times 50 \times 20 = 2.3$ billion bins.

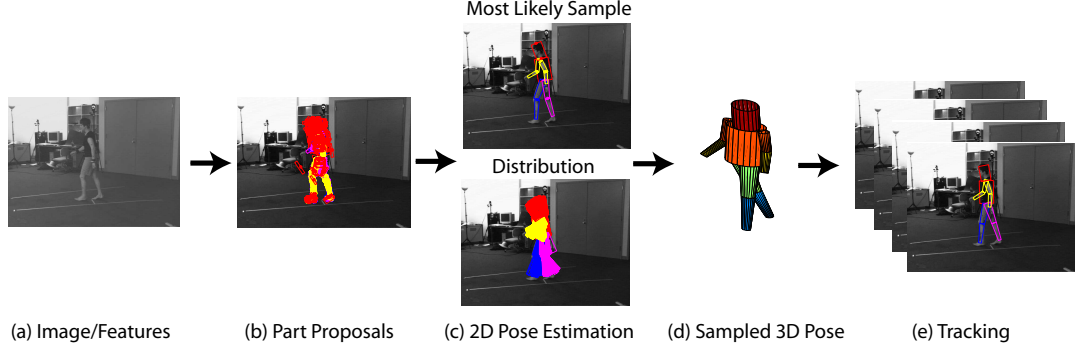


Figure 1.3: **Hierarchical articulate 3D pose inference from monocular image(s).** (a) monocular input image with bottom up limb proposals overlaid (b); (c) distribution over 2D limb poses computed using non-parametric belief propagation; (d) sample of a 3D body pose generated from the 2D pose; (e) illustration of tracking.

graphical model has to be reduced to a tree, for which fast algorithms exist [59]. These tree-structured models, however, are unable to represent important occlusion relationships that require long range interactions between left and right sides of the body. This results in models for which maximum a posteriori (MAP) estimates often prefer incorrect solutions [122, 196]. To deal with this, we propose an extension to our loose-limbed body model that explicitly accounts for occlusions [196] using per-pixel binary variables. The developed inference algorithm works over loopy graphs, accounts for occlusions, and can tractably infer the pose with marginal overhead compared with continuous-state tree structured model.

Sometimes it may be useful to infer articulated 3D pose from a single monocular image. This most general case is challenging because of the inherent depth ambiguities. Even with perfect observations and moderate assumptions on the size and shape of the body, the 3D pose of individual limbs is too unconstrained to be modeled effectively even using non-parametric methods. Instead, we introduce a hierarchical inference framework, where we first infer the 2D pose of the body in the image plane, then infer the 3D pose from the 2D body pose estimates and lastly apply the temporal continuity (tracking) at the 3D pose level. This leads to two important benefits: (1) it helps to reduce the depth and projection ambiguities by looking at a full 2D body pose rather than the pose of individual limbs, and (2) it gives modular, tractable and fully probabilistic solution that allows inference of 3D pose from a single monocular image in the unsupervised fashion.

The presented framework is more general than person pose estimation or tracking. It represents an instance of a more general hierarchical inference process for object detection, where different levels of representation cooperate in inferring the scene using a probabilistic framework. In this framework complex objects are described using a hierarchy of simpler representations; for example, objects can be represented by collections of parts, parts by collections of features, and features by responses of simple operators applied to the image.

1.3 Challenges

Complex appearance and motion of objects as well as imaging conditions lead to many challenges for vision approaches that attempt to localize, estimate the pose of and track objects. Some of these challenges are inherent and result in ambiguities that can only be resolved with prior knowledge; others lead to computational

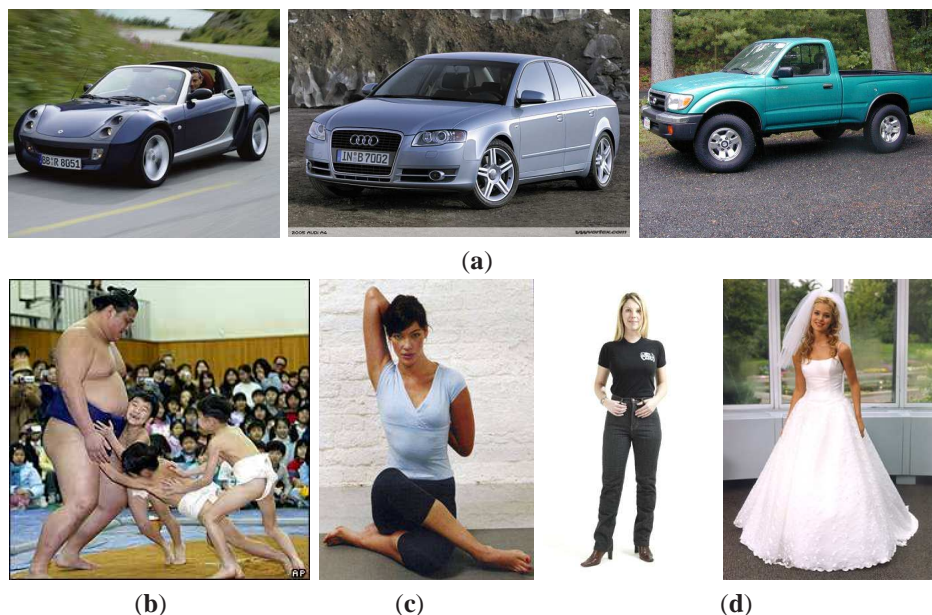


Figure 1.4: **Challenges in localizing and tracking objects in video.** Top row (a) shows the variation in the appearance of rigid object, *cars*; bottom row shows the shape variation (b), self-occlusions (c), and effects of clothing (d) on the articulated objects, *people*.

burdens that require clever engineering solutions. We will describe some of these challenges in this section.

Differences in appearances and shape. Similar objects can vary significantly in physical size, shape, texture and color. Figure 1.4 shows the large variation in the class of (mostly) rigid objects such as cars (a), and even more severe variation in articulated objects such as people in (b). In (b) the sumo wrestler appears at least twice the size of the children, and is likely more than 4 times the weight. These severe variations in size and shape will also result in the differences in motion, often resulting in a more agile motion for slimmer and lighter objects. A good tracking system should then not only be robust to these variations, but rather embrace and make use of them in the form of important distinguishing cues and prior models of motion.

World-occlusions. Object rarely appear by themselves, outside of a laboratory environment. In realistic scenes objects often interact with their environment and other objects which results in occlusions. During occlusions, the appearance of the object is only partially observed and important information that allows reasoning about its state can be missing. In such cases (assuming that they can be detected, which is in itself a hard problem) vision approaches are forced to infer the state and appearance of the object with partially missing data, based on the prior knowledge or by spatial (or temporal) information aggregation.

Self-occlusions. Articulated objects have an additional complexity of being able to self occlude. This is illustrated in the Figure 1.4 (c), where the hands and a significant part of the arm are occluded by the torso and the head. Both world and self-occlusions can be to some extent resolved by synchronously observing the scene and the object from multiple viewpoints, assuming the viewpoints are not degenerate. It can be shown that as number of views grows, the visual hull, defined by carving away parts of the space that are inconsistent with all image views, approaches the true shape of the object [113]. Inferring the pose of the person from multiple views hence is inherently an easier (but often more computationally intensive) problem.

Projection ambiguities. Depth information is lost when 3-dimensional objects in the scene are projected onto the 2-dimensional image plane. This leads to a number of depth and projection ambiguities. As a result



Figure 1.5: **Challenging human motion.** This clip shows an exaggeration of the complexities that a simple walking motion can exhibit when stylized by John Cleese in the episode of the Monty Python’s Flying Circus in 1970. Images were taken from <http://www.univie.ac.at/cga/art/tv.html>.

at best only the relative size of the objects can be recovered, unless something is known *a priori* about the absolute size of one or more objects in the scene. Out of plane rotations also become ambiguous, with both backward and forward rotations able to account for foreshortening in projection. Lastly motions that are along a tangent vector to the image plane may be significantly harder to observe than lateral motions in the image plane.

Kinematic ambiguities. A less intuitive ambiguity arises in articulated objects that have symmetric parts, for which the axis of symmetry is also the axis (or one of the axes) of rotation (*e.g.* arms or legs of a person). In such cases the rotations along these axes (often referred to as *twist*) is nearly unobservable in the image and hence is inherently unrecoverable. Kinematic ambiguities may or may not persist over the extent of the motion. They may arise for some configuration of the body and not for others. For example, consider a straight arm; twisting an upper arm produces almost no difference in the appearance of the body in an image. Now think of the same experiment but with the elbow bent 90 degrees, twisting an arm now would produce a significant variation in an image and hence make the twist of the upper arm much easier to recover.

Kinematic singularities. Kinematic singularities arise due to the typical parameterization of articulated pose in terms of 3D joint angles. Since decomposition of 3D rotational degrees of freedom is not unique, often a single configuration of the joint can be described by two or more different sets of parameters (joint angles). This leads to multi-modal solutions that are difficult to handle using direct optimization methods.

Clothes. Clothing can significantly influence how we perceive articulations of the body. Tightly fitting clothes make observations about the location of limbs easier, loose-fitting clothes on the other hand often obstruct our view of the limbs, making accurate observations impossible. This is illustrated in Figure 1.4 (d). Notice that even when clothes are not present, direct observation of joints and bones is impossible due to layers of body tissue and skin. Hence, vision approaches always face the problem of inferring joint location and orientation from indirect observation of the body.

High dimensionality. While the pose of the rigid object can perhaps be expressed in terms of position and orientation in space resulting in a state-space representation $\in \mathbb{R}^6$ (for 3D pose), the pose of the articulated object such as a person, that has many rigid parts connected at joints, will have to be expressed in \mathbb{R}^{30} or

higher depending on granularity of the model. Some models of human motion that try to achieve more realistic representation (e.g. POSER) use as many as 60 parameters, resulting in the state-space $\in \mathbb{R}^{60}$. Searching for the parameters in this high-dimensional state-space without a good initialization is a very challenging problem.

Viewpoint. Viewpoint can have a dramatic effect on appearance of any object due to the asymmetries of most shapes. This is true even for simple geometric objects. For example, consider a cylinder viewed directly from the side: it looks like a rectangle in the image plane, from the top - a circle. In these degenerate cases image observations alone are not enough to distinguish the cylinder from other simple 3D geometric shapes, e.g. sphere or cuboid. Observing the cylinder as it or the camera moves may help resolve this ambiguity.

Lighting. Lighting also plays a significant role in the imaging process. The most intuitive artifact is inability to observe parts of the image due to the under or over-exposure that may be a result of poor lighting conditions or reflective/specular properties of the object. The less intuitive artifact is shadows. Shadows are often hard to distinguish from objects that cast them for two important reasons. First, shadows are dynamic entities that change with the objects as they move. Hence, using techniques such as background subtraction to discount shadows is ineffective. Second, shadows often have very similar shape to the objects that cast them. Disambiguating shadows from the objects often requires modeling of more complex object properties like texture and/or color, and sometimes even the geometry of the scene.

Complexity of human motion. Human motion itself is very complex. The human body consists of many joints of various types, with different degrees of freedom and ranges of motion. There exist complex correlations between joints that allow dynamic and static balance of the body. There is also a large set of actions that a person can perform and an even larger set of styles [225, 240] in which these actions can be performed. Figure 1.5 shows one example of a very complex motion that results from a skillfully stylized simple action of walking. The complexity and the variability of the human motion, in general, allow few assumptions about the content and dynamics of motion present in images or video. Strong prior models, that make aggressive decisions about the pose or motion in absence of image evidence, while computationally efficient and often helpful in constraining the problem, are also easily violated in realistic scenarios.

Addressing all these challenges is essential to building an accurate, robust and reliable object detection, localization and tracking system. In this thesis we will address some of these challenges explicitly, including high dimensionality, complexity of human motion, self-occlusions, kinematic and projection ambiguities; others such as clothing and shape variations are still left largely unaddressed by the vision community.

1.4 Thesis Outline

Chapter 1. Introduction. The chapter introduces and motivates the thesis, outlines the key ideas and contributions. The chapter also introduces the problems of object detection, articulated pose estimation and tracking. Challenges in these problems are discussed along with motivations for solving them. The chapter also gives an overview of the overall thesis structure.

Chapter 2. State of the Art. This chapter will cover the basics of rigid and articulated object detection, pose estimation and tracking. Kinematic tree models [139] and approaches for articulated tracking using the kinematic tree models including direct optimization methods and Monte Carlo integration methods [54] such

as particle filtering [52, 136] will be discussed. Common features used for both articulated and rigid object inference will also be introduced and discussed. The chapter will also cover differences between bottom-up and top-down approaches to pose estimation and tracking, including regression [4], mixture of experts [206], and nearest-neighbor methods [189]. Differences and advantages of discriminative versus generative methods will also be addressed. Lastly, well established approaches for object detection and categorization including bag of features, AdaBoost [235, 236], constellation model [61] and pictorial structures [59] will be introduced and discussed among others.

Chapter 3. *Graphical Models and Inference.* This chapter will introduce the formalism and statistical methods that play a central role in the thesis. Graphical models [107, 108] will be discussed, including directed, undirected and loopy graphs. Special purpose graphical models that are common to problems of tracking including Hidden Markov Models (HMMs) will be covered. Sampling-based inference methods in graphical models, including Markov Chain Monte Carlo (MCMC) methods like Importance Sampling (IS), Particle Filtering, Gibbs sampling and Kernel Density Estimation (KDE) will be presented. Some theoretical results and limitations of inference in graphical models using leading approaches like Variable Elimination and Belief Propagation will also be discussed. Continuous-state graphical models will be covered in depth, along with approximate inference algorithms of Particle Message Passing (PAMPAS) [99] and Non-parametric Belief Propagation (NBP) [220] developed for those models. A number of extensions that we developed [196, 197, 199] to the standard formulation of Particle Message Passing will also be covered. Among them, mixture models for potential functions, annealing, importance sampling and inference in the presence of hidden variables by analytic marginalization. As part of the discussion on continuous-state graphical models, efficient methods for sampling from the products of Gaussian mixtures [78, 95] will be addressed.

Chapter 4. *Graphical Object Models.* This chapter will motivate the use of graphical models for modeling rigid objects. Based on the formalism introduced in the previous chapter, an inference algorithm for a class of two-layered graphical models, used to model objects [198] in the proposed framework, will also be introduced. Chapter will include experiments on detecting and tracking pedestrians and vehicles geared towards automated vehicle navigation applications. The chapter will conclude with discussion of findings and results.

Chapter 5. *Loose-limbed Body Model.* This chapter will introduce the key contribution, of what we call a loose-limbed body model [199]. It will illustrate its application for 3D pose estimation and tracking from multiple synchronized views [197]. The comparison and relationship to other relevant methods will also be discussed at length. In addition, we will also discuss a novel dataset and methodology for quantitative evaluation of performance. The chapter will conclude with discussion of findings and results.

Chapter 6. *Hierarchical Approach for Monocular 3D Pose-Estimation and Tracking.* The chapter will introduce the notion of inference hierarchy to mediate the complexity of inferring a 3D articulated pose from a single monocular image. It will introduce the formalism for the mixture of experts model [195] used to infer the 3D pose from the 2D articulated pose obtained using the variant of loose-limbed body model discussed in the previous chapter. As part of this effort we will introduce formulation for novel occlusion-sensitive likelihoods [196], to account for occlusions between 2D parts. The chapter will conclude with discussion of findings and results.

Chapter 7. Summary and Discussion. This chapter will summarize the contributions of the thesis, discuss open issues and possible future directions.

1.5 List of Related Papers

The thesis is based on the material from the following published papers, listed in order of relevance.

L. Sigal, S. Bhatia, S. Roth, M. Black and M. Isard. Tracking Loose-limbed People. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 421–428, 2004.

L. Sigal and M. Black. Measure Locally, Reason Globally: Occlusion-sensitive Articulated Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 2041–2048, 2006.

L. Sigal and M. Black. Predicting 3D People from 2D Pictures. In *IV Conference on Articulated Motion and Deformable Objects (AMDO)*, Springer-Verlag LNCS 4069, pp. 185–195, 2006.

L. Sigal, Y. Zhu, D. Comaniciu and M. Black. Tracking Complex Objects using Graphical Object Models. In *1st International Workshop on Complex Motion*, Springer-Verlag LNCS 3417, pp. 227–238, 2004.

L. Sigal, M. Isard, B. Sigelman and M. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *Advances in Neural Information Processing Systems 16 (NIPS)*, pp. 1539-1546, 2004.

CHAPTER 2

State of the Art

Object detection, localization and tracking are among the core problems in computer vision. In the past 20 years there has been significant progress in generic object detection and tracking; as well as dealing with specific classes of objects (*e.g.* faces, people, *etc.*). However, the general case of dealing with non-rigid objects that exhibit complex motion patterns and appearance variations is still largely unsolved.

Difficulties in reasoning about the object position and pose arise due to the complexity of objects themselves, variations in their appearance, motion, interactions and imaging conditions. The problem is significantly simplified, for the class of rigid objects, when variation in appearance is only a function of camera position and imaging conditions. Conversely, dealing with articulated objects such as people is difficult, because of their inherent non-rigid articulate structure, complexity of motions, variations in body shape, and interactions with the environment. The problem of reasoning about people is further complicated by the higher-level applications in the context of which this reasoning must be done. These applications often require full high-dimensional articulated pose of the subject at every frame, in order to draw inferences about the motion or intent.

In this thesis the goal is to introduce a new class of models that can effectively deal with modeling and drawing inferences about complex and articulated objects. Due to the vast amount of literature on the subject, in this chapter we will concentrate on reviewing the literature on the articulated human motion and pose estimation. However, most of the approaches and concepts introduced in the context of articulated motion also apply in the more generic case of rigid objects. We will draw references to generic object detection and localization where appropriate. We will briefly review the state of the art in generic object recognition in Section 2.11.

2.1 Common Assumptions

Human detection, pose estimation and tracking are all difficult problems. While there have been hundreds of methods introduced that attempt to solve these problems in a variety of ways and settings, none exist that can deal with clothed people wearing unknown clothing, moving arbitrarily in a complex environment. To address these difficulties, many assumptions have been made in prior literature to simplify the problem. The typical assumptions can be divided into four sub-categories (see Table 2.1) that deal with the *Environment*, *Camera*, *Person*, and *Motion*.

Environment	Camera
No lighting variation	Known camera parameters
Static known background	Static camera (or motion of camera is known)
Uncluttered background	Camera view is fixed relative to the person
Only one person is present	<ul style="list-style-type: none"> • Motion is lateral to the camera plane • Motion is frontal to the camera plane • Height of the camera is fixed • Face is always visible
Subject	Motion
Known initial pose	Subject remains visible at all times
Known subject	Slow and continuous movement
Cooperative subject	No self- or world- occlusions
Special type/texture/color clothes	Simple movement (only few limbs move at a time)
Tight-fitting clothes	Known movement

Table 2.1: **Common assumptions made by articulated (human) pose estimation and tracking algorithms.** The assumptions are loosely listed by their frequency in the literature with the most common assumptions listed on top.

Environment assumptions are extremely common and are made by most approaches. The first two assumptions of static lighting and static (or nearly static) background ensure that the background of the scene can be relatively easily modeled, resulting in the ability to reliably estimate the silhouette features obtained by the background subtraction process [1, 36, 52, 55, 59, 77, 113, 122, 189, 197]. In addition, the static lighting assumption also ensures that the overall appearance of the body is stable over time. The assumption of an un-cluttered background allows the use of edge features without being distracted by the background clutter [52, 93, 127, 147, 148, 174]. In essence the first three assumptions ensure that a good set of features can be derived from the image. Assuming that there is only one person present in the scene [1, 36, 52, 55, 59, 77, 113, 122, 189, 195, 196, 197] significantly simplifies the problem of association between image features and subjects. With few exceptions [74], approaches that deal with multiple people often reduce the complexity of feature association by only recovering the rough overall pose (*e.g.* position of the body in space [18, 114], or position of blobs associated with upper and lower portions of the body [162, 163, 259]) rather than the full articulation of the body. In addition, when multiple subjects are present in the scene often the scale of the subjects themselves in the image is reduced, leading to the lack of observations (see discussion in Section 2.2).

Camera assumptions are important in simplifying the models and the dynamics used to model people and their motions. The first assumption of known camera parameters (*a.k.a.* calibrated cameras) is needed in order to be able to project the 3D hypothesis of the body in a given pose into the image. This assumption is critical in any 3D reasoning about the subject’s pose in the world. It has been shown in a few instances, however, that the human motion itself can be used to recover the camera parameters [27, 102, 203]. The second assumption of the static or relatively simple camera motion relates back to the ability to estimate silhouettes that have been shown to be robust and useful image features. The various assumptions about the relative placement of the camera with respect to the moving subject are often made to simplify the variation in appearance and motion. Assuming that motion is lateral to the image plane [111, 250], for example, ensures that there is little if any scale or foreshortening effects that are due to depth variations and/or out-of-plane

rotations. In such cases, often the models of dynamics can also be simplified. The frontal motion [59] while exhibits the scale and foreshortening variation, does not suffer from the symmetry ambiguities introduced by left/right body side similarity. The fixed camera height (or view) assumption [125, 189] is often useful for template or exemplar based approaches [189] where a number of exemplars need to be stored explicitly and matched against the image. In such cases fixing the camera height significantly reduces the appearance variations and hence reduces the number of exemplars needed, rendering such approaches tractable. The last assumption, that the face is visible, is one that became more popular in recent years [93, 126, 127]. The face is by far the most salient feature of the human body, and, unlike other body parts, is often straightforward to find reliably (robust face detectors exist [236]). The head is also rotationally asymmetric; this allows approximate inference of the body orientation [197] from head orientation. Hence, part-based approaches that attempt to detect the body by first detecting the salient parts and then propagating the information spatially to other parts of the body (that may not be as discriminative), often require the presence of the face [93, 126, 127].

Subject assumptions are useful in reducing the number of parameters required to model the person and the variation in their appearance. The known initial pose is a frequent assumption [30, 34, 52, 55, 74, 78, 90, 111, 131, 164, 165, 193, 209, 226, 228, 237, 248] that significantly reduces the search space. Knowing the initial pose also transforms the pose estimation problem into one of tracking, where the pose must be recovered incrementally from frame to frame. The known subject assumption ensures that the shape parameters of the body (*e.g.* height, leg length, *etc.*) are not searched over. This assumption is often introduced for convenience to reduce the dimensionality of the state space of the model, which often times is already very high. A cooperative subject [36, 112, 113] is a somewhat looser assumption than that of the known initial pose or known subject. The idea is that by having a subject perform a set of predefined motions [36] and/or having the subject stand in the predefined pose [113] (usually frontal to the camera with arms and legs spread out, *a.k.a.* ‘T’-pose), relative to the camera, the body shape and the initial pose can be obtained automatically. The last two assumptions of special and/or tight fitting clothing greatly simplify feature matching. For example, by wearing a tight fitting suite [74, 142, 156] that has different parts of the body colored [74] or texture mapped with very distinct patterns [130], finding these parts of the body becomes trivial. Even in the absence of special textures or colors, skin-tight clothes facilitates finding of body parts by ensuring that the contours of the body are easily observed. In general, these last two assumptions are considered too restrictive and hence became relatively infrequent in recent years.

Motion assumptions tend to simplify the dynamics of the body which in turn affects the complexity of inference algorithms. The first assumption is very common and is mostly done for convenience. If one knows that the subject is present and visible in all frames, then there is no need to waste resources detecting whether this is in fact true. The second assumption is also very general and basically assumes that in video the frame rate is sufficiently high to ensure that large jumps in the pose from frame to frame are impossible. The third assumption is an important simplification. Modeling occlusions (both self- or world-occlusions) is generally difficult. The reason for this is that it often requires per pixel reasoning and sophisticated models of the scene and pose. Assuming that there are no occlusions greatly simplifies the model. This assumption, however, is rarely satisfied in practice. The last two assumptions deal with the specific models of dynamics, that can significantly simplify the search for body pose. The simple movement assumption ensures that while the articulated pose of the body may be very complex and represented by a high dimensional state-space, the incremental search for the pose in the image sequence would involve searching over only a small sub-set of parameters at any given time. The assumption of the known movement is usually useful in the context of

coordinated or cyclic motions. For example, walking can be represented using two parameters that control the speed and phase of the gait cycle. In this case knowing that the person is walking, can reduce the search from the 30+ degrees of freedom to only 2 abstract degrees of freedom defining the motion in the “walking” sub-space (often referred to as latent space).

2.2 Humans at Different Scales

It is useful to think of the articulated object inference at different scales. Intuitively, the scale of the object in an image is an important clue as to how much can be inferred about the object. For example, while it would be great to infer the full articulated pose of a person that occupies only a few (*e.g.* 10) pixels in an image, it is clearly an impossible task. In particular, we can partition the human motion domain roughly into three scales: *coarse*, *medium* and *fine*.

At the *coarse scale*, the person occupies perhaps 10–50 pixels, as is often the case in the wide area surveillance. At this resolution the person is observed as simply a blob of color or a template [161, 235, 236]. At this scale the articulations are insignificant, since little of individual body parts can be observed. We can, however, reason about the presence/absence of the person and/or the coarse motion of the person in the environment.

At the *medium scale*, the person occupies perhaps 50–100 pixels, the upper and lower body segments become visible and some coarse articulations can be observed [162, 163, 259]. For example, in sporting videos at this scale one can tell when the baseball player hits the ball, or in which phase of the run cycle the player is in.

At the *fine scale*, the person occupies > 100 pixels, individual body parts (*e.g.* legs, arms and feet) can be reliably observed, and one expects to be able to infer the fine articulation of the body [1, 30, 34, 36, 52, 59, 74, 78, 93, 98, 111, 112, 113, 122, 126, 127, 131, 169, 173, 189, 193, 195, 196, 197, 205, 206, 209, 226, 228, 237]. By fine articulation here, we mean articulation of all joints in the human body. In this thesis we mostly address problems of this type.

2.3 Categorization of Approaches

For convenience this section presents a concise categorization and comparison of prior approaches to articulated human motion and pose estimation, encoded in Table 2.2. We pay particular attention to the more recent methods along with methods that are directly or indirectly related to this thesis. We classify approaches according to the three main categories: (1) choice of a model, (2) methods used for inference and (3) features used for observations. In each category we introduce sub-categories to further characterize approaches proposed. The approaches listed are then discussed in further detail in the context of the introduced categories in Sections 2.4 – 2.10.

Table 2.2: **Categorization and comparison of articulated human pose and motion estimation approaches.** Comparison and classification of various human motion approaches from the literature. Emphasis is given to the more recent works. A more comprehensive survey can be found in [64]. Approaches presented in this thesis are listed in bold. In “Method”, **P** and **T** correspond to Pose Estimation and Tracking respectively; **D** and **G** correspond to Generative and Discriminative approach respectively.

Year	Reference First Author	Model			Method			Observations		Motion
		Shape	Parts	Dim	Optimization	Use	Type	Cam	Cues	
2007	This work	R-Elliptical cones	10/15	2D/3D	Particle Message Passing (NBP)	P,T	D,G	1-7	edge + silhouette	N/A
2007	Balan [13]	SCAPE	N/A	3D	Iterated Importance Sampling	T	G	3-4	silhouette	Gaussian
2007	Mundermann [150]	SCAPE	N/A	3D	Articulated ICP	T	G	5-8	voxels	smooth
2007	Srinivasan [213]	Exemplars	6	2D	DP-like Parsing with Pruning	P	D,G	1	shape	-
2006	Agarwal [1]	Mesh / POSER	N/A	3D	Relevance Vector Regression	P	D	1	silhouette	N/A
2006	Gall [69]	Mesh	N/A	3D	Annealed Particle Filter	T	G	4	contour	smooth
2006	Han [78]	Polygonal patches	5/9	2D	Non-parametric BP (NBP)	T	G	1	edge + image template	N/A
2006	Lee [126]	Truncated cones	6/9	3D	Belief Propagation + MCMC	P,T	G	1	face + color + region + silhouette + skin	
2006	Li [131]	R-Elliptical cones	10	3D	MHT in Latent Space	T	G	1	silhouette	Gaussian
2006	Rodgers [177]	Mesh for each body part	16	3D	Loopy Belief Propagation + ICP	P	D,G	-	range scan data features	N/A
2006	Rosenhahn [183]	Free-form surface patches	N/A	3D	Iterative Closest Point (ICP)	T	G	3-4	contour	smooth
2006	Sigal [195]	R-Elliptical cones	10	3D	PAMPAS/NBP + BME	P,T	D,G	1	edge + silhouette	N/A
2006	Sigal [196]	Trapezoids	10	2.5D	PAMPAS/NBP	P	D,G	1	color + edge + silhouette	N/A
2006	Sminchisescu [205]	Mesh / POSER	N/A	3D	Variational EM	P	D,G	1	SIFT descriptors	N/A
2006	Wang [241]	SPM + templates	10	2D	Optimized Unscented PF	T	G	1	edge + template	smooth
2006	Urtasun [226]	Stick-figure	15	3D	Deterministic optimization	T	G	1	WSL image-based tracks	smooth in LDLS
2006	Zhang [257]	Line segments / contour	12	2D	Hybrid search: DP + SMC	P	D,G	1	edge + skin/hair color	N/A
2005	Felzenszwalb [59]	Rectangles	10	2D	Belief Propagation (BP)	P	G	1	silhouette	N/A
2005	Hua [93]	Quadrangulars	10	2D	Data-driven BP	P	G	1	face + lines + skin color	N/A
2005	Kehl [113]	Mesh / POSER	N/A	3D	Stochastic Meta Descent	T	G	4/11	3D voxel + color	smooth
2005	Lan [122]	Rectangles	10	2D	Belief Propagation (BP)	P	G	1	silhouette	walk
2005	Ramanan [169]	Rectangles	10	2D	Belief Propagation (BP)	P	G	1	color + edge	N/A
2005	Ren [174]	Pair of parallel lines	9	2D	Integer Quadratic Programming	P	D	1	edge + parallelism	-
2005	Sminchisescu [206]	Mesh / POSER	N/A	3D	BME using EM	P,T	D	1	shape context	smooth
2004	Elgammal [55]	3D Joints	16	3D	LLE + GRBF			1	silhouette	walk
2004	Lee [127]	Truncated cones	15	3D	MCMC sampling	P	G	1	edge + face + ridge + skin color	N/A
2004	Mori [147]	Patches	9	2D	Normalized Cuts + Assembly	P	D	1	color + edge + focus + shading + shape	N/A
2004	Sigal [197]	R-Elliptical cones	10	3D	PAMPAS / (NBP)	P,T	D,G	4	edge + silhouette	Gaussian

Continued on Next Page...

Table 2.2 – Continued

Year	Reference First Author	Model			Method			Observations		Motion
		Shape	Parts	Dim	Optimization	Use	Type	Cam	Cues	
2004	Urtasun [228]	Ellipsoids	14	3D	Least-squares	T	G	3	stereo	walk, run
2003	Cheung [36]	Voxels	9	3D	Expectation Maximization	P,T	-	8	color + silhouette	smooth
2003	Grauman [77]	3D Joints	19	3D	Probabilistic PCA	P	-	1-4	silhouette	N/A
2003	Shakhnarovich [189]	Mesh / POSER	N/A	3D	K- Nearest Neighbors	P	D	1	multi-scale edge direction hist.	N/A
2003	Ramanan [173]	Rectangle	9	2D	Max-product BP	P,T	-	1	color + edge	N/A
2003	Sminchisescu [209]	Superquadrics	15	3D	Covariance Scaled Sampling	T	G	1	edge + intensity + motion	N/A
2003	Wu [248]	Quadrangular	2/3/6/10	2D	Mean Field Monte Carlo	T	G	1	edge + intensity	const. accel.
2002	Mori [148]	Exemplars	N/A	3D	Shape context + geometry	P	G	1	edge	N/A
2002	Rosales [179]	Markers	20	2D	SMA using EM	P	D	1	silhouette Hu-moments	N/A
2001	Ioffe [96]	Rectangles	9	2D	Viterbi	P,T	G	1	image template	smooth
2001	Plankers [165]	Deformable meta-balls	-	3D	Least-squares	T	G	3	silhouette + stereo	-
2000	Deutscher [52]	Cones	15	3D	Annealed Particle Filter	T	G	3	edge + silhouette	Gaussian
2000	Howe [90]	Patches	14	3D	Gradient descent + EM	T	G	1	image template	N/A
2000	Hu [91]	Rectangles	10	2D	Genetic algorithm	P	G	1	silhouette	const. accel.
2000	Rosales [181]	markers/cylinders	11	3D??	Levenberg-Marquard	P(T?)	D	1	silhouette Hu-moments	N/A
2000	Sidenbladh [193]	Cylinders + spheres	9 + 3	3D	Particle Filter	T	G	1	image intensity	smooth / walk
2000	Taylor [222]	Stick-figure	9	3D	Geometry	P	-	1	marked 2D points	-
1999	Cham [34]	SPM + templates	10	2.5D	MHT	T	G	1	image template	linear
1999	Ioffe [98]	Rectangles	9	2D	MAP by sampling	P	G	1	limb symmetry	N/A
1999	Pavolvić [164]	Templates	6	2D	Viterbi	T	G	1	image template	switching LDS
1999	Wachter [237]	R-Elliptical Cones	15	3D	Iterated extended kalman filter	T	G	1	edge + image template + flow	linear
1998	Bregler [30]	Ellipsoids	10	3D	Least-squares	T	G	3	spatio-temporal image gradient	N/A
1998	Moris [149]	Patches	10	2D	Gradient descent	T	G	1	SSD template	smooth
1996	Gavrila [74]	Tapered superquadrics	12	3D	Best-first search	T	G	4	edge	linear
1996	Ju [111]	Templates	2	2D	Gradient descent	T	G	1	flow	linear
1996	Kakadiaris [112]	Deformable 3D model	2	3D	Kalman Filter	T	G	3	edge	linear

PAMPAS – Particle Message Passing (a variant of non-parametric belief propagation)

BME – Bayesian Mixture of Experts

MHT – Multiple Hypothesis Tracking

EM – Expectation Maximization

SMA – Specialized Mappings Architecture

LDS – Lower Dimensional Space

LDLS – Lower Dimensional Latent Space

SCAPE – Shape Completion and Animation of PEople

2.4 Representing the Body

In order to infer the pose of the body, one first needs to choose a representation for the body. In order to model the body, in general, one needs to represent (1) the articulated skeletal structure and (2) the shape or “flesh” (representing the human tissue and perhaps clothing) that is draped over the skeleton. A particular choice of skeletal structure gives rise to, often non-unique, parameterization of articulations that one would like to infer. Since the human body is complex, a realistic articulated model can have anywhere from 30–60 parameters. The choice of flesh often dictates the features that should be used to match the model to the images. In most vision approaches it is assumed that the flesh is rigidly attached to the skeletal structure and is independent of articulation. Recently, however, more realistic representations that explicitly model correlations between the skeletal structure and the shape have been introduced [6, 8]. These models are able to model such phenomena as bulging of muscles based on the articulation of the body. They also provide basis for much richer set of realistic human shapes. These models have originally been developed in the graphics community for synthesis, and are slowly making their way to pose estimation and tracking applications [13, 150].

A large variety of 2D [59, 78, 93, 98, 111, 122, 169, 173], 2.5D [34, 196] and 3D [1, 30, 36, 52, 74, 112, 113, 126, 127, 131, 189, 193, 195, 197, 205, 206, 209, 226, 228, 237] human models have been proposed in the literature.

For surveillance purposes, simple template based [161, 235] and 2D image blob models [114, 162, 163, 259] have proved effective. Planar articulated representations [111] have also been used for articulated pose estimation and tracking in monocular imagery. These models are effective in recovering the pose of the person, in the cases where the motion is either lateral or frontal to the image plane. In such cases, the foreshortening that is due to out-of-plane rotations is typically insignificant. To handle foreshortening and depth variations 2.5D models have been introduced [34]. In addition to planar articulations these models allow scaling that can account for the foreshortening of limbs in the image. However, these approaches recover the pose and model the constraints imposed by the body in 2D. As a result, some constraints that are straightforward to express in 3D are difficult to encode (*e.g.* interpenetration) in 2D.

Models that are formulated directly in 3D are usually more straightforward, but often are ill-constrained, especially in the monocular case. In such cases, multiple views [36, 52, 74, 112, 113, 193, 197], stereo [228] or strong prior motion models [55, 125, 226] are often needed to regularize the pose recovery. For simple reasoning about a person’s location in space, without reasoning about the pose or articulation, simple 3D occupancy representations are sufficient [18, 100] that use simple geometric structures like boxes [18] or generalized cylinders [100] to model a body as a whole. For human-computer interactions where one needs to reason about the articulations in a constrained environment simple 3D blob models have proved effective [247]. In applications where many cameras are available voxel representation has been used either directly [36, 113] or as an intermediate representation for more parametric body models [215]. Most approaches, however, model the body using a 3D kinematic skeletal structure with associated 3D volumetric parts [14, 52, 193, 209]. The most common models that are relevant to the work presented here will be introduced in the next sections.

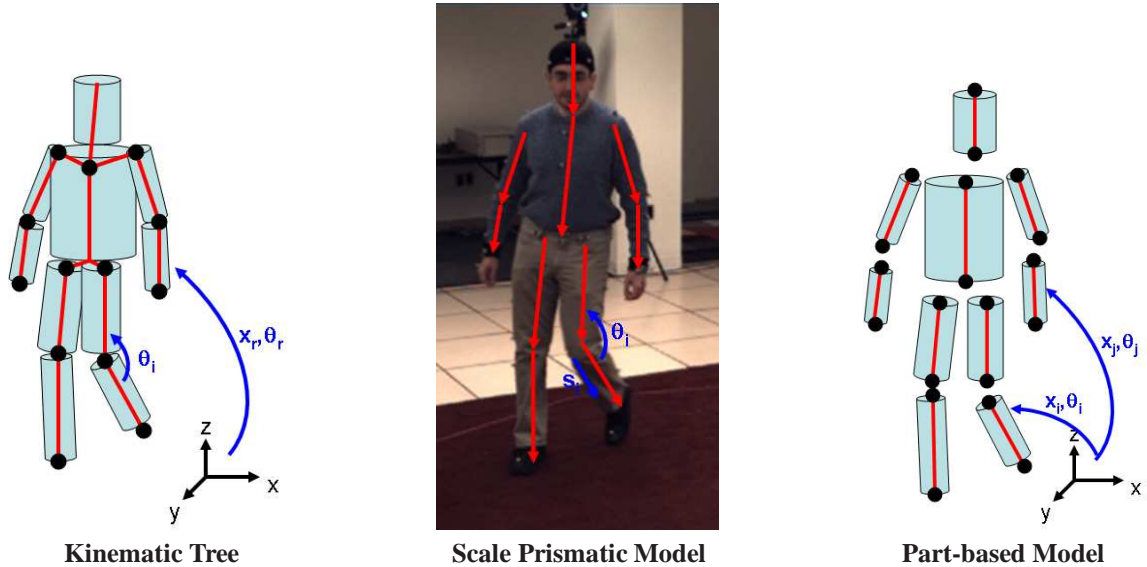


Figure 2.1: **Representing the body.** Three different representations of the body are shown. The 3D kinematic tree model, shown on the left, consists of the tree-structured skeleton (in red) that can be parameterized using a set of joint angles, θ_i , and position and orientation for the root of the tree in global coordinate frame \mathbf{x}_r, θ_r . The skeleton is dressed by cylindrical “flesh”. In the middle, a Scale Prismatic Model (SPM) is shown. Red arrows in SPM designating the segments of the tree structured 2D model. SPM is parameterized in the image plane using 1D planar joint rotations, θ_i , and scalings along limb axes, s_i . Lastly, on the right, a 3D part-based body model is shown, consisting of a set of disconnected cylindrical parts. The part-based model is parameterized by position and orientation of individual body parts in world coordinate space, $\mathbf{x}_i \in \mathbb{R}^3$, $\theta_i \in SO(3)$. This redundant representation while non-intuitive, leads to tractable inference algorithms.

2.4.1 Kinematic Tree

Kinematic trees are by far the most predominant 3D representation of the human body in literature. Kinematic tree based representation assumes skeletal tree structure and represents the pose using a set of parameters $\mathbf{X} = [\mathbf{x}_r, \theta_r, \theta_1, \dots, \theta_N]$, where $\mathbf{x}_r \in \mathbb{R}^3$ is the position of the root of the tree; $\theta_r \in SO(3)$ is the orientation of the root in the world coordinate frame, and $\theta_i, i \in [1, \dots, N]$ is the orientation of the limb i in the parent’s coordinate frame (*a.k.a.* joint angles). It is worth mentioning that the dimensionality of the θ_i would generally depend on the type of the joint. Hinge joints often associated with knees are most often modeled using $\theta_i \in \mathbb{R}^1$, whereas ball-and-socket joints that are often associated with shoulders are more often modeled by $\theta_i \in \mathbb{R}^3$. Hence the final dimensionality of the articulated pose \mathbf{X} will depend on the number of joints being modeled and the joint types assumed. In all cases, this would lead to a high dimensional but not-redundant representation of the pose (typically $\mathbf{X} \in \mathbb{R}^{30+}$).

The basic kinematic tree model leaves open the issue of how the joint angles are parameterized and how the shape of individual limbs are modeled. For the former a number of representations have been employed, including Euler Angles [14], Quaternions [197, 219], and exponential maps [30]. All of these suffer from problems and inconveniences and the choice depends in part on the type of optimization employed by the approach.

Euler angles represent a general 3D rotation, θ_i , by concatenating rotations about the orthogonal coordinate axes R_x, R_y and R_z . The order in which these rotations are concatenated must be specified beforehand.

While this parameterization is intuitive and is often used in biomechanics to study the behavior of joints, it does suffer from what is called the ‘gimbal lock’ problem. When two out of the three rotation axes align, a rotation is lost. As a result small changes in an angle can cause large deviation in the orientation of the limb. This makes this representation very unstable for the use in inverse kinematics and/or direct optimization. For similar reasons interpolation in the Euler angle space is poorly defined.

Quaternions [191] do not suffer from these problems. However, quaternions are represented as 4D unit vectors lying on a 3D spherical shell embedded in 4D space, this makes modeling distributions over quaternions challenging. Using quaternions, a general rotation of angle θ about axis $\vec{v} \in \mathbb{R}^3$ can be written as $\mathbf{q} = [q_x, q_y, q_z, q_w]^T = [\vec{v} \cdot \sin(0.5\theta), \cos(0.5\theta)]$. As mentioned, however, valid rotations must maintain that $\|\mathbf{q}\| = 1$. Since there are four dimensions in which quaternion can change but only 3 degrees of freedom, typical optimization approaches can move away from the unit sphere representing valid rotations. In such cases the solution is often to re-normalize the quaternion, which works well [197, 219] so long as the distance from the sphere is small. Quaternions are also well suited for interpolation and are often preferred for that reason.

Exponential maps [71] is yet another way to model 3D rotations. The idea behind exponential maps is to try to maintain the benefits of quaternions, while getting a representation $\in \mathbb{R}^3$. There are many variants of exponential maps in the literature, however, the basic idea is to model an arbitrary rotation in 3D using a vector $\vec{v} \in \mathbb{R}^3$ such that $\vec{v}/|\vec{v}|$ gives the axis of rotation and $|\vec{v}|$ the amount of rotation. A particular variant that made this representation popular, in the context of articulated human motion, is twists and exponential maps formulation of Bregler and Malik [30].

The shape of the person is often modeled using relatively simple 3D geometric representations including cylinders [180, 193], tapered cones [52, 237], ellipsoids [30], and superquadrics [74, 210, 211]. More complex models such as metaballs [165], mesh-based surface models [215] or free-form surface patches [183] are also gaining popularity.

2.4.2 Scale Prismatic Model

It is clear that 3D models such as the one described in the previous section cannot be fully observed from a single monocular image, and hence either temporal or prior knowledge is often required to address ambiguities that arise. In such cases it sometimes may suffice to estimate the pose of the body in the image plane (not the world) directly. To this end Scale Prismatic Model (SPM) is derived [149] by “projecting” the 3D kinematic tree model into the image and parameterizing the resulting planar skeleton (stick figure) with as few parameters as possible. Each 3D joint of kinematic tree in the SPM is replaced by a 2D joint (planar rotation) with a rotation axis perpendicular to the image plane and scale that accounts for foreshortening effects due to perspective and out-of-plane rotations. The resulting SPM model often leads to a more compact representation of the state than a full 3D kinematic tree model. SPM is a 2.5D generalization of “cardboard” models [111] that have been widely used for 2D articulated tracking.

The planar skeletal structure of SPM also requires a 2D representation for the limbs. In the literature, rectangles [91], rectangles with rounded corners [132], 2D templates [149], and affine flow patches [111] have all been used for this purpose.

2.4.3 Part-based Representation

While it is natural to represent the body using tree-structured kinematic models such as kinematic trees or scaled prismatic models, these models tend to require a very high dimensional parameterization where parameters are correlated in complex ways. This leads to a disadvantage that these models are often computationally expensive (and often intractable) to deal with, especially for the pose estimation task. To address this, it has proven useful to represent the body using a set of disaggregated parts that interact via a set of pair-wise constraints that attempt to enforce the body consistency. As a result, the body is represented using a redundant representation in a global space. This representation that leads to an even higher dimensional parameterization of pose (due to redundancy), decouples many of the parameters, making the search tractable.

The use of disaggregated models for finding or tracking articulated objects date back at least to Fischler and Elschlager’s pictorial structures [62]. Variations on this type of model have been more recently applied to general object detection [33, 44, 198], and articulated pose estimation for people [59, 78, 93, 96, 97, 98, 122, 169, 173, 174, 177, 196, 197, 248], animals [170, 172] and hands [219].

Articulated disaggregated models, model the body using simple 2D (*e.g.* rectangles [59, 122, 169], trapezoids [195, 196], quadrangulars [93, 248], polygonal patches [78], or templates [176]) or 3D parts (*e.g.* right-elliptical cones [197, 199], truncated quadrics [219], or surface models [177]) and set of constraints between parts that are encoded either directly in terms of compatibility [96, 97, 98, 172, 174] or probabilistically [59, 78, 93, 122, 169, 196, 197, 248]. Most probabilistic models [59, 122, 169, 174] rely on the underlying tree-structure of the model for tractable inference and hence are only capable of modeling kinematic constraints. In this thesis (and in [195, 196, 197, 198]) we introduce the means of formulating and inferring the pose using a more diverse set of models that can model any pair-wise relationships between parts statistically. Kinematic, occlusion and inter-penetration constraints can all be modeled. Recently, similar method has been introduced for determining the articulated pose of people from range scan data by Rodgers *et al.* [177].

The pictorial structures approach of Felzenszwalb and Huttenlocher [59] is one of the more influential 2D disaggregated models introduced for articulated pose estimation. The approach models the body parts using rectangles and the kinematic and joint constraints between parts using Gaussian distributions. The model assumes that the state of each limb can be discretized and the inference proceeds to find the globally optimal pose using dynamic programming (that can in this case be interpreted as Belief Propagation, see Section 3.5.2). This basic model has been successfully extended by introducing richer likelihood functions [178] or simple dynamics [122]. More recently, the pictorial structures approach has been extended [169] to elegantly estimate the appearance models of parts jointly with the pose in extended image sequences.

A similar approach to ours has been adopted in [248] for tracking a 2D human silhouette using a dynamic Markov network and later in [93] using data-driven Belief Propagation. A much simplified observation model was adopted in [248] and their system does not perform automatic initialization. In [93] a much richer observation model was used, but the approach is still limited to 2D pose inference in roughly frontal body orientations; the subject is assumed to be facing towards the camera and wearing distinct clothes. The [93, 248] and the method proposed in this thesis use somewhat different inference algorithms and a comparison between these methods merits future research.

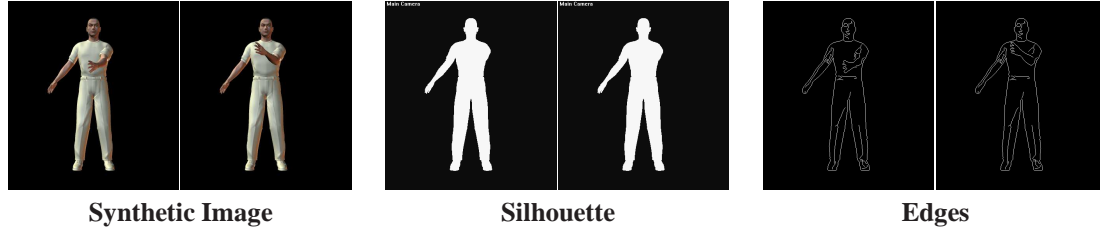


Figure 2.2: **Common features used for articulated pose and motion estimation.** On the left two synthetic images of a person in different postures are shown; corresponding silhouette features are illustrated in the middle. Notice that silhouettes in this case are identical and cannot distinguish between the two poses on the left. Edge features, illustrated on the right, however, clearly contain information required to distinguish the two poses.

2.5 Image Features

The choice of the model for the body often gives rise to the features that are chosen to match the model to the image. A typical inference approach proceeds to extract appropriate features from the image and then either matches these features to features hypothesized by the model, or infers the state of the model directly from them. While ranking of features that are most useful for the task of articulated pose estimation and tracking is still debated (relative importance of some feature choices was tested in [14]), it is a general consensus in the community that combining features leads to better and more robust inference methods. We will describe some of the more common features used in the literature below.

2.5.1 Silhouettes

Silhouettes are among the features most frequently used for two main reasons, **(1)** they can localize the search for the body by excluding large portions of the image and **(2)** they provide features that are easy to compute. Silhouettes, being region based, are also inherently more robust to clutter than edges or contours. However, silhouettes are also ambiguous.

Background subtraction is the process often used to obtain the silhouettes. If the scene background is specifically designed for such a task, as in Chroma-keying [133] where the background is usually blue or green and is distinct from the clothes worn by the subject, then simple thresholding can be used. An alternative is to have an arbitrary background but a subject wearing a distinct colored suit [142, 156]. In both cases very precise silhouettes can be obtained. In more realistic scenarios where the background and foreground are more general, a statistical representation of the background can be learned either on-line or off-line. Often a per pixel model of the background is learned and used to classify the image pixels into foreground and background classes. A sequence of images can be used to learn a per pixel Gaussian (mean and covariance) and Mahalanobis distance can be used to classify the pixels [247] $I_{silhouette}(x, y) = [(\mu_{x,y} - I_{x,y})\Sigma_{x,y}^{-1}(\mu_{x,y} - I_{x,y})^T > \tau]$, where $\mu_{x,y}$ and $\Sigma_{x,y}$ are mean and covariance of the Gaussian distribution for background pixel (x, y) learned from training data and τ is the threshold. In the presence of background motion (e.g. leaves waving in the wind) this approach will lead to poor results. For such scenarios Gaussian mixture models [216] have been proposed to model multi-modality of background pixel observations. More sophisticated models that use either color thresholding [86] or pixel gradients [140] to remove shadows have also been proposed.

Since recovered silhouettes are often noisy, due to pixel variations, morphology is commonly used to ‘clean up’ the silhouette image. Alternatively approaches that embed spatial consistency directly into the background subtraction process have also been proposed [89]. In cases where lighting variations are common (as in prolonged surveillance applications) background models are often updated on-line to account for global variations in lighting.

Even with the most sophisticated background subtraction methods often good subtraction is hard to obtain. In particular, often the background colors appear in the foreground (or vice versa), the shadows are hard to discount completely, and motion in the background creates challenges. Furthermore if multiple foreground objects are present the assignment of foreground region(s) to objects must be addressed.

2.5.2 Color

Background subtraction at best is only capable of separating the foreground object (person) from the background. There is no explicit assignment of silhouette pixels to model parts and inference methods must be employed to simultaneously solve for feature assignment (often not explicitly) and the pose. If color (or texture) of body parts is known, then assignment can be facilitated thus reducing the complexity of the overall inference significantly. However, color information for parts will generally differ from person to person and between clothing types, hence often this information is unavailable. Methods have, however, been proposed that attempt to build the color appearance models for parts automatically either by clustering of coherent spatio-temporal regions [172, 173] or by roughly estimating the pose first using a generic model, learning the appearance, and then re-estimating the pose based on the learned image and person specific model [169].

A by far more common assumption is that some parts of the body are not covered by clothes [93, 126, 127], in which case skin color can be used as the signature for these parts. Skin-color detection and segmentation has a long standing history in computer vision. Jones and Rehg [106] introduced a relatively simple parametric probabilistic model for classifying skin pixels. The key step in proposed skin pixel classification is the computation of $p(\text{skin}|I_{x,y})$ for a given pixel value $I_{x,y}$ at location (x, y) in an image, which is given by Bayes rule:

$$p(\text{skin}|I_{x,y}) = \frac{p(I_{x,y}|\text{skin})p(\text{skin})}{p(I_{x,y}|\text{skin})p(\text{skin}) + p(I_{x,y}|\neg\text{skin})p(\neg\text{skin})}. \quad (2.1)$$

A simple threshold $0 < \tau_{sk} < 1$, $p(\text{skin}|I_{x,y}) > \tau_{sk}$, can then be used to classify a given pixel in an image as belonging to a skin class. The exact value τ_{sk} can be derived based on the risk of misclassification. In this model both $p(I_{x,y}|\text{skin})$ and $p(I_{x,y}|\neg\text{skin})$ are modeled using 16 component Gaussian mixtures learned from 13, 640 hand labeled images; prior is also learned, $p(\text{skin}) = 1 - p(\neg\text{skin}) = 0.1$. The learned values for means and covariances are listed in [106].

2.5.3 Edges

Edge features, that are based on first spatial derivatives of the image, are often used due to their partial invariance to viewpoint, lighting and local contrast. Edges are also capable of discriminating variations in pose that may be ambiguous by looking at silhouettes (*e.g.* they can account for internal structure); this is illustrated in Figure 2.2. Edges, however, are still limited in that they are insensitive to the rotations of the limbs around their axis of symmetry (*i.e.* twists are unobservable).

Edges are also very local, comparing a model edge to an image edge that is one pixel away will lead to nearly zero response. To remedy this edges are often smeared by applying a Gaussian filter over the edge image [14, 52], as a result the extent of the edge is augmented. Furthermore distance transforms are often used to define a more global feature space [11] based on edges.

2.5.4 Contours

Contours refer to the edge representation of the object's full or partial outline. Contours can either be static or dynamic, where later is often referred to as *active contour*. Static contours can be thought of as a more compact representation of the silhouette, and inherently carry no more information than silhouettes themselves. Some invariance to geometry and appearance of the object can be obtained by using contour (or boundary) fragments that are geometrically constrained [88, 158]. A more general formulation of active contours allows the contour of the object (either whole or partial) to deform according to the image edges. The deformations are often controlled by energy functions that consist of two terms: one that attempts to minimize the distance between the contour and the edges in the image, and the other that controls the overall smoothness of the curve. The assumption being that in general we want to fit a relatively smooth contour to the image data. The deformations of the contour can also embed prior knowledge about allowed deformations for a given object class, in such cases the model is often referred to as deformable template. Deformable templates have been successfully applied for pedestrian detection [72] and localization. In general, deformable template models have lost popularity in recent years. This is partially due to the inherent ambiguities that exist in the contour representation, and partially due to the fact that while deformations can account to some extent for articulations they are not well suited for recovering those articulations. Contour ambiguities can be circumvented to some extent by considering contours from multiple calibrated views. Contours from 4 calibrated views have been used to reliably infer the articulated motion in [182, 184]. More recently, deformable templates have been used to localize parts of the body (e.g. the head-shoulder outline [126, 127]) as part of a more sophisticated hierarchical representation of the human body.

2.5.5 Ridges

Ridges refer to the second spatial derivatives of the image at a given scale. Since ridge (or second derivative) filters account for elongated spatial structure in the image, it has been shown that they are effective in modeling the limbs [192] if applied at a particular scale and orientation that is a function of limb width and pose. Intuitively ridge features encode the parallel edge structure of limbs in the body. As with any higher derivative filters ridge filters tend to be noisier than edges alone. They are also highly dependent on the orientation, configuration, and scale of the person in the image.

2.5.6 Image Flow

Image Flow refers to dense motion information that often is obtained using *optical flow* algorithms. Image flow (or optical flow) can be thought of as a vector field in an image that for every pixels defines where that pixel will move to in the next frame. In general to compute optical flow one must assume that the intensity

of the image pixels remains constant over time¹ (*i.e.* no lighting variations) resulting in the following optical flow constraint:

$$\frac{dI(x, y, t)}{dt} = \frac{\partial I(x, y, t)}{\partial x} \frac{dx}{dt} + \frac{\partial I(x, y, t)}{\partial y} \frac{dy}{dt} + \frac{\partial I(x, y, t)}{\partial t} = 0 \quad (2.2)$$

on the spatio-temporal image signal $I(x, y, t)$. Solving the equation for, $[\frac{dx}{dt}, \frac{dy}{dt}]$, results in an optical flow field (for details see [23]). However, assuming that the motion is independent at every pixel, results in an underconstrained set of equations. The common solution to this is to assume that the motion is constant over a neighborhood (*a.k.a.* spatial coherence assumption). A relatively large neighborhood is needed to constrain the solution and add some invariance to image noise. Unfortunately in relatively large regions the assumption of constant motion is likely to be violated due to the differences in depth, lighting and transparencies. As a result, a size of the region that is reasonable must be picked to leverage the two artefacts. Even so, it is often observed that foreground motion ‘smears’ onto background, or vice versa, especially in relatively constant image regions. One alternative that has been proposed to alleviate this, is to assume that there are a known and relatively small number of consistent motion fields (layers) that make up the overall motion [120, 251]. In general robust and accurate image flow is challenging and computationally expensive to compute and hence it is less frequently used for human motion and pose estimation. It is worth mentioning that a simple sparse temporal motion derivatives computed using frame differencing have been shown to be useful features as well [235] for general template based object detection.

Optical flow from multiple camera views can be combined to compute *scene flow* [232]. Just as optical flow is the two-dimensional motion of points in an image, scene flow is the three-dimensional motion of points in the world. Scene flow has been successfully used for temporal and view interpolation of human motion in [231].

2.5.7 Voxels

Just as scene flow is a three-dimensional variant of two-dimensional optical flow, voxel representation is a three-dimensional alternative to two-dimensional silhouette. Voxel data can be acquired using silhouettes from multiple cameras [28, 36, 37, 39, 113]. Typically the volume of interest, corresponding to the visible portion of the scene, is divided into $N \times N \times N$ equal sized² voxels [37]. Each voxel is tested if it belongs to foreground object or background by checking if its projection falls onto silhouette in each camera view. Voxels that in all camera views project onto the silhouette are labeled as belonging to the object. Voxels that at least in one of the camera view fall outside the silhouette are automatically labeled as belonging to background. More recent approaches of Voxel Coloring [36] in addition to silhouette consistency also check for color consistency across multiple views. Voxel features while powerful, in that they directly provide 3D information, are very sensitive to noise in silhouettes. To handle this a probabilistic version of voxel based representation, called *probabilistic occupancy grid*, was introduced in [65]. Instead of assigning each voxel a binary label, in [65], each voxel is assigned a probability of belonging to the foreground object. For further discussion of voxel based methods see Section 2.9.1.

¹Brightness constancy has also been used to formulate likelihood functions directly [192], without explicitly computing the optical flow.

²Alternatively, adaptive partitioning is also possible.

2.5.8 Image Descriptors

Image descriptors are generic features that have been used in the literature for articulated pose estimation, image retrieval, image categorization, generic object recognition, and many other applications. There are many descriptors that exist, that exhibit variety of properties and invariances to geometric transformations. In this section we will cover the most common descriptors from the recent literature.

Haar Wavelets

A set of image based filters are constructed that correspond to oriented derivative filters of various sizes and scales. The response of these filters are considered an over-complete feature representation of the image [144] (or an image patch). In general only a subset of these features is used to represent an object or an image. For example, AdaBoost [236], automatically constructs an object detector by selecting a sub-set of features most useful for classification of a given object class. Haar wavelets and AdaBoost will be discussed in depth in Chapter 4.

SIFT descriptor

Scale-invariant feature transform (SIFT) [135] represents scale-normalized image region (obtained using standard interest point operators³) with the concatenation of gradient orientation histograms relative to several rectangular sub-regions. Image gradient direction and magnitude are computed for every pixel in the region. Histograms of gradient orientation, weighted by gradient magnitude are then computed for a given set of non-overlapping sub-regions. Orientations in the sub-region are normalized with respect to the orientation of the center pixel of sub-region and are histogrammed into Θ bins. The SIFT local descriptor is the concatenation of these gradient orientation histograms for all sub-regions. For convenience, often the scale-normalized image region is broken into 16 sub-regions and 8 orientation histogram bins for each region, resulting in the overall descriptor of size 128. More recently [46] it has been shown that better performance (in the visual categorization task) can be achieved by histogramming not the gradients themselves, but the projections of gradient images onto a set of basis functions learned from training data using PCA.

Shape Context

Shape context [17] is an alternative to the SIFT descriptor, that only works with binary edges (obtained by thresholding the magnitude of the derivatives). Also, instead of histogramming the edges into a regularly spaced grid, a log-polar grid is used. The effect of this is that shape context is much more sensitive to local variations in shape than more global variations. Scale and orientation can be normalized much like in the SIFT case to achieve orientation and scale invariance. Typically the shape context is computed for a set of points equidistantly sampled on the contour of the desired object. The set of histograms corresponding to these points are then considered as the descriptor for the object or the region. The typical setting is to compute the shape context for about 100 points on the contour and have 5 polar and 12 orientation bins. At this resolution the final descriptor is a 6,000 dimensional vector. Since working with a vector of this size is hard, typically

³In the original SIFT [135] formulation a difference of Gaussians (DoG) approach [135] is used for keypoint selection which is an approximation for the Laplacian operator [134]. Alternatively, other approaches like determinant of the Hessian (DoH) [16] can be adopted for the same task. Additional interest point operators frequently used in the literature are Harris [79] and Shi-Tomasi [190] corner detectors.

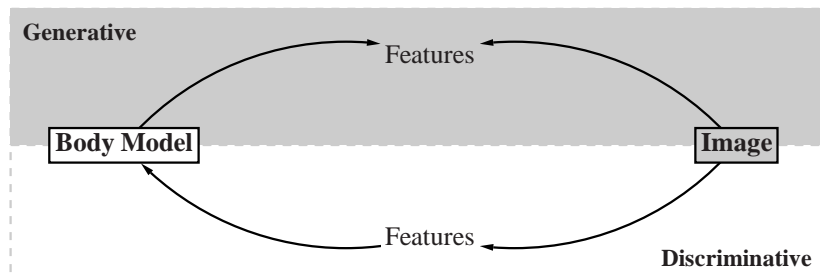


Figure 2.3: **Generative and discriminative pose estimation and tracking.** Illustration of the generative and discriminative pose estimation and tracking frameworks. In a generative framework, the model is used to predict a set of features observed in an image. The model parameters, that typically represent articulated pose, are then optimized such that predicted features match the features observed. In a discriminative framework, observed image features are used directly to infer the model parameters.

the shape context space is quantized and secondary voting is used to get a much lower dimensional histogram representation of the object [1, 4, 166] (typically 100D), which as a consequence loses much of the spatial information encoded in the original 6,000 dimensional vector.

2.6 Pose Estimation and Tracking

We found it useful for the purposes of taxonomy to classify approaches to human motion into pose estimation, tracking, or both. While these two notions are slowly converging, there are still useful distinctions between the approaches that deal with each of these problems. For the purposes of this thesis we define *pose estimation* as estimation of an articulated pose from single monocular or multiocular images and *tracking* as estimating of the articulated pose in an image sequence, where the pose estimate for the first frame is known. Approaches that automatically estimate the articulated pose in the first frame and then track it over an extended image sequence are classified as both *pose estimation* and *tracking* using the above definitions.

It is easy to see that *pose estimation* is an inherently more general and challenging problem, since it assumes nothing about the placement or configuration of the body. If we can estimate the pose of the body from a single image robustly we can also solve tracking by simply performing pose estimation at every frame. Given the state of the art in pose estimation, however, this is still impractical in general. This solution to tracking is also extremely inefficient since it ignores important temporal correlations between poses. To this end, the major contribution of *tracking* approaches is the use of temporal consistency and motion models to efficiently search for the pose at future frames given the current estimate for the position and configuration of the body.

2.7 Discriminative and Generative Methods

Discriminative approaches attempt to learn a direct mapping from image features to 3D pose from either a single image [1, 179, 181, 189, 206] or multiple approximately calibrated views [77]. These approaches tend to use silhouettes [1, 77, 179, 181] and sometimes edges [205, 206] as image features and learn a probabilistic mapping in the form of Nearest Neighbor (NN) search [189], regression [1], Bayesian mixture of experts [206], or specialized mappings [179]. These methods tend to be fast and are often reliable so long as the statistics

of features at runtime were captured well by the training set. In general, however, these approaches have two drawbacks: (1) they tend to provide a black-box solution that gives little insight into the problem, and (2) the performance tends to degrade significantly in cluttered scenes where it is difficult or impossible to extract good features. Generative approaches tend to work better in such cases, since they model the image generation process explicitly.

Generative approaches to human tracking have a long history in vision. Most of these approaches rely on a kinematic tree [139] representation of the body in 2D [111], 2.5D [34], or 3D [30, 52, 193, 210]. In such approaches the pose is defined by a set of parameters representing the global position and orientation of the root, usually a torso, and the joint angles representing the state of each limb with respect to the neighboring part higher up in the tree. The inference in these models amounts to generating a number of hypothesis for the pose, and evaluating the likelihood that a given hypothesis gives rise to the image evidence. Inference in such models, however, often requires stochastic search for the parameters in a high dimensional, 25-50D, state-space. The high dimensionality of the resulting state-space has motivated the development of specialized stochastic search algorithms [52, 136, 193] that either exploit the highly redundant dynamics of typical human motions [193], or use partitioned sampling schemes to exploit the tree-structured nature of the model [136]. These schemes have been effective for tracking people wearing increasingly complex clothing in the increasingly complex cluttered backgrounds [210]. However, even with efficient inference algorithm, search in this high dimensional space without initialization that is close to the solution is computationally impractical. Hence, most of these methods require manual initialization and are hopelessly lost once the tracker fails. To handle these problems disaggregated generative models have been introduced. Further discussion of disaggregated models was given in Section 2.4.3. Some disaggregated models [93] (including the ones introduced in this thesis [196, 197]) could be thought of as spanning both discriminative and generative realm, since they include a discriminative stage to bootstrap the generative inference.

The discriminative and generative methods in the context of graphical models will further be discussed in Section 3.2.3. It is also worth mentioning that there are current and on-going efforts to combine discriminative and generative methods [205], that may lead to more robust solutions in the future.

2.8 Optimization Methods

Most human motion and pose estimation approaches propose some sort of optimization method, direct or probabilistic, to optimize the pose (and/or body model) subject to the image features observed. This section will give an non-exhaustive overview of the methods employed.

Direct optimization. Direct optimization methods [212, 228] often formulate a continuous objective function $F(\mathbf{X}_t, I_t)$, where \mathbf{X}_t is the pose of the body at time t and I_t is the corresponding observed image, and then optimize it using some standard optimization technique. Since $F(\mathbf{X}_t, I_t)$ is highly non-linear and non-convex there is almost never a guarantee that a global optimum can be reached. However, by iteratively linearizing $F(\mathbf{X}_t, I_t)$ and following the gradient with respect to the parameters a local optimum can be reached. If a good estimate from the previous time step is available, and the pose changes slowly over time, then initializing the search with the previous pose often leads to a reasonable solution.

Probabilistic inference. It is often convenient and natural to formulate tracking and pose estimation as probabilistic inference. A probabilistic framework has two advantages over the direct optimization methods: (1) it can encode the confidence of any given articulated interpretation of the image, and (2) and more importantly, it allows one to maintain multi-modal predictions both spatially and over time. Multi-modality arises naturally in human motion estimation, since the body in different postures can look very similar (if not identical) in the image. The number of valid interpretations of the image depend significantly on the features used, imaging conditions and the temporal history. By maintaining a multi-modal pose hypothesis over time, approaches can often benefit by resolving the ambiguities as more information becomes available.

Let us assume that the pose of the body, \mathbf{X}_t , at time t is generated by a dynamic process. In general, for articulated motion estimation we are interested in the joint posterior distribution $p(\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_t | I_0, I_1, \dots, I_t)$, where I_i is a (possibly multiocular) sequence of image observations over time $i \in [0, \dots, t]$. Since dealing with the joint distribution over many high-dimensional variables is hard approximations are often made that only infer the marginals of the joint (see further discussion of this in Section 3.5). The marginal equations are significantly simplified by introducing Markov assumption over the hidden states. The 1-st order Markov assumption⁴ states that pose, \mathbf{X}_t , at time t depends only on the pose at $t - 1$. This model is also known as Hidden Markov Model (HMM) and will be discussed at length in the next chapter.

$$\begin{aligned} p(\mathbf{X}_t | I_0, I_1, \dots, I_t) &= \int_{\mathbf{X}_0} \int_{\mathbf{X}_0} \cdots \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_t | I_0, I_1, \dots, I_t) d\mathbf{X}_0 d\mathbf{X}_1 \cdots d\mathbf{X}_{t-1} = \\ &= \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t, \mathbf{X}_{t-1} | I_0, I_1, \dots, I_t) d\mathbf{X}_{t-1} \end{aligned} \quad (2.3)$$

Using first Bayes' rule (Eq. 2.4) and then assuming the independence of observations (Eq. 2.5), in particular, that I_t is conditionally independent of $[I_0, \dots, I_{t-1}]$ given \mathbf{X}_t , we can re-write the above expression as follows,

$$\begin{aligned} p(\mathbf{X}_t | I_0, I_1, \dots, I_t) &= \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t, \mathbf{X}_{t-1} | I_0, I_1, \dots, I_t) d\mathbf{X}_{t-1} \\ &= \int_{\mathbf{X}_{t-1}} \frac{p(I_0, I_1, \dots, I_t | \mathbf{X}_t, \mathbf{X}_{t-1}) p(\mathbf{X}_t, \mathbf{X}_{t-1})}{p(I_0, I_1, \dots, I_t)} d\mathbf{X}_{t-1} \end{aligned} \quad (2.4)$$

$$= \int_{\mathbf{X}_{t-1}} \frac{p(I_t | \mathbf{X}_t, \mathbf{X}_{t-1}) p(I_0, I_1, \dots, I_{t-1} | \mathbf{X}_t, \mathbf{X}_{t-1}) p(\mathbf{X}_t, \mathbf{X}_{t-1})}{p(I_t) p(I_0, I_1, \dots, I_{t-1})} d\mathbf{X}_{t-1} \quad (2.5)$$

Since the observation, I_t , at time t is assumed conditionally independent of all hidden states (past or future) given the state, \mathbf{X}_t at time t , we can further simplify Eq. 2.5 to Eq. 2.6. Then using conditional probability rules (Eq. 2.7), re-arranging terms (Eq. 2.8) and applying Bayes' rule again to the right-most term (Eq. 2.9) obtain the final recursive expression for Bayesian filtering (Eq. 2.10),

$$\begin{aligned} \underbrace{p(\mathbf{X}_t | I_0, I_1, \dots, I_t)}_{\text{Posterior at time } t} &= \int_{\mathbf{X}_{t-1}} \frac{p(I_t | \mathbf{X}_t, \mathbf{X}_{t-1}) p(I_0, I_1, \dots, I_{t-1} | \mathbf{X}_t, \mathbf{X}_{t-1}) p(\mathbf{X}_t, \mathbf{X}_{t-1})}{p(I_t) p(I_0, I_1, \dots, I_{t-1})} d\mathbf{X}_{t-1} \\ &= \int_{\mathbf{X}_{t-1}} \frac{p(I_t | \mathbf{X}_t) p(I_0, I_1, \dots, I_{t-1} | \mathbf{X}_{t-1}) p(\mathbf{X}_t, \mathbf{X}_{t-1})}{p(I_t) p(I_0, I_1, \dots, I_{t-1})} d\mathbf{X}_{t-1} \end{aligned} \quad (2.6)$$

⁴Similar expressions can be derived for higher order Markov assumptions, where an n-th order Markov assumption refers to the fact that the state \mathbf{X}_t is assumed to depend on the history of $[\mathbf{X}_{t-n}, \dots, \mathbf{X}_{t-1}]$ states.

$$= \int_{\mathbf{X}_{t-1}} \frac{p(I_t|\mathbf{X}_t)p(I_0, I_1, \dots, I_{t-1}|\mathbf{X}_{t-1})p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1})}{p(I_t)p(I_0, I_1, \dots, I_{t-1})} d\mathbf{X}_{t-1} \quad (2.7)$$

$$= \int_{\mathbf{X}_{t-1}} \frac{p(I_t|\mathbf{X}_t)}{p(I_t)} p(\mathbf{X}_t|\mathbf{X}_{t-1}) \frac{p(I_0, I_1, \dots, I_{t-1}|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1})}{p(I_0, I_1, \dots, I_{t-1})} d\mathbf{X}_{t-1} \quad (2.8)$$

$$= \int_{\mathbf{X}_{t-1}} \frac{p(I_t|\mathbf{X}_t)}{p(I_t)} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|I_0, I_1, \dots, I_{t-1}) d\mathbf{X}_{t-1} \quad (2.9)$$

$$= \int_{\mathbf{X}_{t-1}} \frac{1}{Z} p(I_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|I_0, I_1, \dots, I_{t-1}) d\mathbf{X}_{t-1} \\ = \frac{1}{Z} \underbrace{p(I_t|\mathbf{X}_t)}_{\text{Likelihood}} \int_{\mathbf{X}_{t-1}} \underbrace{p(\mathbf{X}_t|\mathbf{X}_{t-1})}_{\text{Temporal Prior}} \underbrace{p(\mathbf{X}_{t-1}|I_0, I_1, \dots, I_{t-1})}_{\text{Posterior at time } t-1} d\mathbf{X}_{t-1}, \quad (2.10)$$

where Z is a normalizing constant. The integral portion of the above equation is referred to as the *prediction* and the term before the integral, $p(I_t|\mathbf{X}_t)$, as the *likelihood*. Furthermore, the first term in the integral, is also known as the *temporal prior* that defines the dynamics or the state evolution process. It is worth noting that the above recursion terminates at $p(\mathbf{X}_0|I_0) = p(\mathbf{X}_0)$, where it is assumed that the distribution over the initial starting pose \mathbf{X}_0 is known. In the case of the pose estimation $p(\mathbf{X}_0|I_0) \neq p(\mathbf{X}_0)$ and itself needs to be inferred.

If the likelihood is Gaussian, $p(I_t|\mathbf{X}_t) = \mathcal{N}(I_t; A_o\mathbf{X}_t, \Sigma_o)$, the initial distribution, $p(\mathbf{X}_0)$, is Gaussian and temporal prior is linear with normally distributed noise, $p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t; A_d\mathbf{X}_{t-1}, \Sigma_d)$, the integral in Eq. 2.10 can be dealt with analytically. This model is commonly called the Kalman Filter and has been used successfully for articulated tracking in some cases [112]. While the Kalman filter provides a probabilistic solution to tracking, this model is only capable of dealing with uni-modal Gaussian predictions of the posterior. Hence, most state of the art probabilistic methods tend to avoid Kalman Filtering in favor of other models that make weaker assumptions on dynamics and observations (*e.g.* particle filtering).

It is worth mentioning that there is significant evidence that the posterior over pose is indeed non-Gaussian and is hard to model using simple parametric distributions. This arises due to non-linear dynamics of the human body and an often non-Gaussian observation model. For example, when a leg hits the ground during the walking cycle, the result is an inelastic collision between the foot and the ground plane that is highly non-linear. In terms of observations, based on simple geometry, we know that mapping between the 3D pose and the 2D pose (which is the only thing that we can observe in the image) is not one-to-one. This means that naturally an observed image would give rise to multiple hypothesis for the 3D pose⁵. Lastly, since body joints move over large ranges but have hard limits, they are not well modeled using Gaussian or other simple distributions.

Constructing models that encode these more realistic phenomena, leads to the forms of the integral in Eq. 2.10 that cannot be dealt with analytically. In such cases a common solution is to approximate the integral using numerical (*e.g.* Monte Carlo) integration. This leads to a family of methods that are commonly known as *Particle Filters*. Particle filters will be covered in more detail in Section 3.6.4. Particle filters have been extensively used for both rigid [157] and articulated object [52, 193] tracking. Unlike the Kalman Filter, Particle Filters are able to deal with complex and multimodal posterior distributions. Particle Filters tend to represent the posterior at time t using a weighted set of N samples (particles) $\{s_t^{(i)}, w_t^{(i)} | i \in [1, \dots, N]\}$, where $s_t^{(i)}$ is an i -th sample and $w_t^{(i)}$ is the corresponding weight, such that $\sum_{i=1}^N w_t^{(i)} = 1$. The most

⁵This ambiguity can be significantly reduced by using multiocular observations.

notable disadvantage of these methods is that they require sampling in high-dimensional spaces to represent the posterior. Since the number of samples required grows exponentially with the dimensionality, (*a.k.a. curse of dimensionality*), most methods rely on some heuristic function that designates the most plausible portion of the space to sample. Consequently, the efficiency of particle filters is greatly effected by the choice of this function.

Annealed Particle Filters (APF) [52]. Due to the high-dimensionality of the state space and often highly non-convex likelihood function with narrow peaks, the number of samples required to model the posterior in the realistic cases is intractably high. The Annealed Particle Filter tries to battle this problem using the concept of simulated annealing initially introduced by Kirkpatrick *et al.* in [116]. Instead of sampling the posterior $p(\mathbf{X}_t|I_0, I_1, \dots, I_t)$ directly a series of distributions with probability densities $p_0(\mathbf{X}_t), p_1(\mathbf{X}_t), \dots, p_M(\mathbf{X}_t)$ are introduced (also referred to in the context of APF as *layers*), where $p_m(\mathbf{X}_t)$ differs only slightly from $p_{m+1}(\mathbf{X}_t)$. These distributions are constructed such that $p_0(\mathbf{X}_t) = p(\mathbf{X}_t|I_0, I_1, \dots, I_t)$, and each subsequent distribution $p_{m+1}(\mathbf{X}_t)$ is designed such that the Markov Chain used to sample from it allows easier movement between regions of the state/search space. The last layer corresponding to $p_M(\mathbf{X}_t)$ allows movement between all regions of the state/search space. The usual method of achieving this is by setting $p_m(\mathbf{X}_t) \propto p_0(\mathbf{X}_t)^{\beta_m}$, where β_m is the *temperature* parameter such that $1 = \beta_0 > \beta_1 > \dots > \beta_M$. As a result the effect of the peaks in the posterior is introduced gradually by adjusting the temperature of the posterior distribution.

APF draws samples from the posterior by first simulating the Markov Chain corresponding to the distribution $p_M(\mathbf{X}_t)$, then using the resulting estimates of the distribution as initialization for simulating the Markov Chain corresponding to $p_{M-1}(\mathbf{X}_t)$ and so on until Markov Chain corresponding to the desired distribution (posterior) $p_0(\mathbf{X}_t)$ is simulated. The proposed method is a heuristic for avoiding local minima and handling narrow peaks that can be problematic when simulating the Markov Chain corresponding to the posterior $p_0(\mathbf{X}_t) = p(\mathbf{X}_t|I_0, I_1, \dots, I_t)$ directly.

Hierarchical Particle Filters [51, 136]. Hierarchical particle filters (*a.k.a. partitioned sampling*) handle the problem of high dimensionality of the state-space in a different way. They use search space decomposition to partition the search space into a number of independent searches. If the state space can be partitioned into parts that can be searched independently, then the computation time would be reduced significantly. Instead of complexity exponential in the number of degrees of freedom, we can have a search strategy that is linear in the number of partitions and exponential in the number of degrees of freedom within a partition. For example, if we partition our state $\mathbf{X}_t \in \mathbb{R}^d$ into K equal partitions (in most realistic cases the partitions will not be equal) $\mathbf{X}_t = [\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \dots, \mathbf{x}_{t,K}]^T$, where $\mathbf{x}_{t,k} \in \mathbb{R}^{d/K}$, then instead of exponential search strategy $O(c^d)$ we can have search strategy that is $O(Kc^{d/K})$, where c is a constant.

In the context of human motion and pose estimation, the partitioning often takes the following form: first find the torso, then given the torso find the head and the upper extremities, then given the upper extremities find the lower extremities, followed by hands and feet. While this strategy is very efficient it suffers from one significant disadvantage, it assumes that the parts that are high in the hierarchy can be localized well (*e.g.* torso). Depending on the imaging conditions and the exact partitioning strategy this assumptions may or may not hold. In general a dynamic data-driven strategy for the partitioned sampling would be preferred. The approach introduced in this thesis that uses graphical models to model the conditional independence between parts of the search space (that correspond to individual body parts) and uses particle message passing to do the search, can be viewed in this way - as dynamic iterative hierarchical search that is not committed to the particular partition strategy. It is worth noting that partitioned sampling can also be combined with annealed

particle filtering for further speed and robustness improvements [51].

Rao-blackwellized Particle Filters (RBPF) [153]. Standard Particle Filters assume that the integral in the Bayesian filtering equation cannot be computed analytically, in some cases however integrating analytically over part of the state space may be easy (*e.g.* a subset of variables in \mathbf{X}_t may be Gaussian). Rao-blackwellized Particle Filters make use of this fact by integrating analytically over the part of the state-space that can be integrated, and sampling the rest. As a result it can be shown that Rao-blackwellized Particle Filters provide a better estimate for the posterior distribution. While RBPF inference has been successfully used in various applications, few approaches have thus far attempted to use it for articulate human motion estimation [104].

Hybrid Methods often attempt to integrate particle filtering with deterministic (gradient) optimization of the posterior [38, 241]. One such example is Hybrid Monte Carlo (HMC) filter [38], that uses multiple Markov Chains that use posterior gradients to rapidly explore the state space, yielding fair samples from the posterior. The resulting approach is claimed to be several thousand times faster than the standard particle filter. Similarly, covariance scaled sampling [210] combines covariance scaled ‘oversized’ sampling with local optimization subject to joint and non-self-intersection constraints.

In [211] MCMC sampler is modified to include a potential function that focuses samples on nearby saddle points based on the local gradient and curvature of the input distribution. This strategy effectively finds local optima in the high dimensional space of articulated poses. Interpretation trees and inverse kinematic reasoning can be used to construct sampling schemes that account for long-range structural ambiguities of 3D human motion [209] observed from a monocular camera. This approach has also been extended in [208] by introducing variational temporal smoothing that accounts for temporal continuity in persistently multi-modal posterior.

Multiple Hypothesis Tracking is an alternative to the Particle Filtering. Instead of representing the posterior distribution over the state explicitly, MHT approaches [34, 131] often formulate the problem of inference as that of explicitly maintaining a fixed number of hypothesis that correspond to the modes of the posterior distribution.

2.9 Number of Views

As was discussed in the Section 2.4 the human body can be represented in either 2D or 3D. If the 2D representation is chosen then at least conceptually one view (or a single image) of the scene should be sufficient to infer the pose of visible parts. In the case of the 3D model, it is unclear how well one can expect to predict the pose from a single view, especially when motion information is unavailable. It is known that multiple 3D poses will result in the same 2D image projection, and as a result most approaches that attempt to solve this problem from monocular imagery must either rely on prior knowledge of the motion [125] or temporal information [193, 195, 206, 209] to resolve ambiguities. The problem of 3D pose inference, however, is significantly simplified when multiple views are available. At least conceptually with sufficient number of non-degenerate⁶ views, the body can be fully observed and the pose recovered with little or no prior assumptions on the motion.

⁶By degenerate views we mean views of the scene that give no additional information. For example, cameras that are very close together resulting in the nearly the same image, would be considered degenerate. Degeneracy may also depend on the features. For example, cameras that are located opposite to each other (180 degrees apart) can produce identical silhouettes.

2.9.1 Multiocular 3D Inference

Most approaches that deal with multiple views can be classified into the ones that either use the *visual hull* explicitly or backproject the 3D model into the image without explicitly reconstructing the volumetric representation. In both cases the knowledge of the camera parameters (both intrinsic and extrinsic) is essential to draw correspondences between information in different cameras.

Visual hull. Visual hull based approaches explicitly solve for the association of features from multiple views, resulting in the approximate 3D bounding geometry of the actual object. It can be shown that as the number of views increase the visual hull tends to approach the true shape of the object. Most visual hull approaches [28, 36, 39, 113] rely on a good background subtraction process and silhouettes to define the *generalized silhouette cone* that originates at the focal point of the camera and runs through the contour of the silhouette. The intersection of the cones from different cameras defines the upper bound on the space occupied by the object. More recent approaches of Voxel Coloring [36] also check color consistency across multiple views. The key problem with these approaches is their reliance on nearly perfect background separation. Noisy silhouettes from even a single camera will result in holes in the 3D volume, significantly corrupting the representation. To attempt to handle this, a probabilistic occupancy grid approach has been introduced in [65], where an equivalent of the visual hull can be obtained by taking the isosurface of the density at a given probability. Once the volume is recovered the tracking of the 3D shape can be performed either by stochastic meta descent [113] or iterative closest point [151].

Backprojection. Alternatively, approaches [52, 74, 77, 112, 197] have used backprojection of the model into the image to ease the burden of the low level observation association and the need for nearly perfect silhouette data. In visual hull approaches, hard decisions are made that may result in the loss of information early (at the feature level). Errors in that stage propagate. The backprojection methods delay hard decisions until later, when more information is available (such as the full body model), that may resolve ambiguities and deal with missing data more effectively. In backprojection methods, the multiple views are handled by the likelihood function, where independence is often assumed across camera views [52, 197] and the product over individual view-based likelihoods is taken as an overall measure of pose match.

2.9.2 Monocular 3D Inference

The case of inferring a full 3D pose of the person from single monocular image is the most general case considered by the community. In general, there have been two categories of approaches for doing this: (1) discriminative methods that attempt to learn the mapping directly from the image features to 3D pose, or (2) methods that recover the 2D pose first and then attempt to characterize the set of 3D poses that are consistent with the 2D interpretation. *Exemplar* and *probabilistic mapping* methods discussed below fall into the first category; *geometric* and what we call *probabilistic 3D reconstruction* methods into the second.

Exemplar Methods

The first class of approaches, that has already been discussed to some extent in Section 2.7, attempts to encode the appearance of the person using a set of generic features (*e.g.* shape context codebook entries [1, 4], histograms of oriented gradients [189], boundary fragments [88], Hu moments of the silhouette [179, 180]) and learn the mapping from these features to the 3D pose representation. One popular method is to collect a

large dataset of *feature-pose* associations and build a database. K -nearest neighbor [87, 166] or approximate K -nearest neighbor [189] methods can then be used to query the database for the entry that matches closely the features observed in the image and return the associated pose. In doing this, one must typically choose the similarity measure that is used to compare the features and an efficient (sub-linear) algorithm for search of the database. In [189] the similarity measure is recovered automatically based on the similarity relationship imposed on the training data. Alternatively in [11] the embedding that preserves the similarity is computed and used for faster query computation. These approaches are effective, but requires a database that spans the set of all possible poses, people, appearances and lighting conditions. This is often impractical for generic applications where motion, subject and lighting are unconstrained. Alternatively, a coarser discretization of this space can be used and a locally linear regression employed to interpolate between the poses [189] to get a continuous estimate for 3D pose. Even so, this class of approaches provides no probabilistic confidence measure for the recovered pose, nor can it easily embed any priors over the poses without explicitly changing the content of the database.

Probabilistic Mapping

Alternative approaches [1, 4, 179, 180, 205, 206] attempt to learn a direct continuous probabilistic mapping between the features and the 3D pose. The set of *feature-pose* associations in this case are treated as training data, based on which a much more compact representation of the relationship between these is recovered. For example, in [1, 4] a simple ridge regression and relevance vector regression (RVM) is used to characterize this mapping. While effective in some situations, the approach inherently assumes that the relationship between the 2D features (silhouettes in this case) and the 3D pose is linear and one-to-one, which in general is not true. In particular, more than one 3D pose can give rise to the same silhouette features depending on the view of the camera, orientation of the person and pose. To address this, the approach of [1, 4] was extended to include the multivalued probabilistic mapping [179, 180, 205, 206], and other features (edges and oriented gradients) that allow observation of occluding parts of the body [205, 206].

Geometric Methods

One alternative to the exemplar and probabilistic methods discussed was proposed by Taylor *et al.* in [222], where an approach was introduced capable of recovering a family of solutions based on pure geometry. Given the knowledge of the set of keypoints corresponding to joint positions in the image (obtained using a manual labeling procedure) and the knowledge of the 3D body segment lengths connecting these points, the approach was able to estimate the 3D configuration(s) of the body consistent with 2D constraints, modulo a scaling factor. While the original proposed method relied on the manually labeled set of keypoints, it was later extended by [148], to work from a set of automatically obtained keypoints. These keypoints were obtained by solving for the 2D pose of the person in an image using a bi-partite matching of shape context features [148]. This geometric approach, however, is relatively unstable with respect to small perturbations of the keypoint locations in 2D.

Probabilistic 3D Reconstruction

Based on the intuitive notion of using 2D pose to constrain the search for plausible 3D poses, Howe *et al.* in [90] proposed the approach that modeled the joint density of the 2D and 3D spatio-temporal poses defined

for a fixed size (11 frame) snippet of video. This joint Gaussian Mixture model learned using EM was then used to derive a conditional distribution of the 3D pose snippet conditioned on the observed 2D pose sequence. The overlapping 3D motion snippets were then merged using a weighted interpolation resulting in the continuous motion. While this spatio-temporal model helped to resolve some of the instabilities due to the jitter of joint positions in a single image, it still relied on the manual initialization at the first frame for 2D tracking, falling short of a fully automatic system. Similar in spirit, an approach was introduced by Brand in [29], where silhouette moments defined over a motion sequence were used to reconstruct 3D pose. More recently, there have been attempts to reconstruct the 3D pose from a monocular image, using intermediate 2D pose estimates; an approach taken in Chapter 6 of this thesis. In [127] for example, MCMC sampling was used to search for the 3D pose that is consistent with 2D probabilistic observations derived from a single image, based on automatic canonical detection of body parts.

2.9.3 Sub-space Methods

So far we have talked about approaches for monocular 3D pose estimation that are relatively general and assume little about the nature of the motion itself. Making assumptions about the motion, however, significantly simplifies the problem in many cases. In particular many simple repetitive motions can be represented by low-dimensional manifolds in a much higher dimensional space of all possible human motions. This is the key assumption of the sub-space methods. For example, the walking motion of a known subject can be parameterized by two parameters: phase and speed [55, 160]. Only one additional dimension is necessary to capture additional variations across the view of the person [125]. Variations across multiple walking people have been shown to be captured well in the 3 dimensional non-linear sub-space obtained using Gaussian Processes Latent Variable Models (GPLVM) [227]. GPLVM has a convenient probabilistic form that defines bi-directional mapping from and to the latent space. Furthermore, the latent space can be optimized to preserve dynamics [226], resulting in the model where pose estimation and tracking can all be performed in the latent space significantly reducing the computation required from the search in \mathbb{R}^d where typically $d = 30+$, to search in \mathbb{R}^3 . A similar approach that uses a Mixture of Factor analyzers for non-linear manifold learning was introduced by Li *et al.*[131]. There is, however, an inherent limitation in these models in that the motions must be relatively simple and/or cyclic. At the moment it is unclear how these approaches can be extended to work in more general settings where motions are of varying content and complexity.

2.10 Quantitative Evaluation

A variety of statistical [3, 4, 14, 52, 93, 196, 197, 206] as well as deterministic methods [147, 189, 222] have been developed for tracking people from single [3, 4, 59, 93, 122, 146, 147, 170, 173, 174, 178, 196] as well as multiple [14, 52, 77, 197] views. All these methods make different choices regarding the state space representation of the human body and the image observations required to infer this state from the image data. Despite clear advances in the field, evaluation of these methods remains mostly heuristic and qualitative. As a result, it is difficult to evaluate the current state of the art with any certainty or even to compare different methods with any rigor.

Quantitative evaluation of human pose estimation and tracking is currently limited due to the lack of common “ground truth” datasets with which to test and compare algorithms. Instead qualitative tests are

widely used and evaluation often relies on visual inspection of results. This is usually achieved by projecting the estimated 3D body pose into the image (or set of images) and visually assessing how well the estimates explain the image [52, 59, 174]. Another form of inspection involves applying the estimated motion to a virtual character to see if the movements appear natural [206]. The lack of the quantitative experimentation at least in part can be attributed to the difficulty of obtaining 3D ground truth data that specifies the true pose of the body observed in video data.

To obtain some form of ground truth, previous approaches have resorted to custom action-specific schemes (or tricks); *e.g.* motion of the arm along the circular plate of known diameter [112]. Alternatively synthetic data has been extensively used [3, 4, 77, 189, 206] for quantitative evaluation. With packages such as POSER (*e frontier*, Scotts Valley, CA), semi-realistic images of humans can be rendered and used for evaluation. Such images, however, typically lack realistic camera noise, often contain very simple backgrounds and provide simplified types of clothing. While synthetic data allows quantitative evaluation (3D pose is known), current datasets are still too simplistic to capture the complexities of natural images of people and scenes.

For 2D human pose/motion estimation, quantitative evaluation is more common and typically uses hand labeled data [93, 170, 173]. While quantitative evaluation does occur, the datasets are typically unique to each research group, preventing direct comparison of methods. Furthermore, for both 2D and 3D methods, no standard error metrics exist and results are reported in a variety of ways which prevent direct comparison; *e.g.* average root-mean-squared (RMS) angular error, silhouette overlap, joint center distance, *etc.*

One of the contributions of this thesis is the new dataset, that we call HUMANEVA, that contains large amount of synchronized motion capture and multi-camera video data. A number of subjects were captured using the specialized setup performing a set of predefined actions in regular clothing. As part of this effort, we also outline a set of metrics for measuring error, that in our view make it easier to compare various methods on equal footing. The details of the dataset and the metrics will be discussed in Section 5.7.1; the data can be obtained from <http://vision.cs.brown.edu/humaneva/>.

2.11 Generic Object Detection, Localization and Categorization

Thus far we have concentrate on the articulated object pose estimation and tracking, however, the approaches introduced can be generalized to work for generic object detection tasks. We briefly cover the three major classes of approaches for generic object detection, localization and categorization bellow.

2.11.1 Sliding Window Classifiers

Sliding window classifiers refer to the class of approaches that attempt to encode the appearance of the object as a whole and then classify the patches in the images as either conforming to the object model or not. The classifier is typically learned from a fixed size registered set of image templates. To find objects at different positions, scales and rotations in the image typically the image itself is transformed to a fixed set of canonical scales and orientations and the classifier is slid (hence the name) across this set of constructed images. Typically the classifier will not respond only at a single location and scale where an object is present, but rather will respond in the neighborhood producing a family of hypothesis. To resolve this the standard approach is to apply non-maximal suppression, and report only the results that correspond to the set of local maxima. Note that although the classifiers are learned, the non-maximal suppression procedure is typically

designed by hand.

One of the prime examples of the sliding window classifier is AdaBoost introduced by Viola and Jones [236]. A cascade of classifiers is learned based on Haar Wavelets that were discussed in Section 2.5.8. The cascade allows fast classification, by quickly rejecting regions that are unlike the object (regions of constant color or texture) and spending more time resolving harder ambiguous cases. The approach can also be amended to deal gracefully with detection of multiple objects, by choosing features that are common to multiple classes of objects (but still discriminative) for classification [223]. An alternative is to use a support vector machine (SVM) classifier based on either Haar wavelets [144] or PCA based features [124]. One important challenge for these methods is appearance changes that result from both the viewpoint of the object and the variations within an object class. Typically only minor variations in both can be accounted for by these classifiers.

2.11.2 Part-based Models

A number of authors in recent literature [59, 144, 249] suggested that modeling complex objects by components explicitly and then combining [59, 144] or statistically fusing [93, 249] the information is superior to the global appearance approaches (that model variations in parts implicitly) in the presence of partial occlusions, out-of-plane rotation and/or local lighting variations. Component-based detection is also capable of handling highly articulated objects, for which a single appearance model may be hard to learn. To this end, it is common to represent objects as collections of features with distinctive appearance, spatial extent, and position [33, 61, 144, 235, 236, 243]. There is, however, a large variation in how many features one must use and how these features are detected and represented. There are also variations in how much geometry is encoded in the model. Typically part based approaches detect a set of interest points or keywords, based on which local (often scale and/or rotation invariant) image descriptors are derived. The object models are then learned based on these descriptors in supervised [198, 236], semi-supervised [61] or non-supervised [204] fashion.

The simplest model in this category is the **bag of words** model [45, 58, 204] that originated in the document analysis community [84]. The key idea is that any object can be represented using a codebook of visual descriptors/codewords. In this model the spatial relationships between the parts are ignored and only the presence/absence of the codewords is encoded using a histogram based representation. As a result these approaches tend to be very useful in image categorization, where one only needs to reason about the object presence. They are not able to infer the position, rotation or configuration of the object in the image however.

The **constellation model** is a very influential model for object class detection that was introduced by Weber *et al.* [243] and later extended by Fergus *et al.* [61]. This is a generative model defined over interest point locations and appearances. Unlike the “bag of words” model, the constellation model strongly parameterizes the geometric relationships of parts using a joint Gaussian over both centroid positions of all parts and individual appearances of parts themselves. Assuming that we have a set of N parts in an object and the appearance of each part is encoded using a 128 dimensional SIFT [135] vector, we can express the model as a Gaussian in \mathbb{R}^{128N} . Since this model has a simple Gaussian form the probability of a set of N keypoints can easily be evaluated, however, we must search over all possible assignment of M descriptors found in the test image to N parts encoded in the model. Hence the complexity of performing localization of a single object is $O(M^N)$, where M is typically around 100 – 500. This exponential complexity in the number of

parts ensures that this model is only tractable for objects that can be encoded using small number of parts ($N \approx 5$). The model also cannot easily handle clutter or occlusions. A recent extension to this model called a *common frame* model [145] encodes the position of parts relative to the centroid of the object, leading to a more efficient inference algorithm.

The **pictorial structures** model [59] that was discussed in the context of disaggregated models for human pose and motion in Section 2.4.3 encodes a looser pair-wise geometric relationships between parts, allowing efficient inference of the configuration along with the position and orientation. Unlike the “constellation model”, the detection and localization can be done in the time linear in the number of parts. Similar models have also been introduced by Agarwal *et al.* [5], Amores *et al.*[7] and Opelt *et al.*[158]. These approaches differ significantly in the features used to encode the appearance and in the specifics of the model, however, the common underlying premise is to model both appearance and pair-wise geometric constraints on the parts.

2.11.3 Hierarchical Composition Models

Part-based models deal well with deformable and articulate objects, but also tend to be relatively slow (apart from the “bag of words” model that is not able to perform localization). To be able to deal with deformable structure faster, a new class of methods has recently started to emerge. This relatively novel class of models attempts to model compositionality of objects in terms of parts. This compositionality is most often encoded by a hierarchical model. In this model the root of the hierarchy corresponds to a full model of the object with all its intricacies, and the lower-levels to simpler features that are easier and faster to detect. This hierarchical structure facilitates rapid object detection and inference. In [263] a shape based hierarchy is defined and encoded using a statistical graphical model. The inference in this model can be done efficiently using Belief Propagation (BP), resulting in the reported performance that is 100 times faster than competitors.

Athitsos *et al.*[10] introduced a very flexible approach that uses grammar like syntax to detect and localize deformable objects that can have variable structure (*i.e.* varying number of sub-parts). One example of such a class of objects is branches with leaves. The approach is an extension of Hidden Markov Models (HMMs), often used for analysis of temporal data, that in this case is adapted to modeling of the variable deformable spatial structure of an object. In a similar attempt, probabilistic grammars have also been used by Zhu *et al.* [262] to model and detect objects.

CHAPTER 3

Graphical Models and Inference

Graphical models have a wide applicability in statistics, machine learning, statistical physics, and more recently computer vision. Graphical models capture the way a joint distribution over all random variables can be decomposed into a product of factors each depending on only a subset of variables. This local decomposition of the joint distribution often leads to tractable inference algorithms. Graphical models also provide simple and intuitive way to visualize the structure of probabilistic models.

A probabilistic graphical model in general can be encoded using a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that comprises of a set of *nodes* or *vertices*, \mathcal{V} , and a set of *edges*, \mathcal{E} . Each vertex, $i \in \mathcal{V}$, in this graph is associated with a random variable \mathbf{X}_i . These variables can either be continuous or discrete depending on the problem. Each edge $(i, j) \in \mathcal{E}$ can be thought of as a probabilistic relationship between random variables associated with pair of distinct nodes $i \in \mathcal{V}$ and $j \in \mathcal{V}$. It is often useful to partition vertices, \mathcal{V} , in a graphical model into two disjoint sets, $\mathcal{V} = \{\mathcal{V}_\mathbf{X}, \mathcal{V}_\mathbf{Y}\}$, where the second set $\mathcal{V}_\mathbf{Y}$, corresponds to the nodes in the graph that are associated with variables $\mathbf{Y} = \{\mathbf{Y}_i | i \in [1, \dots, M]\}$ (where $M = |\mathcal{V}_\mathbf{Y}|$) that are directly observed, and the first set $\mathcal{V}_\mathbf{X}$ corresponds to the nodes in the graph that correspond to variables $\mathbf{X} = \{\mathbf{X}_i | i \in [1, \dots, N]\}$ (where $N = |\mathcal{V}_\mathbf{X}|$), that are not observed directly but the value of which is of some interest. It is notationally convenient to shade the nodes $\in \mathcal{V}_\mathbf{Y}$ gray, to make it visually clear that they can be observed.

Graphical models in general can be categorized into three categories: directed, undirected, and factor graphs. Directed models, also called *Bayesian Networks* (BN), are useful for expressing causal relationships. If the graph is directed then the edges, that are often depicted using arrows, correspond to the conditional dependencies of the child nodes (nodes toward which the arrows are pointing) on the parents (nodes from which the arrows originate). *Undirected models*, also known as *Markov Random Fields* (MRF), are used to encode constraints or correlations between random variables. In undirected graphical models the edges are depicted using arrow-less lines between nodes. *Factor graphs* are a relatively recent addition to the graphical model family, that generalizes both directed and undirected models. A factor graph is defined as an undirected bipartite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{F}, \mathcal{E}\}$, where \mathcal{V} and \mathcal{E} are defined as before, and \mathcal{F} is the set of additional vertices that are called *factors*.

Graphical models themselves only encode the structure of the joint distribution using a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ (or $\mathcal{G} = \{\mathcal{V}, \mathcal{F}, \mathcal{E}\}$ in the case of the factor graph), the specific forms of the relationships between random variables in the graph are not specified explicitly. Hence, in addition to specifying the graphical model, one must also specify the parameters of the graphical model, θ , where the form of these parameters will depend

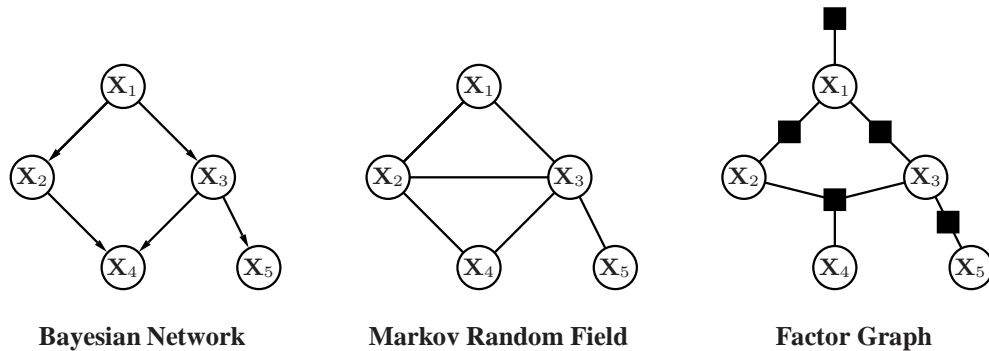


Figure 3.1: **Graphical model families.** Three families of graphical models that will be discussed in this chapter are illustrated. All three graphs can encode the same underlying joint distribution, $p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5)$, given the proper choice of parameters. Different choices of parameters would lead to different encoded joint distributions.

on the problem and the parameterization chosen for the variables and their relationships.

While graphical models define a rich set of models, there are only a few canonical operations that one is often interested in performing using these models. In particular, (1) learning of model structure, (2) learning of model parameters given the structure of the model, and (3) inference using the model where both structure and the parameters are known. The first task is by far most complex and deals with estimating the nodes, \mathcal{V} , in the graph and the connections between nodes, \mathcal{E} , corresponding to the relationship between random variables. In general to be able to do useful structure learning one typically must assume sparseness priors on both the edges and the nodes, attempting to recover the graph with as few nodes and edges as possible subject to the observed data. We will not address structure learning (*a.k.a. model selection*) in this thesis, and refer readers to [12, 129, 200, 201] for some recent work in this area. *Parameter learning* refers to estimating parameters θ given the model structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and subject to the data observed. We will cover a few examples of this in Section 3.4. The last task of *inference* is central to this thesis and will be covered in depth in this chapter. Inference in the graphical model, typically refers to finding the value, or the distribution over the values of all or some sub-set of hidden variables given the observations. Consequently parameter learning can often be cast as an inference problem itself.

In the following sections we introduce and compare several different classes of graphical models, including directed, undirected and factor graphs. We also introduce some specific instances of models within each class that are both common and useful for the purposes of this thesis. We also introduce methods for learning parameters and doing inference in these models.

3.1 Graphical Model Building Blocks

In this section we will introduce the set of distributions commonly referred to in this thesis and their properties. These distributions will play a key role in constructing more complex models used throughout this thesis, and in doing inference in these models.

3.1.1 Exponential Family

The *exponential family* of distributions is a class of distributions that serve as building blocks in graphical models, and give rise to rich probabilistic models used throughout the thesis. The distribution $p(\mathbf{X}|\theta)$, where

\mathbf{X} is a random variable and θ is a set of parameters, is said to be part of the *exponential family* if it can be written in the following form:

$$p(\mathbf{X}|\theta) = \frac{1}{Z(\theta)} h(\mathbf{X}) \exp [\theta^T t(\mathbf{X})] \quad (3.1)$$

where,

- θ is a vector of parameters (*a.k.a. natural or canonical parameters*)
- $t(\mathbf{X})$ is a function referred to as *sufficient statistics*
- $Z(\theta)$ is a *normalizing constant* (*a.k.a. partition function*) defined as
 - $Z(\theta) = \int h(\mathbf{X}) \exp [\theta^T t(\mathbf{X})] d\mathbf{X}$ for continuous variable \mathbf{X} , and
 - $Z(\theta) = \sum_{\mathbf{X}} h(\mathbf{X}) \exp [\theta^T t(\mathbf{X})]$ for discrete choice of \mathbf{X}
- $h(\mathbf{X})$ is a function of \mathbf{X} .

Many distributions can be written in this form, including Bernoulli, Poisson, Gaussian, Beta and Gamma densities. While the exponential family has many convenient properties, one that is worth mentioning is that the joint probability of N *i.i.d.* samples from the distribution, $\mathcal{D} = \{x_i \sim p(\mathbf{X}|\theta) | i \in [1, \dots, N]\}$, can be written in the following form,

$$\begin{aligned} p(\mathcal{D}|\theta) = p(x_1, \dots, x_N|\theta) &= \prod_{i=1}^N p(x_i|\theta) \\ &= \prod_{i=1}^N \frac{1}{Z(\theta)} h(x_i) \exp [\theta^T t(x_i)] \\ &= \left[\prod_{i=1}^N \frac{1}{Z(\theta)} h(x_i) \right] \exp \left[\theta^T \sum_{i=1}^N t(x_i) \right], \end{aligned} \quad (3.2)$$

which suggests that the dimensionality of the sufficient statistic remains the same with the number of samples. This, in turn means that in order to characterize a distribution in the exponential family, it is sufficient to compute the sufficient statistics. Once we have sufficient statistics for the distribution the samples themselves give no additional information about the distribution that generated them. This gives a convenient compact form for representing distributions in this family. For the list of other common properties of exponential family we refer the reader to [22, 107].

3.1.2 Gaussian Distribution and Properties

In this section we will review a *Gaussian* (or *Normal*) distribution, which is a prime example of the exponential family. A univariate Gaussian distribution with mean μ and variance σ^2 on random variable $\mathbf{X} \in \mathbb{R}$ can be written as,

$$p(\mathbf{X}|\mu, \Sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left[-\frac{1}{2} \frac{(\mathbf{X} - \mu)^2}{\sigma^2} \right]. \quad (3.3)$$

Alternatively we can also introduce the shorthand notation $\mathcal{N}(\mathbf{X}|\mu, \Sigma)$ or $\mathcal{N}(\mathbf{X}; \mu, \Sigma)$. It is easy to see that a univariate Gaussian is an exponential family distribution with the following parameterization,

$$\theta = \begin{bmatrix} \mu/2\sigma^2 \\ -1/2\sigma^2 \end{bmatrix}, \quad t(\mathbf{X}) = \begin{bmatrix} \mathbf{X} \\ \mathbf{X}^2 \end{bmatrix}, \quad Z(\theta) = \exp \left(\frac{\mu}{2\sigma^2} + \log \sigma \right), \quad h(\mathbf{X}) = \frac{1}{2\sqrt{2\pi}}. \quad (3.4)$$

If \mathbf{X} is multivariate random variable, $\mathbf{X} \in \mathbb{R}^d$, then the distribution can be written in the more general form,

$$p(\mathbf{X}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left[-\frac{1}{2}(\mathbf{X} - \mu)^T \Sigma^{-1}(\mathbf{X} - \mu) \right], \quad (3.5)$$

where Σ is now a covariance matrix and μ a multivariate mean. The Gaussian distribution has a number of convenient properties that make it very useful for modelling and inference tasks. The two most important properties that relate to the product of Gaussian distributions and conditional distribution of jointly Gaussian variables are stated below.

Product of Gaussian distributions

Product of two or more Gaussian distributions is also a Gaussian distribution. For example, product of M Gaussian distributions $p(\mathbf{X}_i) = \mathcal{N}(\mathbf{X}_i|\mu_i, \Sigma_i)$, $i \in [1, \dots, M]$ is

$$p(\mathbf{Y}) = \prod_{i=1}^M p(\mathbf{X}_i) = \mathcal{N}(\mathbf{Y}|\mu_{\mathbf{Y}}, \Sigma_{\mathbf{Y}}), \quad (3.6)$$

where

$$\Sigma_{\mathbf{Y}} = \left(\sum_{i=1}^M \Sigma_i^{-1} \right)^{-1} \quad \mu_{\mathbf{Y}} = \Sigma_{\mathbf{Y}} \left(\sum_{i=1}^M \Sigma_i^{-1} \mu_i \right). \quad (3.7)$$

Conditional Gaussian distribution

A conditional distribution of two or more jointly Gaussian variables is also a Gaussian [22, 217]. Consider a case of two jointly Gaussian variables \mathbf{X} and \mathbf{Y} ,

$$\mathcal{N} \left(\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \middle| \begin{bmatrix} \mu_{\mathbf{X}} \\ \mu_{\mathbf{Y}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathbf{X}} & \Sigma_{\mathbf{XY}} \\ \Sigma_{\mathbf{YX}} & \Sigma_{\mathbf{Y}} \end{bmatrix} \right). \quad (3.8)$$

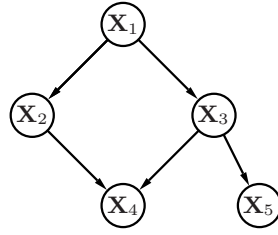
We can write conditional distribution $p(\mathbf{X}|\mathbf{Y})$ as a normal distribution with the following parameters for mean and covariance respectively:

$$\mu_{\mathbf{X}|\mathbf{Y}} = \Sigma_{\mathbf{XY}} \Sigma_{\mathbf{Y}}^{-1} (\mathbf{Y} - \mu_{\mathbf{Y}}) \quad (3.9)$$

$$\Sigma_{\mathbf{X}|\mathbf{Y}} = \Sigma_{\mathbf{X}} - \Sigma_{\mathbf{XY}} \Sigma_{\mathbf{Y}}^{-1} \Sigma_{\mathbf{YX}}. \quad (3.10)$$

3.2 Bayesian Networks

Bayesian Networks is a family of graphical models that characterize how the joint distribution over a set of N variables, $p(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N)$, factors into a set of conditional relationships imposed by the structure of the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. By the product rule, it can be shown that the joint distribution defined by the graph can be written as the product of conditional distributions for each node, where the variable associated with the node is conditioned on all the parents of that node in the graph. Hence, for a general directed graph with $N = |\mathcal{V}|$ variables, the joint distribution can be written as:



Conditional Independence Constraints
(imposed by the graph structure)

$$\begin{aligned} \mathbf{X}_1 &\perp\!\!\!\perp \{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\} \\ \mathbf{X}_2 &\perp\!\!\!\perp \{\mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\} \\ \mathbf{X}_3 &\perp\!\!\!\perp \{\mathbf{X}_2, \mathbf{X}_4, \mathbf{X}_5\} \\ \mathbf{X}_4 &\perp\!\!\!\perp \{\mathbf{X}_1\} \mid \{\mathbf{X}_2, \mathbf{X}_3\}, \mathbf{X}_4 \perp\!\!\!\perp \{\mathbf{X}_5\} \\ \mathbf{X}_5 &\perp\!\!\!\perp \{\mathbf{X}_1\} \mid \{\mathbf{X}_3\}, \mathbf{X}_5 \perp\!\!\!\perp \{\mathbf{X}_2, \mathbf{X}_4\} \end{aligned}$$

$$p(\mathbf{X}) = p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5) = p(\mathbf{X}_5|\mathbf{X}_3)p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1)$$

Figure 3.2: **Bayesian Networks.** Example of Bayesian network graphical model. The joint distribution factors into the product of conditional distributions as illustrated above. All the conditional independences imposed by the graph itself are also listed. Notice that even though there seems to be a loop in the graph, there are no directed cycles.



Figure 3.3: **Markov Chain.** Graphical model corresponding to a first-order Markov Chain. First-order Markov assumption encoded in the model presumes that the state, \mathbf{X}_t , at time t is only a function of the state at $t - 1$ for all $t \in [2, \dots, T]$, where in the example above $T = 9$.

$$p(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{|\mathcal{V}|}) = \prod_{i \in \mathcal{V}} p(\mathbf{X}_i | \mathbf{X}_{A(i)}), \quad (3.11)$$

where $A(i) \in \mathcal{V}$ is defined as a function that returns all parents of the node $i \in \mathcal{V}$ in a graph; $\mathbf{X}_{A(i)}$ is then the set of associated variables $\{\mathbf{X}_k | k \in A(i)\}$. The equation above expresses the factorization properties of the joint distribution and holds for all joint distributions and all definitions of variables \mathbf{X}_i , $i \in \mathcal{V}$. In order to ensure that factorization holds, an important restriction on the graph topology must be maintained. In particular, graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ cannot contain cycles, (*i.e.* it must be a *directed acyclic graph* (DAG)). In other words, there cannot exist a path from any node in the graph along the directed edges that leads back to the node itself. Example of the directed graphical model and the factorization of the joint distribution over all the variables is given in Figure 3.2.

3.2.1 Markov Chains

Markov Chains are among the simplest directed graphical models. A first-order Markov Chain is defined on a series of random variables $\{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ such that the following conditional independence holds for $n \in [1, \dots, N - 1]$:

$$p(\mathbf{X}_{n+1} | \mathbf{X}_1, \dots, \mathbf{X}_n) = p(\mathbf{X}_{n+1} | \mathbf{X}_n). \quad (3.12)$$

This conditional independence can be encoded in the graphical model as is shown on Figure 3.3. The Markov Chain can then be specified by the initial distribution $p(\mathbf{X}_1)$ and the conditional distribution for the subsequent variables (*a.k.a.* transition probabilities). A Markov Chain is called *homogenous* if the conditional

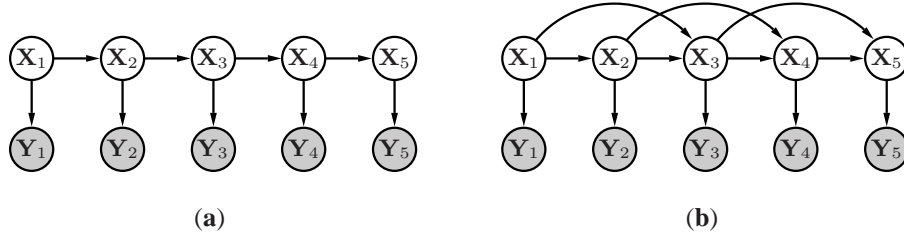


Figure 3.4: **Hidden Markov Models.** Directed graphical model representation of the temporal Hidden Markov Model (HMM) with $T = 5$ observations and hidden variables are shown. Models that illustrate first-order and second-order Markov dynamics are shown in (a) and (b) respectively. In general an N -th order Markov assumption in the directed model above states that the hidden variable \mathbf{X}_t is conditionally independent of all observation past or future given the estimates for $\{\mathbf{X}_{t-N}, \dots, \mathbf{X}_{t-1}\}$, i.e. $p(\mathbf{X}_t | \mathbf{X} \setminus \mathbf{X}_t) = p(\mathbf{X}_t | \mathbf{X}_{t-N}, \dots, \mathbf{X}_{t-1})$, where $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T\}$.

distributions are the same for all variables in the model. Marginal probability of a particular variable in the chain can be computed recursively using the following,

$$p(\mathbf{X}_{n+1}) = \sum_{\mathbf{X}_n} p(\mathbf{X}_{n+1} | \mathbf{X}_n) p(\mathbf{X}_n) \quad \text{or} \quad p(\mathbf{X}_{n+1}) = \int_{\mathbf{X}_n} p(\mathbf{X}_{n+1} | \mathbf{X}_n) p(\mathbf{X}_n) d\mathbf{X}_n, \quad (3.13)$$

depending on whether the variables are discrete or continuous respectively. The distribution $p(\mathbf{X})$ is said to be *stationary* if the following condition holds,

$$p(\mathbf{X}) = \sum_{\mathbf{X}} p(\mathbf{X}_{n+1} | \mathbf{X}) p(\mathbf{X}) \quad \text{or} \quad p(\mathbf{X}) = \int_{\mathbf{X}} p(\mathbf{X}_{n+1} | \mathbf{X}) p(\mathbf{X}) d\mathbf{X}. \quad (3.14)$$

Markov Chains are extremely useful for inference of other more complex models as will be shown in Section 3.6.3. In particular, one can design a Markov Chain in such a way as to facilitate sampling from an arbitrary complex distribution. To this end another property of Markov Chains must be introduced, *ergodicity*. Ergodicity ensures that for a given choice of the stationary distribution $p(\mathbf{X})$, $p(\mathbf{X}_n)$ will converge to $p(\mathbf{X})$ as $n \rightarrow \infty$ irrespective of initial choice of distribution $p(\mathbf{X}_1)$. Such a stationary distribution is also called an *equilibrium* distribution. It is worth mentioning that while a Markov Chain may have a number of stationary distributions, an ergodic Markov Chain will have only one equilibrium distribution.

3.2.2 Hidden Markov Models

One of the most common Bayesian Networks is the *Hidden Markov Model* (HMM). A Hidden Markov Model is often used to model temporal stochastic processes. The HMM is widely used in speech recognition [168], natural language processing [35], analysis of biological data [81], and computer vision applications. In computer vision, HMMs are particularly useful for temporal modeling of object motion over time [14, 51, 52, 157].

In an HMM, as is suggested by the name the nodes in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ are partitioned into two subsets $\mathcal{V} = \{\mathcal{V}_{\mathbf{X}}, \mathcal{V}_{\mathbf{Y}}\}$, where the first set corresponds to the hidden variables $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T\}$ that are not directly observed and the second to the observations $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_T\}$ that are generated based on the hidden states. Furthermore there is an evolution process that is defined on the hidden variables \mathbf{X}_t , $t \in [1, \dots, T-1]$. For example, we can assume that the hidden states evolve according stationary first-order

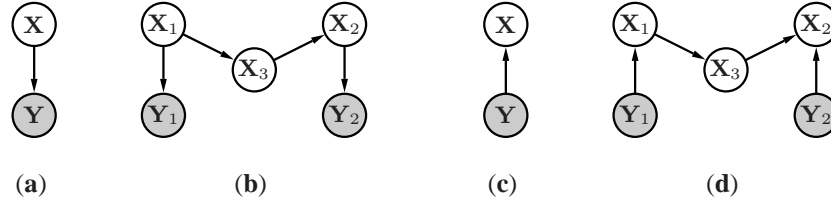


Figure 3.5: **Generative and discriminative graphical models.** A symbolic graphic representation of generative (a) and discriminative (b) models are shown. A specific instance of the generative and discriminative model is shown in (b) and (c) respectively.

temporal Markov process as illustrated in Figure 3.4 (a). The joint distribution over all variables can then be written according to the Bayesian Network rules as follows:

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{Y}_1 | \mathbf{X}_1) p(\mathbf{X}_1) \prod_{t=2}^T p(\mathbf{Y}_t | \mathbf{X}_t) p(\mathbf{X}_t | \mathbf{X}_{t-1}). \quad (3.15)$$

If we further assume that hidden variables are discrete and can assume K states, then the total number of parameters required to encode the model is $|\theta| = K + K(K - 1)$, where the prior, $p(\mathbf{X}_1)$, can be encoded using K parameters and the conditional, $p(\mathbf{Y}_t | \mathbf{X}_t)$, using a matrix with $K(K - 1)$ parameters (where each parameter will encode the probability of transitioning from a given state at time $t - 1$ to any other state at time t). Since the sum of all K transition probabilities from a given state at time $t - 1$ is 1, there are actually only $K - 1$ free parameters. Higher order Markov models are also possible (e.g. a second-order temporal Hidden Markov Model is illustrated in Figure 3.4 (b)). However, the number of parameters required to encode the model will grow exponentially with the order of the model. In particular, M -th order HMMs will require $|\theta| = K + K^M(K - 1)$ parameters. It is worth mentioning that HMMs can be used to encode spatial as well as temporal structure. For example, HMMs have been successfully used for deformable shape matching in [10]. HMMs also do not need to be stationary, in which case the number of parameters will depend on the length of the chain itself. For a sequence with T observations, an M -th order HMM, will contain $|\theta| = K + (T - 1)K^M(K - 1)$ parameters.

The formulation above also holds if variables are continuous. In such cases often a linear-Gaussian dynamical model is chosen for the conditional $p(\mathbf{X}_t | \mathbf{X}_{t-1}) = a\mathbf{X}_{t-1} + b$, where a corresponds to the deterministic component and b to the noise that is usually assumed to be zero mean normal $b \sim \mathcal{N}(0, \Sigma)$. This is also known in the literature as the *autoregressive* (AR) dynamical model. Generic and articulated object tracking in the computer vision literature is often formulated using HMMs with first- or second-order autoregressive dynamics [24, 51].

3.2.3 Generative and Discriminative Graphical Models

The Hidden Markov Model is a prime example of a *Generative Graphical Model*. Generative graphical models refer to the class of models that aim at modeling the process by which the data is generated. They attempt to estimate the joint distribution over all hidden and observed variables and then manipulate the joint distribution to compute the desired probability densities (e.g. marginals or conditionals). For example, if one is interested in inferring the state of hidden variables $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$, as is the case for classification, then the joint distribution $p(\mathbf{X}, \mathbf{Y})$ can be conditioned on the observations $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_M\}$

resulting in the desired conditional $p(\mathbf{X}|\mathbf{Y})$. Since the joint distribution is often complex and can contain many variables it is often desirable to constrain the distribution before attempting to estimate it. *Conditional independence constraints* encoded in the graphical models are one way of doing this. Alternatively or in conjunction, one can impose prior distributions over the parameters of the joint distribution or the variables themselves, such priors are often referred to as *hyperpriors*. Typically one can impose a hierarchy of such priors, with the model being less sensitive to the parameters that are higher in the hierarchy.

In general, the generative model can be drawn symbolically in the form of the directed graph in Figure 3.5 (a). The hyper-node \mathbf{X} in the graph corresponds to all the hidden variables $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ and node $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_M\}$ to all observations. The arrow designates that all arrows in the graph are only allowed to point from the hidden variables to the observations and not vice versa (nothing is assumed about the relationship of hidden variables themselves). To estimate the joint distribution, that is often written according to directed graphical model rules as, $p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})$, one needs to estimate the *prior* $p(\mathbf{X})$ and the *conditional* (often referred to as the *likelihood*) $p(\mathbf{Y}|\mathbf{X})$. Since generative models are flexible and can potentially encode all knowledge about the variables and their relationships, they often have good generalization properties (*i.e.* they can deal with data that was not part of the training). In other words, since generative approaches aim at modeling the data generation process, they can draw inferences about all possible data values. Unfortunately, building realistic generative models is both hard and computationally expensive, hence, most approaches only model the most important relationships that result in weak but tractable generative models.

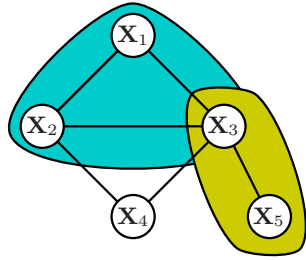
Since in some cases we may be only interested in the prediction of the hidden variable state from the observed data, there may be no need to estimate the full joint distribution $p(\mathbf{X}, \mathbf{Y})$. Instead, *discriminative models* attempt to estimate the conditional $p(\mathbf{X}|\mathbf{Y})$ directly. Alternatively these models can be thought of as estimating the direct (and often probabilistic) mapping from the observations to the hidden states. The directed graphical model depiction of discriminative models can be seen in Figure 3.5 (c). The major difference is in the direction of the arrows that now point from the observations to the hidden nodes depicted by the hyper-nodes \mathbf{X} and \mathbf{Y} . This has significant implications, however, in that discriminative models cannot model any prior information about the hidden variables; conversely they often encode prior relationships on the data itself making it hard for the discriminative approaches to generalize.

3.3 Undirected Graphical Models

Undirected graphical models capture correlations or constraints between variables instead of causal (or conditional) relationships encoded by directed graphical models. Undirected graphical models fall into the category of generative models described in Section 3.2.3.

3.3.1 Markov Random Fields

Markov Random Fields (MRFs) represent a class of graphical models that can be characterized by an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the edges, \mathcal{E} , imply conditional independence relationships between random variables associated with the nodes, \mathcal{V} . In particular, for a given node i with associated random variable \mathbf{X}_i , let us assume we have a function $A(i) \subset \mathcal{V}$ that returns all neighbors that are connected to i by an edge. We can then express \mathcal{V} for a given choice of i using three disjoint sets $\mathcal{V} = \{i, A(i), B(i)\}$,



Conditional Independence Constraints
(imposed by the graph structure)

$$\begin{aligned} \mathbf{X}_1 &\perp\!\!\!\perp \{\mathbf{X}_4, \mathbf{X}_5\} \mid \{\mathbf{X}_2, \mathbf{X}_3\} \\ \mathbf{X}_2 &\perp\!\!\!\perp \{\mathbf{X}_5\} \mid \{\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4\} \\ \mathbf{X}_4 &\perp\!\!\!\perp \{\mathbf{X}_1, \mathbf{X}_5\} \mid \{\mathbf{X}_2, \mathbf{X}_3\} \\ \mathbf{X}_5 &\perp\!\!\!\perp \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_4\} \mid \{\mathbf{X}_3\} \end{aligned}$$

$$p(\mathbf{X}) = \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) \psi_{35}(\mathbf{X}_3, \mathbf{X}_5)$$

Figure 3.6: **Markov Random Field.** Example of an MRF graphical model. The joint distribution factors into the product of potentials as illustrated above. All the conditional independences imposed by the graph itself are also listed.

where $B(i) = \mathcal{V} \cap \{i, A(i)\}$. The set of random variables associated with $A(i)$ and $B(i)$ can be written as follows $\mathbf{X}_{A(i)} = \{\mathbf{X}_j \mid j \in A(i)\}$ and $\mathbf{X}_{B(i)} = \{\mathbf{X}_j \mid j \in B(i)\}$ respectively. The conditional independence constraints encoded by the graph can then be expressed using the following relationship, $p(\mathbf{X}_i, \mathbf{X}_{B(i)} \mid \mathbf{X}_{A(i)}) = p(\mathbf{X}_i \mid \mathbf{X}_{A(i)}) p(\mathbf{X}_{B(i)} \mid \mathbf{X}_{A(i)})$ for $\forall i \in \mathcal{V}$. In other words, we can say that any variable \mathbf{X}_i is *conditionally independent*, given its neighbors, of all other variables in the model. Conditional independence is very important in design of efficient inference algorithms for these graphical models.

For MRFs it is useful to define the notion of the *clique*. A clique, c , is defined as the set of fully connected nodes in the graph. The random variables associated with a clique can be denoted as $\mathbf{X}_c = \{\mathbf{X}_i \mid i \in c\}$. According to the Hammersley and Clifford Theorem (restated here for completeness), the joint distribution over all variables can be parameterized by a product of *potential functions* defined on the cliques of the graph. In particular,

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{X}_c), \quad (3.16)$$

where \mathcal{C} is the set of all cliques in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. It is easy to see that in general the parameterization using the cliques is not unique. To get a unique parameterization, often *maximal cliques* are used to represent the graph, where maximal clique is defined as the largest set of fully connected nodes in the graph.

For example, the joint distribution for the undirected graph in Figure 3.6, can be written as follows:

$$p(\mathbf{X}) = \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) \psi_{35}(\mathbf{X}_3, \mathbf{X}_5). \quad (3.17)$$

Theorem 3.3.1 (Hammersley-Clifford Theorem¹) Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graphical model, where each vertex $i \in \mathcal{V}$ corresponds to the random variable \mathbf{X}_i . Let \mathcal{C} be a set of cliques of the graph \mathcal{G} . Then, a probability distribution defined as the product of normalized positive functions (symmetric in their arguments) defined on the cliques is always Markov with respect to the graph,

$$p(\mathbf{X}) \propto \prod_{c \in \mathcal{C}} \psi_c(\mathbf{X}_c). \quad (3.18)$$

Alternatively, any positive joint density function, $p(\mathbf{X}) > 0, \forall \mathbf{X}$, which is Markov with respect to the

¹Formulation of Hammersley-Clifford Theorem used here is borrowed to a large extent from [218].

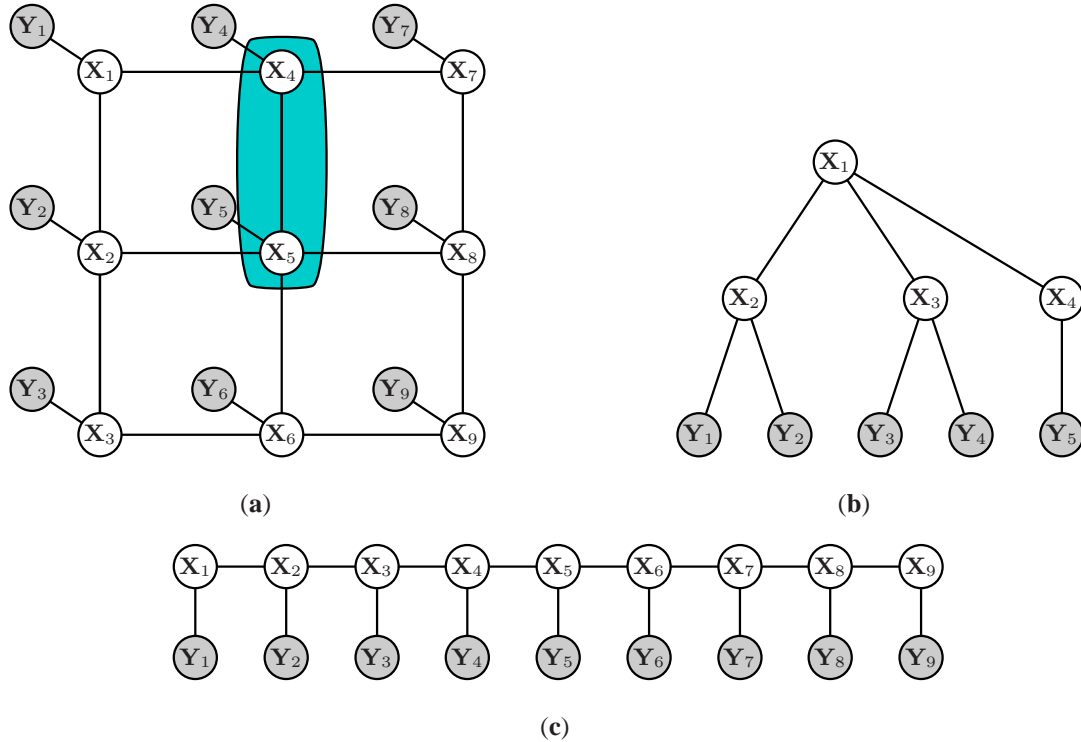


Figure 3.7: **Pair-wise Markov Random Field.** Examples of three common pair-wise Markov Random Fields are shown. Nodes corresponding to hidden variables $\{X_1, \dots, X_N\}$ are depicted using unfilled circles, observations $\{Y_1, \dots, Y_M\}$ using shaded nodes. In (a) a grid-based graphical model is depicted, often used in computer vision application (e.g. image restoration, super-resolution, image segmentation and stereo). In (b) a tree-structured graph is depicted. Inference methods in tree-structured graphs, such as Belief Propagation, are often shown to have favorable properties. Lastly, in (c) an undirected version of the Hidden Markov Model obtained by *moralization* (see text) is illustrated.

graph, implies that there exist positive functions ψ_c (symmetric in their arguments) such that Eq. 3.18 holds.

Proof. The proof of this theorem is somewhat involved, and we refer the reader to the original published version of this theorem in [40].

3.3.2 Pair-wise Markov Random Fields

A special case of the more general MRF framework is the *pair-wise* Markov Random Field where the cliques are explicitly restricted to the pairs of nodes connected by the edges in the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Such special case is clearly a restriction on the more general MRF formulation presented in the previous section, but is useful for many applications. In such models, it is often convenient to partition the nodes $\mathcal{V} = \{\mathcal{V}_X, \mathcal{V}_Y\}$, that correspond to observable variables $\mathbf{Y} = \{Y_1, Y_2, \dots, Y_M\}$ and hidden variables $\mathbf{X} = \{X_1, X_2, \dots, X_N\}$ respectively. The potential functions can also be partitioned into two disjoint sets, the first set corresponding to the edges that are between the hidden variables and the observations (*a.k.a. local likelihoods*), and the second set corresponding to the edges between hidden variables. We will denote the first set of functions

using $\phi_i(\mathbf{X}_i, \mathbf{Y}) = \psi_i(\mathbf{X}_i, \mathbf{Y})$ and the second set using the old notation $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$. The reason for considering two sets of potentials separately, is motivated by the empirical observation that the two function types typically vary significantly in their complexity. In particular, while $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ often have simple parametric forms, $\phi_i(\mathbf{X}_i, \mathbf{Y})$ are often considerably more complex. The joint distribution can be written using the following:

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{E}, i \in \mathcal{V}_{\mathbf{X}}, j \in \mathcal{V}_{\mathbf{X}}} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \prod_{i \in \mathcal{V}_{\mathbf{X}}} \phi_i(\mathbf{X}_i, \mathbf{Y}). \quad (3.19)$$

Undirected Version of Hidden Markov Model

A Bayesian Network can always be converted into MRF by a process called *moralization*, by *marrying* (connecting) all the parents of the common child and removing the direction of the arrows. After this transformation the resulting undirected graph will have the same joint probability as the Bayesian network, but it may no longer preserve all the conditional independence properties of the graph. For example, if we consider an HMM, we can get a corresponding undirected graphical model after the moralization that looks like a model in Figure 3.7 (c). The joint distribution encoded in this undirected model, according to the rules of graphical models, can be factored in the following way:

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{Y}_1 | \mathbf{X}_1) p(\mathbf{X}_1) \prod_{t=2}^T p(\mathbf{Y}_t | \mathbf{X}_t) p(\mathbf{X}_t | \mathbf{X}_{t-1}). \quad (3.20)$$

If we let

$$Z = 1 \quad (3.21)$$

$$\psi_{1,2}(\mathbf{X}_2, \mathbf{X}_1) = p(\mathbf{X}_2 | \mathbf{X}_1) p(\mathbf{X}_1) \quad (3.22)$$

$$\psi_{t,t+1} p(\mathbf{X}_{t+1}, \mathbf{X}_t) = p(\mathbf{X}_{t+1} | \mathbf{X}_t) \quad (3.23)$$

$$\phi_t(\mathbf{X}_t, \mathbf{Y}_t) = p(\mathbf{Y}_t | \mathbf{X}_t), \quad (3.24)$$

then the distribution encoded by the undirected graph is exactly the same as the one depicted by the directed model in Figure 3.4 (a). However, the undirected model is more general and allows for bi-directional potential functions. This is useful for batch estimation of the posterior, where one wants to ensure that the distribution over the state of \mathbf{X}_t is affected by the future state \mathbf{X}_{t+1} as well as the past \mathbf{X}_{t-1} .

3.3.3 Factor Graphs

A *factor graph* is an undirected graphical bipartite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{F}, \mathcal{E}\}$, where \mathcal{V} is a set of nodes associated with random variables $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{|\mathcal{V}|}\}$, \mathcal{F} is the set of function nodes, and \mathcal{E} is the set of undirected edges $(i, j) \in \mathcal{E}$ between factors $i \in \mathcal{F}$ and random variables $j \in \mathcal{V}$ on which they operate. For each node in the graph $A(i) \in \{\mathcal{V}, \mathcal{F}\}$ is the neighborhood operator that returns all nodes that are connected to the node i , where $i \in \{\mathcal{V}, \mathcal{F}\}$. Formally, $j \in A(i)$ if and only if there exist an edge $(i, j) \in \mathcal{E}$. Notice that due to the bipartite nature of the graph (see Figure 3.1) if $i \in \mathcal{F}$, then $A(i) \subset \mathcal{V}$, and vice versa. Hence the set of random variables that are associated with the nodes connected to a factor i , can be denoted

as $\mathbf{X}_{A(i)} = \{\mathbf{X}_j | j \in A(i)\} \subset \mathbf{X}$. Each function node $i \in \mathcal{F}$ in the graph has an associated real-valued compatibility function $\psi_i(\mathbf{X}_{A(i)})$ that operates on all the neighbors. Similar to the other graphical models we can easily write the joint distribution over all variables \mathbf{X} using the graphical model structure as follows,

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{i \in \mathcal{F}} \psi_i(\mathbf{X}_{A(i)}), \quad (3.25)$$

where Z is the partition function or the normalizing constant. In the cases where potential functions ψ_i are proper probability distributions, such explicit normalization is unnecessary. In general, these potential functions can be interpreted as local compatibility or constraints between random variables. It is worth mentioning that typically they do not correspond to the marginals $\psi_i(\mathbf{X}_{A(i)}) \neq p(\mathbf{X}_{A(i)})$.

Factor graphs are able to represent a richer set of graphical models, and most directed and undirected graphical models can be written in the factor graph form given the particular choice of potential functions. For example, Markov random fields can always be represented by a factor graph with one function node per clique in MRF (*a.k.a. clique hypergraph*).

3.4 Parameter Estimation

Given a known graphical model structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in most cases one must learn the parameters of the model denoted by θ . Below we discuss the two most popular algorithms for doing this: Maximum Likelihood Estimation (MLE) and Expectation-Maximization (EM).

3.4.1 Maximum Likelihood

Maximum likelihood estimation (MLE) is an approach for deriving estimates for parameters θ . The key idea in MLE is that the true estimate of the parameters, θ , is the one that makes the observed data under the model most likely. In other words, assuming that we have the right model, we should choose the parameters in such a way as to maximize our chance of producing the data that we already observed.

Assuming that we have a likelihood function $\mathcal{L}(\theta) = p(\mathcal{D}|\theta)$, we would like to maximize the probability of the set of observations $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ drawn from $p(\mathbf{X}|\theta)$. Notice that unlike in inference, where we assume that parameter vector θ is fixed and \mathbf{X} is a variable or a set of variables, here we are assuming the opposite. In particular, we fix our observations and search for parameters θ that best account for these observations. To this end, the maximum likelihood estimator for θ can be defined as follows:

$$\hat{\theta}_{ML} = \arg \max_{\theta} \mathcal{L}(\theta) = \arg \max_{\theta} p(\mathcal{D}|\theta) = \arg \max_{\theta} p(x_1, x_2, \dots, x_N|\theta) \quad (3.26)$$

In order to solve the equation above, we need to differentiate the likelihood function with respect to the parameter vector θ . Since often this likelihood function is in the exponential family, it is useful to first take the log of the likelihood function, resulting in the equivalent but more convenient form of,

$$\hat{\theta}_{ML} = \arg \max_{\theta} \ln \mathcal{L}(\theta) = \arg \max_{\theta} \ln p(\mathcal{D}|\theta) = \arg \max_{\theta} \ln p(x_1, x_2, \dots, x_N|\theta). \quad (3.27)$$

MLE has a number of nice asymptotic properties. For example, if one assumes that observations are independent and identically distributed (drawn with replacement from the target joint distribution), then it

can be shown that MLE estimator is *asymptotically optimal*. In other words, as the number of observations $N \rightarrow \infty$, the bias of MLE estimator tends to 0, resulting in an unbiased estimator with lowest possible mean squared error.

Maximum Likelihood for Multivariate Gaussian Distribution

MLE is often useful for estimating parameters of a parametric distribution. For example, in this section we will use MLE to estimate parameters of the distribution defined on a random variable \mathbf{X} , that is assumed to be distributed according to the multivariate d -dimensional Normal (*a.k.a.* Gaussian) distribution, $p(\mathbf{X}|\theta) = \mathcal{N}(\mathbf{X}|\mu, \Sigma)$. Given the form of the normal distribution,

$$p(\mathbf{X}|\theta) = p(\mathbf{X}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{X} - \mu)^T \Sigma^{-1}(\mathbf{X} - \mu)\right), \quad (3.28)$$

we can write the expression for the ML estimator for the mean, μ , based on N *i.i.d.* observations $\{x_1, x_2, \dots, x_N\}$ (where $x_i \sim p(\mathbf{X}|\theta)$) as follows,

$$\hat{\mu}_{ML} = \arg \max_{\mu} \ln p(x_1, x_2, \dots, x_N|\mu, \Sigma) = \quad (3.29)$$

$$= \arg \max_{\mu} \ln \prod_{i=1}^N \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right) = \quad (3.30)$$

$$= \arg \max_{\mu} \sum_{i=1}^N -\frac{1}{2} \ln [(2\pi)^d |\Sigma|] - \frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \quad (3.31)$$

now by taking the partial derivative with respect to μ and setting it equal to 0,

$$\frac{\partial}{\partial \mu} \sum_{i=1}^N -\frac{1}{2} \ln [(2\pi)^d |\Sigma|] - \frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) = 0 \quad (3.32)$$

$$\sum_{i=1}^N \Sigma^{-1}(x_i - \mu) = 0 \quad (3.33)$$

$$(3.34)$$

we can derive the following estimator for μ ,

$$\hat{\mu}_{ML} = \frac{1}{N-1} \sum_{i=1}^N x_i, \quad (3.35)$$

which corresponds to a sample mean. A similar exercise for covariance, Σ , would give us,

$$\hat{\Sigma}_{ML} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \hat{\mu}_{ML})^T (x_i - \hat{\mu}_{ML}). \quad (3.36)$$

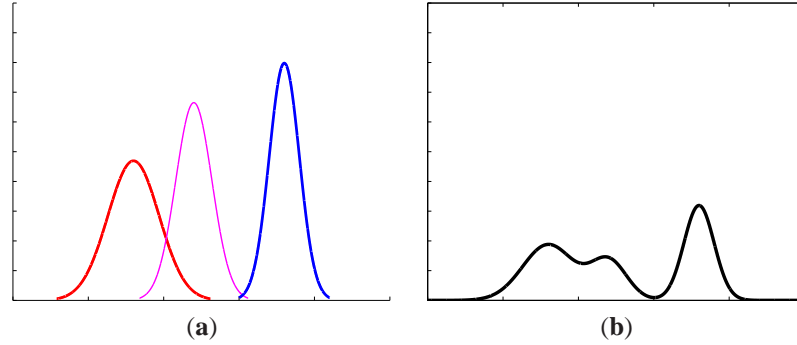


Figure 3.8: **Gaussian Mixture Model Illustration.** GMM model consisting of the three weighted Gaussian components (a) is illustrated in (b). The weight of components is associated with the line width in (a). Both red and blue components have a weight of $\delta_{red} = \delta_{blue} = 0.4$ and magenta component weight of $\delta_{magenta} = 0.2$.

3.4.2 Expectation-Maximization

When some of the variables in a graph cannot be directly observed (are hidden), doing ML estimation is often computationally very challenging since one needs to marginalize over all possible values of hidden variables,

$$\hat{\theta}_{ML} = \arg \max_{\theta} \ln \mathcal{L}(\theta) = \arg \max_{\theta} \ln \left[\sum_{\mathbf{X}} p(\mathbf{X}, \mathbf{Y} | \theta) \right], \quad (3.37)$$

denoted by \mathbf{X} . In most cases the resulting marginal likelihood distribution $\mathcal{L}(\theta)$ has a complex form leading to complicated expressions for the ML estimation. In such cases the *expectation-maximization* (EM) algorithm [48] is often used to obtain a local ML estimate [21, 48, 155]. In general, EM is guaranteed to converge to a local maximum (or saddle point) of the observed data likelihood function. EM could be thought of as a coordinate ascent on the likelihood function. Since the likelihood function is in general non-convex and contains many local maxima, the result of the EM often depends on good starting values. In practice, EM is often run a number of times (*a.k.a.* random restarts) with different initial conditions to avoid, at least in part, convergence to local maxima.

EM is an iterative procedure that alternates between performing an expectation (**E**) step, which computes an expectation of the likelihood by including the hidden variables as if they were observed, and a maximization (**M**) step, which computes the ML estimate of the parameters by maximizing the expected likelihood found in the E step. In particular the general EM algorithm can be outlined as follows,

E step: Complete expected value for the latent variables using posterior $\mathcal{L}(\mathbf{X} | \mathbf{Y}, \theta^{(k)})$, assuming that an estimate for the parameters, $\theta^{(k)}$, from the previous iteration is given.

M step: Estimate the new value for the parameters $\theta^{(k+1)}$ using the following equation,

$$\theta^{(k+1)} = \arg \max_{\theta^{(k+1)}} \sum_{\mathbf{X}} \mathcal{L}(\mathbf{X} | \mathbf{Y}, \theta^{(k)}) \ln \mathcal{L}(\mathbf{X}, \mathbf{Y}, \theta^{(k+1)}). \quad (3.38)$$

To bootstrap the algorithm above, an initial value for the parameters $\theta^{(0)}$ must be chosen (often at random).

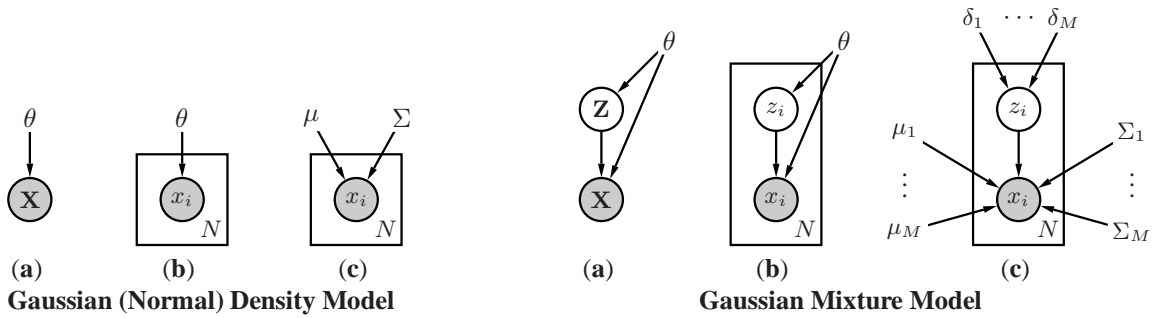


Figure 3.9: **Graphical Models for Gaussian and Gaussian Mixture Model.** In (a) graphical models for the Gaussian and Gaussian Mixture Model (GMM) are shown on the (left) and (right) respectively. In (b) a graphical representation of a Gaussian and GMM for a set of N *i.i.d.* samples x_i is shown. In the case of GMM (right) corresponding latent cluster labels z_i are also shown. In (b) we also introduce a new *plate* notation [32], denoted by the box with a label N . Using plates, N instances of the content in the box are represented compactly by the notation shown. Lastly in (c) all the parameters of the two models are explicitly shown, instead of a parameter vector θ . Notice that models depicted in (a) are useful for inference, and models in (b) and (c) for parameter estimation.

The algorithm above iterates until convergence, *i.e.* $\theta^{(k+1)} \approx \theta^{(k)}$.

Expectation-Maximization for Gaussian Mixture Model

A Gaussian or other unimodal distributions in the exponential family are often too restrictive to model realistic multi-modal data; a *Gaussian Mixture* is a convenient distribution for modeling such cases. A *Gaussian Mixture* is the model with M mixture components, as is shown in Figure 3.8, each of which in themselves are Gaussian. It is worth noting that Gaussian Mixture is not part of the exponential family of distributions introduced in Section 3.1.1. Nevertheless Gaussian mixture has a number of convenient properties² that are inherited from the Gaussian components. The model can be written as follows,

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}, \theta)p(\mathbf{Z}), \quad (3.39)$$

where \mathbf{Z} is a multinomial hidden indicator variable that tells which mixture component generated the observation \mathbf{X} . For a given value of the indicator $p(\mathbf{Z} = m)$, the observation has a normal distribution $\mathcal{N}(\mu_m, \Sigma_m)$, $m \in M$.

Alternatively this model can be written in the following form:

$$p(\mathbf{X}|\theta) = \sum_{m=1}^M \delta_m \mathcal{N}(\mathbf{X}|\mu_m, \Sigma_m), \quad (3.40)$$

where $\theta = \{\mu_1, \Sigma_1, \delta_1, \dots, \mu_M, \Sigma_M, \delta_M\}$ and $\sum_{m=1}^M \delta_m = \sum_{m=1}^M p(\mathbf{Z} = m) = 1$. Intuitively, the model tells us that the data is generated by first sampling $z_i \sim p(\mathbf{Z})$ and then given z_i sampling from the respective Normal mixture component. Since clearly we cannot observe z_i and are only able to observe the final $x_i \sim$

²Similar to the Gaussian, a product of Gaussian Mixtures is in itself a Gaussian Mixture. Also, the conditional distribution of two or more variables that jointly have a Gaussian Mixture form, is also a Gaussian Mixture.

<p>Input: number of mixture components, M, assumed in the model N <i>i.i.d.</i> observations $\{x_1, x_2, \dots, x_N\}$ drawn from $p(\mathbf{X})$ initial estimates for parameters $\theta^{(0)} = \{\theta_1^{(0)}, \dots, \theta_M^{(0)}\}$, where $\theta_m^{(0)} = \{\mu_m^{(0)}, \Sigma_m^{(0)}, \delta_m^{(0)}\}$, $m \in [1, \dots, M]$ (typically obtained using K-means)</p> <p>Output: estimates for parameters $\theta = \{\theta_1, \dots, \theta_M\}$, where $\theta_m = \{\mu_m, \Sigma_m, \delta_m\}$</p> <p>E step: Complete expected value for the latent variables $\{z_1, z_2, \dots, z_N\}$ using posterior $p(\mathbf{Z} \mathbf{X}, \theta^{(k)})$, assuming that an estimate for the parameters, $\theta^{(k)}$, from the previous iteration is given:</p> $p(z_n = m x_n, \theta_m) = \frac{p(x_n z_n = m, \theta_m)p(z_n = m)}{\sum_{i=0}^M p(x_n z_n = i, \theta_i)p(z_n = i)}, \quad (3.41)$ <p>where $n \in [1, \dots, N]$.</p> <p>M step: Estimate the new value for the parameters $\theta^{(k+1)}$ using the following equations,</p> $\theta^{(k+1)} = \arg \max_{\theta^{(k+1)}} \sum_{\mathbf{Z}} p(\mathbf{Z} \mathbf{X}, \theta^{(k)}) \ln p(\mathbf{X} \mathbf{Z}, \theta^{(k+1)}), \quad (3.42)$ <p>which translates to the following set of equations for individual parameters:</p> $\delta_m^{(k+1)} = \frac{1}{N} \sum_{n=1}^N p(z_n = m) \quad (3.43)$ $\mu_m^{(k+1)} = \frac{\sum_{n=1}^N p(z_n = m)x_n}{\sum_{n=1}^N p(z_n = m)} \quad (3.44)$ $\Sigma_m^{(k+1)} = \frac{\sum_{n=1}^N p(z_n = m)(x_n - \mu_m^{(k+1)})(x_n - \mu_m^{(k+1)})^T}{\sum_{n=1}^N p(z_n = m)} \quad (3.45)$ <p>Loop: Repeat above for a fixed number of iterations, K, or until $\theta^{(k+1)} \approx \theta^{(k)}$ (up to some chosen precision ϵ), then let $\theta = \theta^{(k+1)}$.</p>

Algorithm 1: Expectation-Maximization for Gaussian Mixture Model. Iterative algorithm for obtaining estimates for the parameters of the Gaussian Mixture Model (GMM).

$p(X|Z = z_i, \theta)$, doing direct Maximum Likelihood estimation of the parameters becomes very complicated. Using EM, however, parameters can easily be derived using the iterative approach described in Algorithm 1.

3.4.3 Parameter Estimation with Hyperpriors

So far we have made an explicit distinction between parameters of the model and the variables associated with the model (see Figure 3.10 (a)). However, there is nothing special about the parameters and we can treat them as latent random variables themselves (see Figure 3.10 (b)) [19, 137]. In a Bayesian sense, the parameters can be reinterpreted as variables and have priors imposed on them much like the variables themselves do (see Figure 3.10 (c)). These priors are often called *hyperpriors*. This was already alluded to in Section 3.2.3. For example, in the case of Gaussian Mixtures it is often convenient to impose a symmetric Dirichlet prior with concentration parameter α/M on the mixture weights,

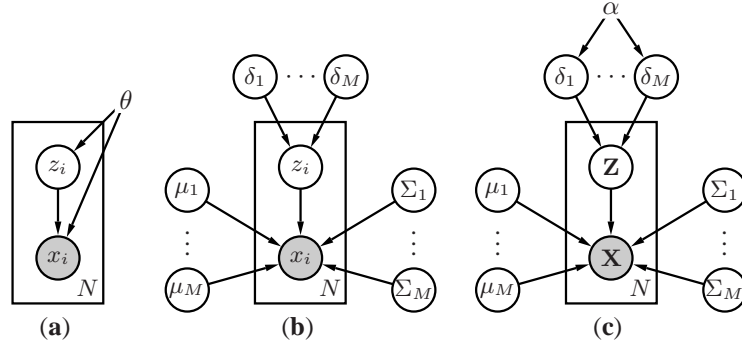


Figure 3.10: **Gaussian Mixture Model with Hyperpriors.** Graphical model representation for N *i.i.d.* samples drawn from the Gaussian Mixture Model (GMM) is shown in (a). In (b) the same model is shown where the parameters of the model are treated as variables themselves, on which a *hyperprior* with *hyperparameters* α is imposed in (c).

$$p(\delta_1, \delta_2, \dots, \delta_M | \alpha) \sim \text{Dir}(\alpha/M, \dots, \alpha/M) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/M)^M} \prod_{m=1}^M \delta_m^{\alpha/2-1}. \quad (3.46)$$

By letting $M \rightarrow \infty$ we get an infinite model that allows inference over the number of mixtures in addition to the parameters of the mixtures themselves. In fact, in the Bayesian sense the parameters are just nuisance variables that should be integrated out. For details on the use of hyperpriors in graphical models we refer reader to [107].

3.5 Inference

Given a graphical model encoded by the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a set of known (or estimated) parameters θ , typically one is interested in inferring the posterior distribution $p(\mathbf{X} | \mathbf{Y}, \theta)$ from the joint $p(\mathbf{X}, \mathbf{Y} | \theta)$ distribution encoded by the model, where \mathbf{X} is the set of hidden or latent variables and \mathbf{Y} is the set of observed variables. Sometimes, we are only interested in a subset of variables $\mathbf{X}_U \subset \mathbf{X}$, in which case only the *marginals*,

$$p(\mathbf{X}_U) = \int_{\mathbf{X} \setminus \mathbf{X}_U} p(\mathbf{X} | \mathbf{Y}, \theta) d\mathbf{X} \setminus \mathbf{X}_U \quad \text{or} \quad p(\mathbf{X}_U) = \sum_{\mathbf{X} \setminus \mathbf{X}_U} p(\mathbf{X} | \mathbf{Y}, \theta) \quad (3.47)$$

(depending on whether the variables are continuous or discrete) are needed.

In fact in most situations computing the full posterior $p(\mathbf{X} | \mathbf{Y}, \theta)$ is prohibitively expensive, and marginals are computed and used as a summary of the posterior instead. Typically it suffices to estimate the marginals for all or some subset of variables. For example, given a joint distribution $p(\mathbf{X}) = p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5)$ encoded in directed graphical model illustrated in Figure 3.2, compute $p(\mathbf{X}_1)$. Alternatively, we may want to compute all marginals, $p(\mathbf{X}_i)$, $i \in [1, \dots, 5]$. Using the marginals we can also easily compute conditional distributions of the form $p(\mathbf{X}_U | \mathbf{X} \setminus \mathbf{X}_U, \mathbf{Y}, \theta)$, where \mathbf{X}_U is as before subset of variable that are of interest, and $\mathbf{X} \setminus \mathbf{X}_U$ are all hidden variables excluding those in \mathbf{X}_U .

In this section we will discuss some approaches for doing inference in various graphical models introduced thus far. In particular, we will discuss the Elimination algorithm and the Belief Propagation (specifically the sum-product) algorithm. For additional information on other popular approaches to inference that are not covered in the thesis (*e.g.* junction-tree algorithm [26, 105, 246], mean field [68, 254], variational inference [109, 238]) we refer reader to the literature.

3.5.1 Variable Elimination

Let us consider the directed graph in Figure 3.2, that has the joint distribution

$$p(\mathbf{X}) = p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5) = p(\mathbf{X}_5|\mathbf{X}_3)p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1). \quad (3.48)$$

Let us further assume that all variables in the graph $\mathbf{X}_i, i \in [1, 5]$ are discrete with L possible states, and we want to compute the marginal $p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) = \sum_{\mathbf{X}_5} p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5)$. Doing this explicitly will result in the computation time that has complexity exponential in the number of variables in the joint distribution, $O(L^5)$. However, by taking advantage of conditional independence properties encoded by the graph structure and distributing the sum by moving it all the way in,

$$p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) = p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1) \sum_{\mathbf{X}_5} p(\mathbf{X}_5|\mathbf{X}_3), \quad (3.49)$$

we get an algorithm that has complexity that is quadratic in the number of discrete states, $O(L^2)$. The *Variable Elimination* (VE) algorithm [47, 258] does precisely this by rearranging terms of the product making up the joint distribution and moving the summations as far as possible inward. It is also convenient to introduce variables $m_i(\mathbf{X}_{S_i})$, where $S_i \subset \mathcal{V}$ corresponds to variables that appear in the sum but are not being summed over. These intermediate variables are called *messages*. For example, if we want to compute marginal $p(\mathbf{X}_1)$ for the same graph in Figure 3.2:

$$\begin{aligned} p(\mathbf{X}_1) &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \sum_{\mathbf{X}_5} p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5) \\ &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \sum_{\mathbf{X}_5} p(\mathbf{X}_5|\mathbf{X}_3)p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1) \\ &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1) \underbrace{\sum_{\mathbf{X}_5} p(\mathbf{X}_5|\mathbf{X}_3)}_{m_5(\mathbf{X}_3)} \\ &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1)m_5(\mathbf{X}_3) \\ &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1)m_5(\mathbf{X}_3) \underbrace{\sum_{\mathbf{X}_4} p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)}_{m_4(\mathbf{X}_2, \mathbf{X}_3)} \\ &= \sum_{\mathbf{X}_2} \sum_{\mathbf{X}_3} p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1)m_5(\mathbf{X}_3)m_4(\mathbf{X}_2, \mathbf{X}_3) \end{aligned}$$

$$\begin{aligned}
&= \sum_{\mathbf{X}_3} p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_1)m_5(\mathbf{X}_3) \underbrace{\sum_{\mathbf{X}_2} m_4(\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_2|\mathbf{X}_1)}_{m_2(\mathbf{X}_1, \mathbf{X}_3)} \\
&= \sum_{\mathbf{X}_3} p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_1)m_5(\mathbf{X}_3)m_2(\mathbf{X}_1, \mathbf{X}_3) \\
&= p(\mathbf{X}_1) \underbrace{\sum_{\mathbf{X}_3} m_5(\mathbf{X}_3)m_2(\mathbf{X}_1, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)}_{m_3(\mathbf{X}_1)} \\
&= p(\mathbf{X}_1)m_3(\mathbf{X}_1).
\end{aligned}$$

Notice that the complexity of the algorithm is governed by the maximum message size. The algorithm described above is called *Variable Elimination*, in essence, because it eliminates one variable at the time from the graph, until the graph corresponding to the marginal is left. Notice that while the order of the elimination is not unique³ there is an overall flow that can be established in that we first must eliminate children nodes (nodes that have no outgoing edges), then their parents and so on.

Notice that a similar elimination procedure can be done if \mathbf{X}_i , $i \in [1, \dots, 5]$ are continuous and not discrete, in which case the sums in the above formulation will be replaced by the integrals over the continuous random variables. For example,

$$\begin{aligned}
p(\mathbf{X}_1) &= \int_{\mathbf{X}_2} \int_{\mathbf{X}_3} \int_{\mathbf{X}_4} \int_{\mathbf{X}_5} p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5) d\mathbf{X}_2 d\mathbf{X}_3 d\mathbf{X}_4 d\mathbf{X}_5 \\
&= \int_{\mathbf{X}_2} \int_{\mathbf{X}_3} \int_{\mathbf{X}_4} \int_{\mathbf{X}_5} p(\mathbf{X}_5|\mathbf{X}_3)p(\mathbf{X}_4|\mathbf{X}_2, \mathbf{X}_3)p(\mathbf{X}_3|\mathbf{X}_1)p(\mathbf{X}_2|\mathbf{X}_1)p(\mathbf{X}_1) d\mathbf{X}_2 d\mathbf{X}_3 d\mathbf{X}_4 d\mathbf{X}_5.
\end{aligned}$$

Computing marginals in the graph where variables are both discrete and continuous would involve integrating and summing over the respective variables. So far we have only considered directed graphical models, but the Elimination algorithm works for undirected graphs as well in a similar manner. Consider computing the marginal $p(\mathbf{X}_2)$ for the graph illustrated in Figure 3.6. Similar to above,

$$\begin{aligned}
p(\mathbf{X}_2) &= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \sum_{\mathbf{X}_5} p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5) \\
&= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \sum_{\mathbf{X}_5} \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) \psi_{35}(\mathbf{X}_3, \mathbf{X}_5) \\
&= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) \underbrace{\sum_{\mathbf{X}_5} \psi_{35}(\mathbf{X}_3, \mathbf{X}_5)}_{m_5(\mathbf{X}_3)} \\
&= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \sum_{\mathbf{X}_4} \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) m_5(\mathbf{X}_3)
\end{aligned}$$

³In the example discussed, $\sum_{\mathbf{X}_4}$ can be done before $\sum_{\mathbf{X}_5}$, or the $\sum_{\mathbf{X}_2}$ before $\sum_{\mathbf{X}_3}$, without effecting the complexity.

$$\begin{aligned}
&= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) m_5(\mathbf{X}_3) \underbrace{\sum_{\mathbf{X}_4} \psi_{234}(\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4)}_{m_4(\mathbf{X}_2, \mathbf{X}_3)} \\
&= \sum_{\mathbf{X}_1} \sum_{\mathbf{X}_3} \frac{1}{Z} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) m_5(\mathbf{X}_3) m_4(\mathbf{X}_2, \mathbf{X}_3) \\
&= \sum_{\mathbf{X}_3} \frac{1}{Z} m_5(\mathbf{X}_3) m_4(\mathbf{X}_2, \mathbf{X}_3) \underbrace{\sum_{\mathbf{X}_1} \psi_{123}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)}_{m_1(\mathbf{X}_2, \mathbf{X}_3)} \\
&= \sum_{\mathbf{X}_3} \frac{1}{Z} m_5(\mathbf{X}_3) m_4(\mathbf{X}_2, \mathbf{X}_3) m_1(\mathbf{X}_2, \mathbf{X}_3) \\
&= \frac{1}{Z} \underbrace{\sum_{\mathbf{X}_3} m_5(\mathbf{X}_3) m_4(\mathbf{X}_2, \mathbf{X}_3) m_1(\mathbf{X}_2, \mathbf{X}_3)}_{m_3(\mathbf{X}_2)} \\
&= \frac{1}{Z} m_3(\mathbf{X}_2).
\end{aligned}$$

The key observation with undirected variant is that the normalizing constant, Z , can be factored out in all but the last step. This significantly simplifies computation of Z , that is typically unknown. By delaying the computation of Z to the very end, we can compute it by summing over a single variable $Z = \sum_{\mathbf{X}_2} m_3(\mathbf{X}_2)$; computing it beforehand would result in summation over all variables.

One shortcoming of the Elimination algorithm is that while it is efficient for computing single marginals, it is inefficient for computing marginals over all the variables. The reason for this is that it requires re-computation of the sums (or messages) for every marginal. However, it is easy to see that these messages will always be the same (though for an individual marginal not all messages may be required to be computed). Reusing these messages is essential in tractable computation of an arbitrary set of marginals. This is the premise behind the Belief Propagation algorithm outlined in the next section.

3.5.2 Belief Propagation

Belief Propagation (BP) is a popular inference algorithm for computing marginals of functions on undirected graphical models. BP is an instance of the more general sum-product algorithm that operates on factor graphs [119]. It can be proved that BP is guaranteed to converge to the exact marginals on tree-structured graphs [108]. In graphs that contain cycles BP, often in this case referred to as *Loopy Belief Propagation* (LBP), can lead to a tractable approximation to the marginals (exact inference is NP-hard [43]). LBP is not guaranteed to converge, however, and in case of convergence will only converge to a fixed point (not necessarily corresponding to a true marginal). It can be shown that the fixed point of LBP is equivalent to the stationary point of the Bethe approximation of the free energy [255], hence LBP will always lead to a lower energy state. In practice, LBP is widely used and has excellent empirical performance in many applications [221]. Most BP algorithms in the literature have concentrated on the models where variables corresponding to the nodes in the graph are discrete, however recently, attempts have been made in proposing approximate inference algorithms that can deal with continuous-state graphs of arbitrary topology [99, 220]. Table 3.1 outlines the various flavors of Belief Propagation algorithms, the details of which will be discussed in the following sections.

	Discrete-state		Continues-state			
	Tree	Loopy Graph (LBP)	Tree	Loopy Graph	Tree	Loopy Graph
Joint, $p(\mathbf{X})$	$= \frac{1}{Z} \prod_{(i,j) \in A} \psi_{ij}(\mathbf{X}_j, \mathbf{X}_i) \prod_{i \in [1, \dots, N]} \phi_i(\mathbf{X}_i)$					
Message, $m(X_i)$	$= \frac{1}{Z} \sum_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i)$		$= \frac{1}{Z} \int_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i) d\mathbf{X}_i$			
Marginal, $b(X_i)$	$= \frac{1}{Z} \phi_i(\mathbf{X}_i) \prod_{k \in A(i)} m_{ki}(\mathbf{X}_i)$					
Estimated Marginals $b(X_i)$	exact	approximate	exact	exact	approximate	approximate
Representation of $m(X_i)$	exact	exact	exact	exact	approximate	approximate
Complexity	$O(NL^2) / O(NL)$	$O(NL^C)$	$O(N)$		$O(NDM^2)$	
	N – Number of nodes in a graph L – Number of discrete states C – Size of the largest <i>clique</i> , defined as the largest set of fully connected nodes, in the graph. D – Largest <i>degree</i> , defined as the number of edges incident on the node, of the node in a graph. M – Number of components required to represent the message.					

Table 3.1: **Inference using Belief Propagation.** Summary of the known BP algorithm variants with complexity and known theoretical limitations. We will use the continuous-state Non-parametric Belief Propagation (NBP) approach of [99] on loopy-graphs designated in bold. For further description, including description of equations, please see text.

Discrete Belief Propagation

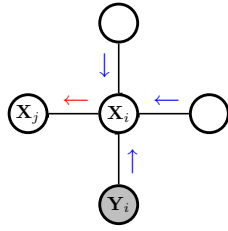
Belief propagation can, in general, be introduced in the context of the pair-wise MRF formulation [253] of Section 3.3.2. Consider a set of latent (*a.k.a.* hidden) variable nodes $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ and a corresponding set of observed nodes $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N\}$. Please note that 1-to-1 correspondence of the latent and observation nodes is simply for notational convenience and is not required by the framework or inference algorithm. The conditional independence of the latent variables is expressed by a neighborhood set A . A pair of node indices $(i, j) \in A$ if the node \mathbf{X}_j is not conditionally independent of \mathbf{X}_i given all other nodes in the graph. For notational simplicity we will define a function $A(i)$ that will return all neighbors of i . More formally $j \in A(i) \iff (i, j) \in A$. When \mathbf{X}_i are discrete random variables, we can assume, without loss of generality, that they can take on some value $x_i \in [1, 2, \dots, L]$. The observation and hidden nodes are related by the real-valued observation (or likelihood) function $\phi_i(\mathbf{X}_i, \mathbf{Y}_i) \equiv \phi_i(\mathbf{Y}_i | \mathbf{X}_i) \equiv \phi_i(\mathbf{X}_i)$; connected hidden nodes by a potential (or correlation) function $\psi_{ij}(\mathbf{X}_j, \mathbf{X}_i)$. The joint probability over $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ can then be written as:

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{(i,j) \in A} \psi_{ij}(\mathbf{X}_j, \mathbf{X}_i) \prod_{i \in [1, \dots, N]} \phi_i(\mathbf{X}_i), \quad (3.50)$$

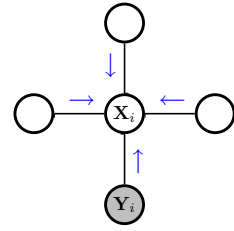
where Z is a normalizing constant that ensures that $p(\mathbf{X})$ integrates to 1.

A brute force inference algorithm that simply enumerates all possible states for \mathbf{X} and evaluates $p(\mathbf{X})$, would lead to $O(L^N)$ run-time (as was already discussed in Section 3.5.1), which is infeasible even for small values of L and N . BP that exploits the conditional independence structure of the graphical model would lead to a solution, that allows computation of arbitrary sub-set of marginals, in $O(NL^C)$, where $C \ll N$. The BP algorithm operates in two stages: **(1)** it introduces auxiliary random variables $m_{ij}(\mathbf{X}_j)$ that can be intuitively understood as *messages* from hidden node i to node j about what state node j should be in, and **(2)** computes the approximation to the marginal distribution of \mathbf{X}_i (often referred to as the belief). Messages are computed iteratively using the equation below

$$m_{ij}(\mathbf{X}_j) \propto \sum_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i), \quad (3.51)$$

Message Passing

$$m_{ij}(\mathbf{X}_j) \propto \sum_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i)$$

Belief Estimation

$$b_i(\mathbf{X}_i) \propto \phi_i(\mathbf{X}_i) \prod_{k \in A(i)} m_{ki}(\mathbf{X}_i)$$

Figure 3.11: **Belief Propagation.** On the left a new outgoing message, in red, is computed from local image evidence and incoming messages from neighboring nodes, in blue. On the right approximate marginal densities are determined from the normalized product of the local observations with messages sent from all neighboring nodes.

and beliefs, where required, are given by

$$b_i(\mathbf{X}_i) \propto \phi_i(\mathbf{X}_i) \prod_{k \in A(i)} m_{ki}(\mathbf{X}_i). \quad (3.52)$$

It should be noted that in the case of tree structured graphs each message is only needs to be sent/computed once. The algorithm will first send messages from all leaf nodes to the vertices adjacent to the respective leaves and then will continue sending messages up to the root and then back down towards the leaves until all messages have been sent exactly once. In this case of the describe tree-structured graphs, BP sometimes is also referred to as the Viterbi algorithm [234] or forward-backward algorithm. Consequently, in the case of the loopy graphs the algorithm needs to iterate and send a complete set of messages a number of times. In the case of the loopy graph the order in which messages should be sent is not explicitly defined by the graph, hence a schedule according to which messages should be sent has to be defined.

The complexity of the proposed discrete BP algorithm is $O(NL^2)$ for tree structured models and $O(NL^C)$ for loopy graphs, where N is the number of nodes in the graph, L a fixed number of discrete states each \mathbf{X}_i can assume, and C is the size of the maximal clique in the graph. Hence, it is only tractable with relatively few states L . Recently, it has been shown [59] that for tree structured models with a particular restrictive choice of $\psi_{ij}(\mathbf{X}_j, \mathbf{X}_i) = \mathcal{N}(\mathbf{X}_i - \mathbf{X}_j | \mu_{ij}, \Sigma_{ij})$ the inference can be done in $O(NL)$. This allows tractable inference in models that contain on the order of 1 million states.

Discrete Belief Propagation for Articulated Pose Inference

One application of discrete pair-wise MRFs and BP that is directly related to this thesis, was introduced by Felzenszwalb *et al.* in [59] and later extended in [121, 122]. The key idea in these methods is to model an articulated structure (a person) in 2D using a discrete graphical model and formulate the articulated pose estimation as inference in this graphical model, computed using BP. In this graphical model each node, i , in the graph corresponds to the body part and the corresponding discrete random variable, \mathbf{X}_i , to the pose of this body part in the image space. The edges in the graph encode the kinematic and joint constraints, ensuring that the parts are loosely connected. The pose for each part is encoded using a state vector $\in \mathbb{R}^4$, allowing to account for x - and y - positions, scale, and rotation of the part in the image. In [59, 122] the state of each

variable in the graph \mathbf{X}_i , was discretized to 70 x - and y - positions, 10 scales and 32 rotations, leading to a total of $70 \times 70 \times 10 \times 32 = 1,568,000$ possible discrete states for each variable \mathbf{X}_i . Even with such relatively coarse discretization, that can lead to as much as 3-pixel and 5.6-degree quantization error for a typical 640×480 image, inference using BP in a graph of general topology with unconstrained choice for the potentials is impractical (see Table 3.1). To ensure that inference is computationally tractable, tree-structure for the graphical model was assumed [59]. This leads, in general, to quadratic inference complexity in the number of states. However, with such a large state space even this is impractical. The restrictive choice of the potential functions [59], $\psi_{ij}(\mathbf{X}_j, \mathbf{X}_i) = \mathcal{N}(\mathbf{X}_i - \mathbf{X}_j; \mu_{ij}, \Sigma_{ij})$, however, led to tractable linear inference in $O(NL)$.

In this thesis we consider a more general case of this model, by modeling the body using a richer continuous state graphical model. Notice, that in the case of the 3D pose (being addressed in this thesis) discretization of the state $\mathbf{X}_i \in \mathbb{R}^6$, similar to the one introduced in [59], would lead to $\approx 1.721 \times 10^{13}$ states rendering even linear complexity discrete BP methods intractable. Furthermore, the imposed tree structure requirement of [59, 122] does not allow models that incorporate natural constraints such as those related to inter-penetrations or occlusions. Incorporating such constraints would lead to loops in the graphical model structure. The benefit of these richer constraints will be discussed in detail in Chapters 5 and 6. The required choice of $\psi_{ij}(\mathbf{X}_j, \mathbf{X}_i)$ is also too restrictive to model realistic statistical relationships between variables. For example, joint limits are hard to model using these simple Gaussian distributions. In this thesis we introduce a richer class of models that are able to address these issues.

Continuous Belief Propagation

When \mathbf{X}_i 's are continuous random variables as is true for our case, the equation for the message (Eq. 3.51) must be rewritten as follows,

$$m_{ij}(\mathbf{X}_j) \propto \int_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i) d\mathbf{X}_i, \quad (3.53)$$

by replacing the sum with an integral over \mathbf{X}_i . Note that the joint distribution $p(\mathbf{X})$ and belief $b(\mathbf{X}_i)$ in the continuous case can be written identically to discrete version above (see Eq. 3.50 and Eq. 3.52 respectively). If $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ and $\phi_i(\mathbf{X}_i)$ are both Gaussian then the marginal distribution at each node is also Gaussian (regardless of the graph topology) and the integration can be performed exactly [244]. However, this case is restrictive and uncommon. In most vision problems both the likelihoods and potentials are multi-modal and non-Gaussian. For simplicity and without loss of generality let us assume that we can model $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ and $\phi_i(\mathbf{X}_i)$ in such cases using a Gaussian mixture model. It can then be shown that the representation required to represent the messages and the marginals in this case grows exponentially (and in the case of the loopy-graphs without bound [117]) and has to be approximated to produce tractable inference algorithms. This gives rise to what are called *Non-parametric Belief Propagation* (NBP) algorithms that tend to approximate messages using fixed-length kernel densities and integrals by Monte-Carlo integration. We will describe the extension to the variant of NBP first introduced in [99], called Particle Message Passing (PAMPAS), in Section 3.7. However, to do so, we must first introduce the class of Monte Carlo methods.

3.6 Monte Carlo Methods

In many cases inference and learning approaches introduced in Sections 3.5.1 and 3.5.2 are intractable, especially in the cases of continuous variables, and complex multi-modal distributions. *Monte Carlo* (MC) methods [138, 175], introduced as early as 1949 by Metropolis and Ulam [141], provide a numeric approximation to these tasks by using samples of densities instead of densities themselves. In principle MC approximations can be shown to lead to exact solutions as the number of samples $N \rightarrow \infty$. In practice, computational resources often require inference using a relatively small number of samples, in which cases the success of the MC method depends on the efficiency of the designed sampling scheme.

The key observation, is that many inference tasks over continuous variables can be expressed as the expectation of some appropriately chosen function $f(\mathbf{X})$, $\mathbb{E}[f(\mathbf{X})]$, such that

$$\mathbb{E}[f(\mathbf{X})] = \int_{\mathbf{X}} f(\mathbf{X})p(\mathbf{X}) d\mathbf{X} \quad (3.54)$$

where $p(\mathbf{X})$, $\mathbf{X} \in \mathbb{R}^d$, is the target density we are trying to approximate. If we approximate $p(\mathbf{X})$ using N independent weighted samples $\{s^{(n)}, w^{(n)} | n \in [1, \dots, N]\}$, where the $\sum_{n=1}^N w^{(n)} = 1$, then we can write

$$\mathbb{E}[f(\mathbf{X})] = \int_{\mathbf{X}} f(\mathbf{X})p(\mathbf{X}) d\mathbf{X} \approx \sum_{n=1}^N w^{(n)} f(s^{(n)}). \quad (3.55)$$

3.6.1 Importance Sampling

The basic MC approximation assumes that we can sample from the target distribution, $s^{(n)} \sim p(\mathbf{X})$, in which case $w^{(n)} = 1/N$. In most cases, in particular in most vision applications, this is typically intractable. *Importance sampling* [214] can be used in such cases to facilitate the inference. In particular, let us assume we have a *proposal* distribution $q(\mathbf{X})$ that is easy to sample. The expectation can then be re-written as,

$$\mathbb{E}[f(\mathbf{X})] \approx \sum_{n=1}^N w^{(n)} f(s^{(n)}) = \sum_{n=1}^N w^{(n)} \frac{f(\tilde{s}^{(n)})p(\tilde{s}^{(n)})}{q(\tilde{s}^{(n)})}, \quad (3.56)$$

where $\tilde{s}^{(n)} \sim q(\mathbf{X})$. The equation simplifies to

$$\mathbb{E}[f(\mathbf{X})] \approx \sum_{n=1}^N \tilde{w}^{(n)} f(\tilde{s}^{(n)}) \quad (3.57)$$

if we let

$$\tilde{w}^{(n)} = \frac{1}{Z} w^{(n)} \frac{p(\tilde{s}^{(n)})}{q(\tilde{s}^{(n)})}, \quad Z = \sum_{i=1}^N w^{(i)} \frac{p(\tilde{s}^{(i)})}{q(\tilde{s}^{(i)})}. \quad (3.58)$$

Hence, importance sampling estimates the target expectation via a collection of weighted samples from the proposal density $\{\tilde{s}^{(n)}, \tilde{w}^{(n)} | n \in [1, \dots, N]\}$. The choice of the importance function $q(\mathbf{X})$ will dictate the effectiveness of the proposed approximation. Designing good proposal functions is critical for tractable inference. Building good proposal functions, however, is hard; particularly in high-dimensional state spaces.

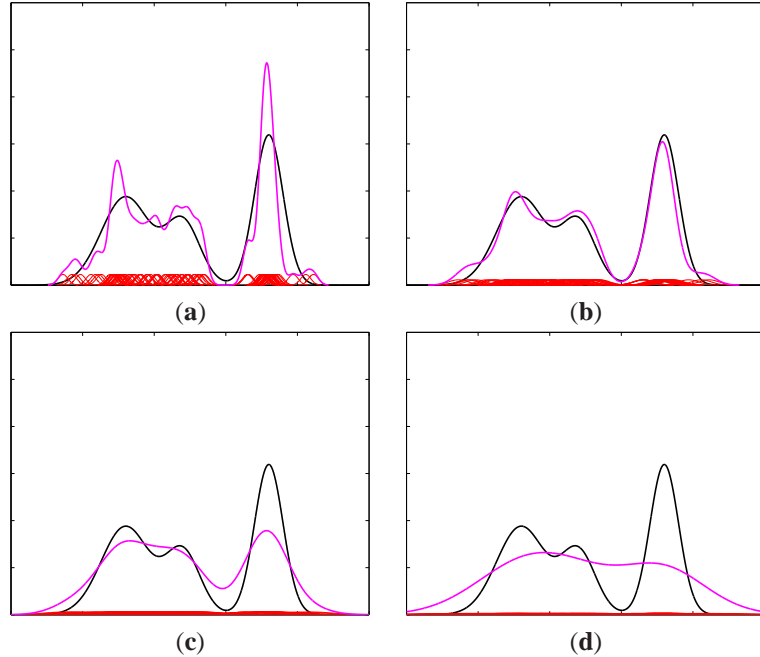


Figure 3.12: **Kernel density bandwidth estimation.** The effect of bandwidth in Kernel Density Estimation (KDE) is illustrated. The target distribution and KDE approximation based on the same 100 Gaussian kernels are shown in (black) and (magenta) respectively. The low bandwidth (a) leads to erratic peaks in the approximated density; high bandwidth (d) leads to over-smoothing. Appropriate bandwidth leads to good approximation of the density (b). The *rule-of-thumb* bandwidth estimate is shown in (c).

3.6.2 Kernel Density Estimation

Monte Carlo methods give a tractable solution to computing the expectations, but do not provide a sensible way of estimating the target density $p(\mathbf{X})$. In particular in MC methods, the target density is approximated using a weighted mixture of Dirac delta functions,

$$p(\mathbf{X}) = \sum_{i=1}^N w^{(i)} \delta(s^{(i)} - \mathbf{X}) \quad \sum_{i=1}^N w^{(i)} = 1. \quad (3.59)$$

In some cases a continuous estimate of the target density would be preferred. One way this can be achieved is by fitting a parametric density function to the samples, however, this requires knowledge of the structure of the underlying density function. Furthermore, the number of samples is often too few to robustly fit complex parametric densities. One alternative, is to use nonparametric density estimation methods [94, 202], that smooth the raw sample set with a *kernel* function of choice. This intuitively places more probability mass in the regions that contain many particles with high weight. A frequent choice for a kernel function is a Gaussian. Given a Gaussian kernel, a *Kernel Density Estimate* (KDE) of the target density $p(\mathbf{X})$ can be written as a Gaussian Mixture,

$$p(\mathbf{X}) = \sum_{i=1}^N w^{(i)} \mathcal{N}(\mathbf{X}|s^{(i)}, \Sigma^{(i)}) \quad \sum_{i=1}^N w^{(i)} = 1. \quad (3.60)$$

with *bandwidth* (in this case corresponding to covariance matrix) $\Sigma^{(i)}$. The results of the KDE estimation can

	Kernel Function	Rule-of-thumb Bandwidth Estimator
Gaussian	$\mathcal{K}(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$	$\Lambda = 1.06\sigma N^{-1/5}$
Uniform	$\mathcal{K}(u) = \frac{1}{2}I(u \leq 1)$	$\Lambda = 1.84\sigma N^{-1/5}$
Epanechnikov	$\mathcal{K}(u) = \frac{3}{4}(1 - u^2)I(u \leq 1)$	$\Lambda = 2.35\sigma N^{-1/5}$
Quartic	$\mathcal{K}(u) = \frac{15}{16}(1 - u^2)I(u \leq 1)$	$\Lambda = 2.78\sigma N^{-1/5}$

Table 3.2: **Kernel density estimators.** Kernel function for various kernel estimators are listed, along with the corresponding *rule-of-thumb* estimators for bandwidth Λ . N in the above equations is the number of samples in the kernel density estimate, and σ is the standard deviation of the samples (similar formulas can be derived for multivariate kernel estimators).

be seen in Figures 3.13 and 3.14. The quality of the kernel estimate depends on the bandwidth parameter, that must be estimated based on the density of samples. Intuitively if the samples are dense, not much smoothing is needed to get a smooth and continuous estimate for the target density. If the samples are sparse, however, then small bandwidth would lead to erratic peaks and hence a larger bandwidth is preferred. This is illustrated in Figure 3.12. Typically a single bandwidth is selected for all the kernels, *i.e.* $\Sigma^{(i)} = \Lambda, \forall i \in [1, \dots, N]$. However, variable bandwidth methods exist, that estimate different bandwidth for different portions of the space based on the local (instead of global) density of samples [41].

The extensive body of literature exists on automatic bandwidth selection [41, 94, 186, 202]. One of the simplest methods is Silverman's [202] *rule-of-thumb* bandwidth estimator, that combines covariance estimation with an asymptotic formula; derived assuming Gaussian form for the target density. Similar formula can be computed for other choices of a kernel function. While rule-of-thumb gives a convenient simple form for the bandwidth estimation, it often oversmooths the target distribution, particularly in the cases where it is multi-modal.

While we only introduced a Gaussian kernel thus far, other kernels can be used and may be appropriate depending on the form of target density. In particular, the formulation in Eq. 3.60 can be generalized as follows,

$$p(\mathbf{X}) = \sum_{i=1}^N w^{(i)} \mathcal{K}\left(\frac{\mathbf{X} - s^{(i)}}{\Sigma^{(i)}}\right) \quad (3.61)$$

where \mathcal{K} is the kernel function and $\Sigma^{(i)}$ is as before a bandwidth parameter. Various popular choices for the kernel function and the rule-of-thumb estimates for the common bandwidth $\Lambda, \Sigma^{(i)} = \Lambda, \forall i \in [1, \dots, N]$, are given in Table 3.2.

3.6.3 Markov Chain Monte Carlo

When a target density $p(\mathbf{X})$ is complex and high-dimensional, often obtaining an importance function $q(\mathbf{X})$ that is both close to the target density (for efficiency) and easy to sample from (for computational tractability) is hard. To address this, *Markov Chain Monte Carlo* (MCMC) methods provide an alternative solution that allows one to draw approximate samples from the target density explicitly. The idea in all MCMC methods is to construct a Markov Chain (previously discussed in Section 3.2.1) that has the same stationary distribution as the desired density $p(\mathbf{X})$. The samples can then be drawn from the density by simulating the Markov

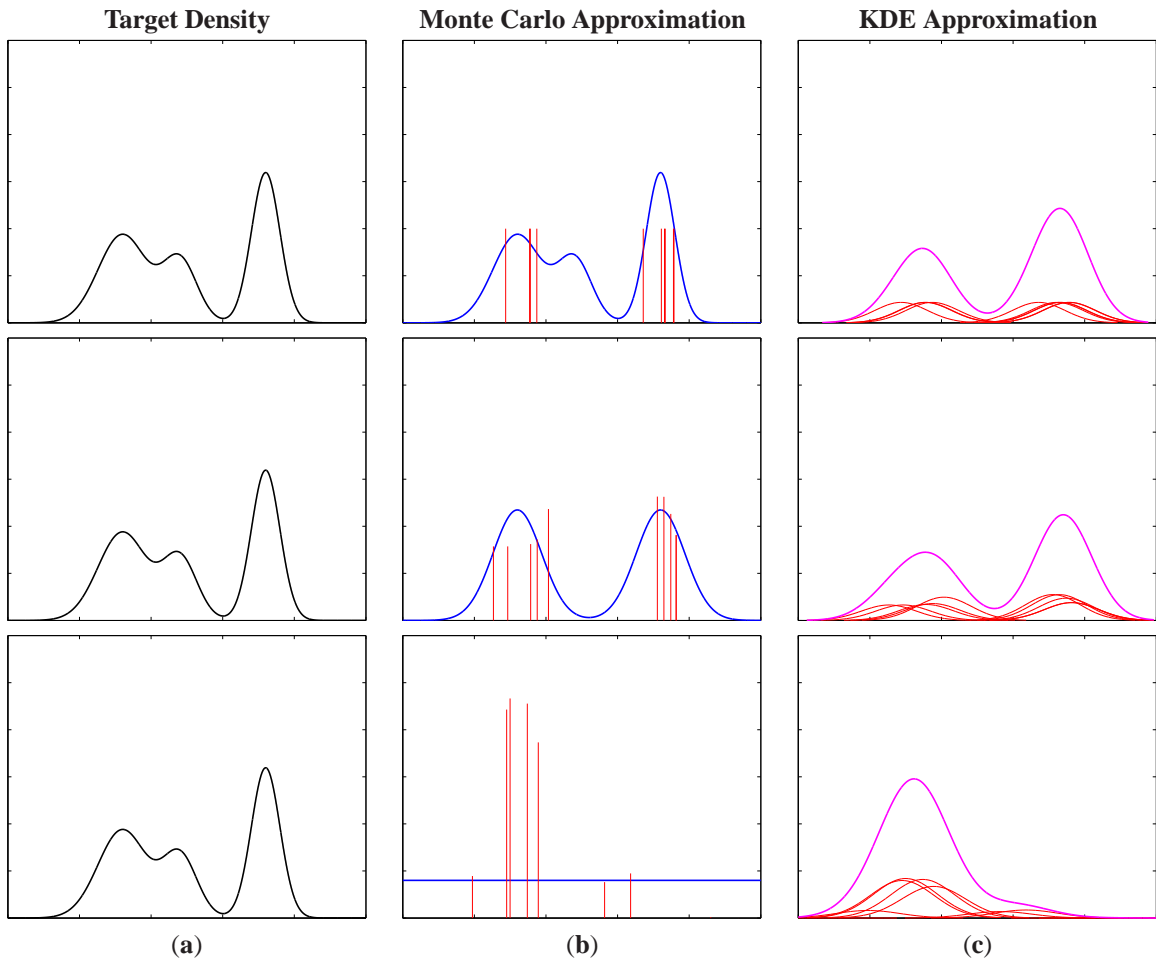


Figure 3.13: **Non-parametric representation of distribution (coarse)**. Monte Carlo approximation of the one-dimensional target distribution $p(\mathbf{X})$ (a) is shown using $N = 10$ weighted samples. Left column shows the target distribution being approximated, same in all cases. In (b) different importance/proposal distributions $q(\mathbf{X})$ (blue) and sample-based approximation of the target density based on 10 weighted samples drawn from these importance distribution (red) are shown. The height of the line representing the sample corresponds to the normalized importance weight. In (c) continuous approximation to the target distribution (magenta) obtained by Kernel Density Estimation (KDE) is shown. KDE places a Gaussian kernel (red) at every sample location obtained in (b). The bandwidth for the kernels was selected using a *rule-of-thumb* criterion. Rows of the figure illustrate the effects of different importance function. Top row shows the optimal importance function $q(\mathbf{X}) = p(\mathbf{X})$, middle row a *good* importance function $q(\mathbf{X}) \approx p(\mathbf{X})$, and bottom row a *poor* importance function. Clearly the quality of approximation is effected by how well the importance function matches target distribution. Also notice that *rule-of-thumb* estimate for the bandwidth tends to oversmooth the KDE approximation.

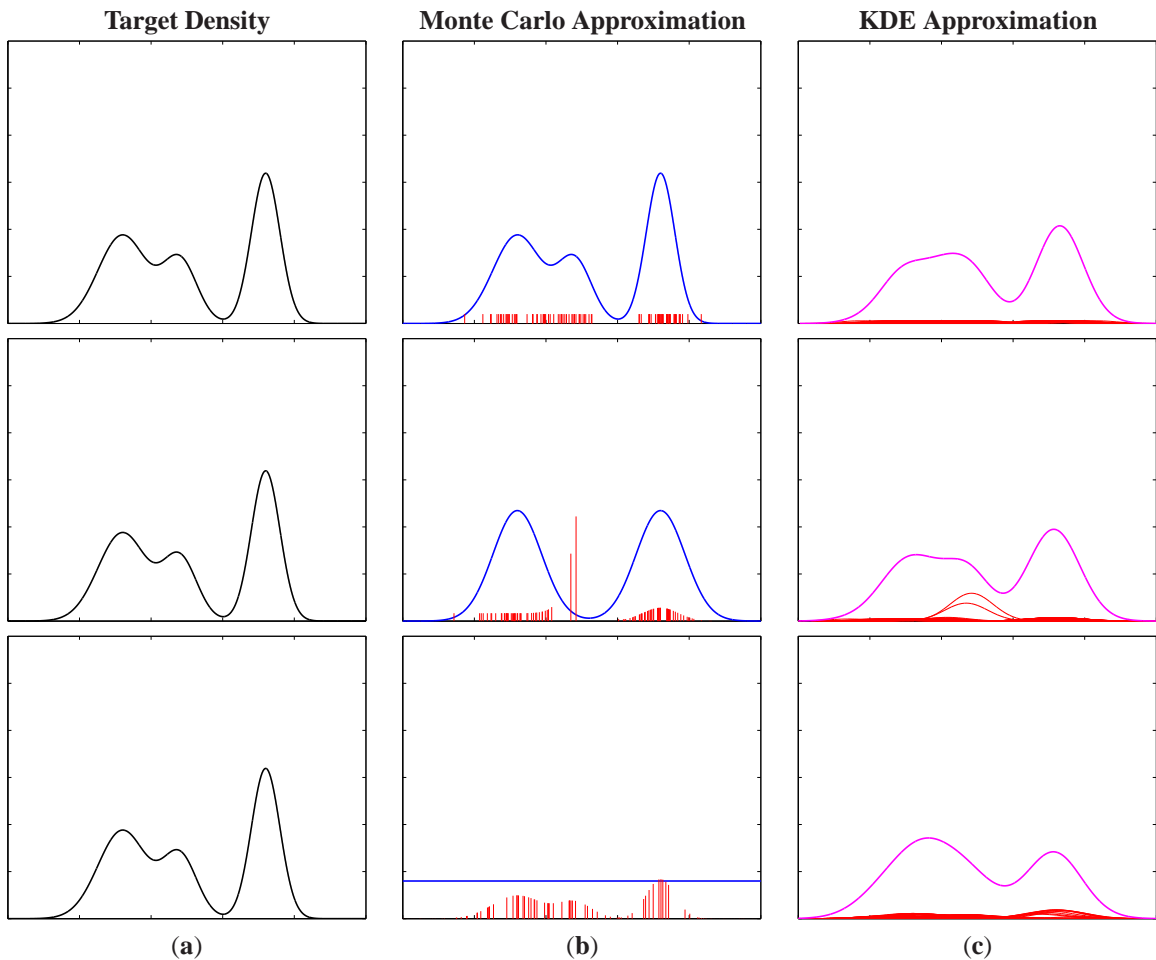


Figure 3.14: **Non-parametric representation of distribution (fine)**. Monte Carlo approximation of the one-dimensional target distribution $p(\mathbf{X})$ (a) is shown using $N = 100$ weighted samples. Left column shows the target distribution being approximated, same in all cases. In (b) different importance/proposal distributions $q(\mathbf{X})$ (blue) and sample-based approximation of the target density based on 100 weighted samples drawn from these importance distribution (red) are shown. The height of the line representing the sample corresponds to the normalized importance weight. In (c) continuous approximation to the target distribution (magenta) obtained by Kernel Density Estimation (KDE) is shown. KDE places a Gaussian kernel (red) at every sample location obtained in (b). The bandwidth for the kernels was selected using a *rule-of-thumb* criterion. Rows of the figure illustrate the effects of different importance function. Top row shows the optimal importance function $q(\mathbf{X}) = p(\mathbf{X})$, middle row a *good* importance function $q(\mathbf{X}) \approx p(\mathbf{X})$, and bottom row a poor importance function. Notice that the quality of approximation with 100 samples is considerably better, than with 10 (see Figure 3.13), in all cases. In fact, a poor choice of the importance function (bottom) to some extent here is remedied by the relatively large number of samples. As $N \rightarrow \infty$, the approximation approaches the true target distribution regardless of the choice of the importance function (so long as it is defined on the same domain).

Input: target density $p(\mathbf{X})$, that cannot be sampled directly
symmetric proposal distribution $q(a|b) = q(b|a)$
Output: sample $x \sim p(\mathbf{X})$.

1. Start with some initial value x_0 , such that $p(x_0) > 0$.
2. For each of $t = 0, \dots, T$ iterations,
 - (a) Sample next candidate state \tilde{x} from proposal distribution $\tilde{x} \sim q(\cdot|x_t)$ defined above.
 - (b) Compute acceptance probability

$$\alpha = \min \left(\frac{p(\tilde{x})}{p(x_t)}, 1 \right). \quad (3.62)$$

Notice that since we are computing a ratio, the normalizing constant Z cancels, making this convenient for cases where partition function cannot be computed explicitly.

- (c) Accept the candidate proposal with probability α . In other words draw a random sample from uniform distribution, $r \sim \mathcal{U}(0, 1)$, and let

$$x_{t+1} = \begin{cases} \tilde{x} & r < \alpha \\ x_t & \text{otherwise} \end{cases} \quad (3.63)$$

3. Let $x = x_T$.

Algorithm 2: Metropolis Sampler.

Chain for a number of steps. It can be shown that asymptotically the samples obtained in this way are unbiased samples from the target density. While finding such a chain sounds hard, it is easy to do in practice. Simulating the chain is also relatively simple, though it may take a significant amount of time to converge to the desired distribution. We will now describe a few methods for constructing and simulating the Markov Chains that have these desired properties.

Metropolis-Hastings Sampler

Let us assume we want to sample from the target density $p(\mathbf{X})$ where sampling directly from $p(\mathbf{X})$ is impractical (usually because partition function, *a.k.a.* normalizing constant, Z is hard to compute). Let us further assume that we have a *proposal distribution* (*a.k.a.* jumping distribution) $q(a|b)$ that given a current estimate for the sample, b , defined on the domain of \mathbf{X} proposes an alternative, a . This can be non-informative distribution (*e.g.* random walk), the only constraint is that this distribution must be symmetric, *i.e.* $q(a|b) = q(b|a)$. The Metropolis algorithm [141] generates a sample from the target distribution by running Algorithm 2.

It is easy to see that the outlined algorithm generates a Markov Chain with states $\{x_0, x_1, \dots, x_T\}$ since the state at time t , x_t only depends on the state at $t - 1$. After a sufficiently long generation process this Markov Chain approaches a stationary distribution and samples from the chain are samples from the target density $p(\mathbf{X})$.

Hastings [80] generalized the Metropolis algorithm, to work with an arbitrary proposal function, that must not be symmetric. In that case the acceptance probability α must be modified to be the following:

$$\alpha = \min \left(\frac{p(\tilde{x})q(x_t|\tilde{x})}{p(x_t)q(\tilde{x}|x_t)}, 1 \right), \quad (3.64)$$

The resulting algorithm is commonly called Metropolis-Hasting algorithm [80]. Once the samples from the target distribution are generated in this way, we can of course use them in the Monte Carlo framework to approximate the desired expectation.

Gibbs Sampler

The *Gibbs sampler* [75] is a special case of the Metropolis-Hastings sampler where the proposed states are always accepted, $\alpha = 1$. Let $p(\mathbf{X})$ be once again the target density we want to sample from. Let us further assume that the state space can be partitioned in some way, $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$. The Gibbs sampler samples from $p(\mathbf{X})$ by iteratively sampling from the univariate conditionals of the form $p(\mathbf{X}_i | \mathbf{X} \setminus \mathbf{X}_i)$ by keeping $N - 1$ variables fixed at any given time. Such conditional distributions are often easy to simulate, as opposed to the full joint. Thus a Gibbs sampler simulates N random variables sequentially, rather simulating all variables at once subject to the joint target distribution.

At any given time t a particular variable i is selected for resampling, and the rest are kept fixed. In the Metropolis-Hastings algorithm context, the Gibbs sampler can be defined by specifying a particular form for the proposal distribution,

$$q(\tilde{x}|x^{(t)}) = \begin{cases} \tilde{x}_i \sim p(\mathbf{X}_i | \{\mathbf{X}_j = x_j^{(t)} | j \in [1, \dots, N] \setminus i\}) \\ \tilde{x}_j = x_j^{(t)} \end{cases} \quad j \in [1, \dots, N] \setminus i \quad (3.65)$$

that specifies that the next proposed state from $x^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \dots, x_N^{(t)}\}$ for a current choice of the variable, say $i = 1$, will be $\tilde{x} = \{\tilde{x}_1, x_2^{(t)}, \dots, x_N^{(t)}\}$ sampled according to conditional as stated above. The acceptance probability for this particular choice of the proposal can be written,

$$\alpha = \min\left(\frac{p(\tilde{x})q(x_t|\tilde{x})}{p(x_t)q(\tilde{x}|x_t)}, 1\right) \quad (3.66)$$

$$= \min\left(\frac{p(\tilde{x}_1, x_2^{(t)}, \dots, x_N^{(t)})p(x_1^{(t)}|x_2^{(t)}, \dots, x_N^{(t)})}{p(x_1^{(t)}, \dots, x_N^{(t)})p(\tilde{x}_1|x_2^{(t)}, \dots, x_N^{(t)})}, 1\right) \quad (3.67)$$

$$= \min\left(\frac{p(\tilde{x}_1, x_2^{(t)}, \dots, x_N^{(t)})p(x_1^{(t)}, x_2^{(t)}, \dots, x_N^{(t)})p(x_2^{(t)}, \dots, x_N^{(t)})}{p(x_1^{(t)}, \dots, x_N^{(t)})p(\tilde{x}_1, x_2^{(t)}, \dots, x_N^{(t)})p(x_2^{(t)}, \dots, x_N^{(t)})}, 1\right) \quad (3.68)$$

$$= \min(1, 1) = 1, \quad (3.69)$$

confirming that we should always accept the proposed state. This analysis holds for any choice of variable i . Hence, the Gibbs sampler can be more compactly described using Algorithm 3.

As the above equations are iterated, the sample $x^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \dots, x_N^{(t)}\}$ converges to a sample from the target density $p(\mathbf{X})$. It has been shown that permuting the order in which the variables are resampled, sometimes improves the rate of convergence. This can be easily done by sampling i in the beginning of each iteration from uniform discrete distribution, $i \sim \mathcal{U}(1, N)$.

3.6.4 Sequential Importance Sampling

The *Sequential Importance Sampling* (SIS), frequently also called *Particle Filtering* (PF), is a Monte Carlo (MC) based method that gives rise to an extensive body of literature on sequential Bayesian filtering developed

<p>Input: set of uni-variate conditional distributions of the following form:</p> $p(\mathbf{X}_1 \mathbf{X}_2, \mathbf{X}_3, \dots, \mathbf{X}_N)$ \vdots $p(\mathbf{X}_i \mathbf{X}_1, \dots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \dots, \mathbf{X}_{N-1})$ \vdots $p(\mathbf{X}_N \mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{N-1})$ <p>Output: sample from the joint $x \sim p(\mathbf{X})$, where $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_1, \dots, \mathbf{X}_N\}$</p> <ol style="list-style-type: none"> 1. Select an initial value for all variables $x^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \dots, x_N^{(0)}\}$ 2. For each $t \in [1, \dots, T]$ iterations <ol style="list-style-type: none"> (a) For each $i \in [1, \dots, N]$, sample from an appropriate conditional: $x_i^{(k)} \sim p(\mathbf{X}_i x_1^{(k-1)}, \dots, x_{i-1}^{(k-1)}, x_{i+1}^{(k-1)}, \dots, x_N^{(k-1)}) \quad (3.70)$ $x_j^{(k)} = x_j^{(k-1)}, j \in [1, \dots, N] \setminus i, \quad (3.71)$ <p style="text-align: center;">where $k = (t-1)N + t$.</p> 3. Let $x = \{x_1^{(TN)}, x_2^{(TN)}, \dots, x_N^{(TN)}\}$.

Algorithm 3: Gibbs Sampler.

over at least 10 years [9, 54]. *Sequential Bayesian filtering* refers to a class of inference methods that estimate the values of a variable that evolves over time (often based on some underlying stochastic dynamical system/process). The method is commonly called *filtering* when the estimates are done sequentially in time, it is also referred to as *smoothing* when the estimate for the state involves future observations (as in undirected graphical HMMs), and *prediction* in cases where one is faced with estimating future values for the evolution of the system based only on the past observations.

Sequential Importance Sampling (SIS) gives rise to a number of variant methods for doing sequential posterior estimation. In this section we will first introduce the generic SIS approach and then a few variants, including Sampling Importance Resampling (SIR) (*a.k.a.* Condensation). The key idea in SIS and Particle Filtering in general, as with any MC approach, is to approximate the posterior using a set of weighted samples. Let us consider inference in the Hidden Markov Model illustrated in Figure 3.4 (a). Assuming that we have a model defined for T time instants, the joint distribution can be written as was discussed in Section 3.2.2,

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{Y}_1 | \mathbf{X}_1) p(\mathbf{X}_1) \prod_{t=2}^T p(\mathbf{Y}_t | \mathbf{X}_t) p(\mathbf{X}_t | \mathbf{X}_{t-1}). \quad (3.72)$$

We can approximate the desired posterior density, $p(\mathbf{X}_{1:T} | \mathbf{Y}_{1:T}) = p(\mathbf{X}_1, \dots, \mathbf{X}_T | \mathbf{Y}_1, \dots, \mathbf{Y}_T)$, that often cannot be computed directly, using the importance sampling approach introduced in Section 3.6.1. This is done by introducing an importance density that is easy to sample from, $q(\mathbf{X}_{1:T} | \mathbf{Y}_{1:T}) = q(\mathbf{X}_1, \dots, \mathbf{X}_T | \mathbf{Y}_1, \dots, \mathbf{Y}_T)$, and approximating the desired posterior using a weighted sample set of N samples,

$$p(\mathbf{X}_{0:T} | \mathbf{Y}_{1:T}) \approx \sum_{i=1}^N w_{1:T}^{(i)} \delta(\mathbf{X}_{0:T} - s_{1:T}^{(i)}) \quad (3.73)$$

where

$$s_{1:T}^{(i)} \sim q(\mathbf{X}_{1:T} | \mathbf{Y}_{1:T}) \quad (3.74)$$

$$w_{1:T}^{(i)} \propto \frac{p(s_{1:T}^{(i)} | \mathbf{Y}_{1:T})}{q(s_{1:T}^{(i)} | \mathbf{Y}_{1:T})}. \quad (3.75)$$

Unfortunately this form of inference is often both inconvenient and computationally intractable for two reasons: **(1)** it requires devising and sampling from a proposal function in a very high dimensional space that grows with the number of nodes in the graph, and **(2)** it requires that the entire process has been observed (batch mode inference). SIS remedies both of these problems by estimating the posterior recursively (*a.k.a. on-line posterior estimation*).

Let us assume for the moment that we have a sample based representation of the posterior up to, but not including, time t , $\{s_{1:t-1}^{(i)}, w_{1:t-1}^{(i)} | i \in [1, \dots, N]\}$. If we want to approximate the posterior recursively or sequentially, and choose an appropriate importance density that factors,

$$q(\mathbf{X}_{1:t} | \mathbf{Y}_{1:t}) = q(\mathbf{X}_t | \mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t}) q(\mathbf{X}_{1:t-1} | \mathbf{Y}_{1:t-1}), \quad (3.76)$$

then samples from $s_{1:t}^{(i)} \sim q(\mathbf{X}_{1:t} | \mathbf{Y}_{1:t})$ can easily be constructed recursively. This is achieved by taking an already existing sample based representation for the posterior at time $t-1$, and augmenting each particle $s_{1:t-1}^{(i)}$ with a sample based estimate of the state at time t , $s_t^{(i)} \sim q(\mathbf{X}_t | s_{1:t-1}^{(i)}, \mathbf{Y}_{1:t})$. The recursive equation for the weights can also be derived by following rules of importance sampling, resulting in the following:

$$w_{1:t}^{(i)} \propto w_{1:t-1}^{(i)} \frac{p(\mathbf{Y}_t | s_t^{(i)}) p(s_t^{(i)}, s_{t-1}^{(i)})}{q(s_t^{(i)} | s_{1:t-1}^{(i)}, \mathbf{Y}_{1:t})}. \quad (3.77)$$

The equation further simplifies if we assume that we are only interested in the marginal distribution $p(\mathbf{X}_t | \mathbf{Y}_{1:t})$ (*a.k.a. filtered estimate*) and not the full posterior, in which case the equation for the weights is reduced to,

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{Y}_t | s_t^{(i)}) p(s_t^{(i)}, s_{t-1}^{(i)})}{q(s_t^{(i)} | s_{t-1}^{(i)}, \mathbf{Y}_t)}. \quad (3.78)$$

resulting for the sample based representation for the marginal posterior density,

$$p(\mathbf{X}_t | \mathbf{Y}_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \delta(\mathbf{X}_t - s_t^{(i)}). \quad (3.79)$$

The outlined approach is the basis of the General Particle Filter, Algorithm 4.

Degeneracy, Impoverishment and Resampling

A common problem with SIS filter is what is frequently known as *degeneracy*. In particular, often after a few iterations, all but one sample will have a negligible weight, contributing virtually nothing to the approximation of the marginal posterior. In [118] a suitable simple measure of degeneracy was introduced, to measure effective sample size, N_{eff} , at any given time t ,

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w_t^{(i)})^2} \quad (3.80)$$

Clearly degeneracy effects are undesirable, since they lead to poor approximations of the posterior (at high computational cost⁴). The two possible solutions are to either choose a good proposal function, or to resample the particle set. Even with good proposal functions, often resampling is necessary (but not as frequently).

The basic idea of *resampling* is, whenever significant degeneracy is observed ($N_{eff} < N_{th}$, where N_{th} is an empirically chosen threshold), to eliminate particles with the low weight, replacing them with particles that have a more significant support. The resampling is implemented by drawing with replacement a new set of unweighted samples (representing the same density) $\{\tilde{s}_t^{(i)}, \tilde{w}_t^{(i)} | i \in [1, \dots, N]\}$, $\tilde{w}_t^{(i)} = \frac{1}{N}$ for all $i \in [1, \dots, N]$, where the probability of drawing a sample is proportional to the weight of the sample in the original set $\{s_t^{(i)}, w_t^{(i)} | i \in [1, \dots, N]\}$. In other words, $p(\tilde{s}_t^{(i)} = s_t^{(i)}) = w_t^{(i)}$.

While resampling helps to reduce degeneracy in particle filters, it may introduce other problems. The most significant of such problems is what often referred to as *sample impoverishment*, which arises when one or few particles have a high weight relative to the rest. In such a case the new resampled approximation will have many repeated samples. This lack of diversity in samples tends to translate into poor approximation of the posterior. This problem is particularly acute where the process noise is small, often leading to collapse of all particles to a single sample point.

Sampling Importance Resampling Filter or Condensation

Thus far we have introduced a conceptual Particle Filtering framework that is common to many recursive Bayesian approaches. To build a realistic algorithm a choice of importance function must be specified. In this section we will introduce one variant of the Particle Filter called *Sampling Importance Resampling* (SIR) [76] or in the computer vision community better known as *Condensation* introduced by Blake and Isard in [25].

This particular Particle Filter variant is derived by making two assumptions: **(1)** that we going to use the *temporal prior* as a proposal function (i.e. $q(\mathbf{X}_t | \mathbf{X}_{t-1}, \mathbf{Y}_{1:t}) = p(\mathbf{X}_t | \mathbf{X}_{t-1})$), and **(2)** resampling is applied at every iteration (i.e. $N_{th} = \infty$). Notice that this choice of importance density assumes a relatively simple temporal prior from which samples can be generated. In practice, often linear models with Gaussian noise are chosen for $p(\mathbf{X}_t | \mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t | \beta \mathbf{X}_{t-1}, \Sigma)$. Also notice that since resampling is done at every time step, the computation of weights reduces to a particularly simple (non-recursive) form,

$$w_t^{(i)} \propto p(\mathbf{Y}_t | s_t^{(i)}), \quad (3.86)$$

that only depends on the *likelihood*. This particular variant of a particle filter has been applied extensively to object tracking in computer vision domain; for discussion and more detailed derivation see Section 2.8.

⁴Samples, while contributing little to the overall quality of approximation, still need to be updated in the filtering framework.

<p>Input: sample based approximation to the marginal posterior at time $t - 1$ $p(\mathbf{X}_{t-1} \mathbf{Y}_{1:t-1}) \approx \{s_{t-1}^{(i)}, w_{t-1}^{(i)} i \in [1, \dots, N]\}$</p> <p>Output: sample based approximation to the marginal posterior at time t $p(\mathbf{X}_t \mathbf{Y}_{1:t}) \approx \{s_t^{(i)}, w_t^{(i)} i \in [1, \dots, N]\}$</p> <ol style="list-style-type: none"> 1. For each sample $i \in [1, \dots, N]$ <ol style="list-style-type: none"> (a) Draw $s_t^{(i)} \sim q(\mathbf{X}_t s_{t-1}^{(i)}, \mathbf{Y}_t)$ from proposal function $q(\cdot)$. (b) Compute the sample weight $w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{Y}_t s_t^{(i)}) p(s_t^{(i)}, s_{t-1}^{(i)})}{q(s_t^{(i)} s_{t-1}^{(i)}, \mathbf{Y}_t)}. \quad (3.81)$ <ol style="list-style-type: none"> 2. Normalize weights for each sample $i \in [1, \dots, N]$ $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^N w_t^{(i)}} \quad (3.82)$ 3. Calculate effective sample size $N_{eff} = \frac{1}{\sum_{i=1}^N (w_t^{(i)})^2} \quad (3.83)$ 4. If $N_{eff} < N_{th}$, resample the particle set by drawing with replacement from the sample based approximation of the density $p(\mathbf{X}_t \mathbf{Y}_{1:t}) \approx \{s_t^{(i)}, w_t^{(i)} i \in [1, \dots, N]\}$. <ol style="list-style-type: none"> (a) For each sample $k \in [1, \dots, N]$, $\tilde{s}_t^{(k)} \sim \{s_t^{(i)}, w_t^{(i)} i \in [1, \dots, N]\} \quad (3.84)$ $\tilde{w}_t^{(k)} = \frac{1}{N} \quad (3.85)$ (b) For each sample $k \in [1, \dots, N]$, let $s_t^{(k)} = \tilde{s}_t^{(k)}$ and $w_t^{(k)} = \tilde{w}_t^{(k)}$

Algorithm 4: Generic Particle Filter.

Number of Particles

All particle filters, use a set of N weighted samples to represent the posterior. As the number of samples $N \rightarrow \infty$ the approximation approaches the true posterior, as with standard importance sampling. However, in practice, due to computational cost, inference must be done with as few samples/particles as possible. In general, it is hard to automatically select the number of particles needed for good posterior approximation. The number of particles will depend on the structure and shape of the posterior, proximity of proposal distribution to the true posterior, the complexity of underlying dynamical process, observation noise, and dimensionality of the state-space. In [136], a lower bound for the number of particles needed, known as *survival rate*, is derived. In particular,

$$N \geq \frac{N_{min}}{\gamma^d}, \quad (3.87)$$

where N is the lower bound on the number of samples needed, subject to the dimensionality of the state space d , a survival rate $\gamma \ll 1$ that is a constant related to how well the posterior is approximated by the filter (and is a function of posterior and proposal distribution shape and complexity); N_{min} is a parameter designating the minimum number of particles to survive the resampling. The survival rate γ will typically be lower for noisy posterior distributions that are not modeled well, requiring more samples to represent them adequately. Consequently, for proposal distributions that are poor approximations to the posterior, γ will be low as well, leading to similar artifacts. Lastly, the number of samples required to model a high dimensional posterior, according to this metric, will grow exponentially with the dimensionality of the state-space d . This is known in computer vision as the *curse of dimensionality*.

Regularized Particle Filter

The general Particle Filter framework as well as the particular instance of SIR (or Condensation) in the previous section attempts to resolve the issue of degeneracy with resampling and/or good proposal densities. However, as mentioned before, resampling often leads to sample impoverishment. At least in part this problem can be attributed to the fact that when the approximation to the density (encoded using a weighted sample set) is resampled, we are sampling from a discrete representation of the posterior instead of the full continuous approximation.

A *Regularized Particle Filter* (RPF) [154] was introduced to remedy this phenomenon. The RPF is identical to SIR filter introduced in the previous section, in all but one respect. During resampling, RPF resamples from the continuous approximation of the marginal posterior obtained using Kernel-based approximation introduced in Section 3.6.2. RPF bears striking similarity to the Particle Message Passing (PAMPAS) [99] approach, that will be discussed in the next section and used throughout this thesis. In fact, Particle Message Passing (PAMPAS) is a generalization of the RPF filter that allows inference in graphs of arbitrary topology. For topology of Hidden Markov Models (HMMs) it can be shown that PAMPAS reduces to RPF.

3.7 Particle Message Passing

Particle filters introduced in the previous section are effective for inference in many different models and applications, however, they are customized for temporal filtering or estimation problems in Hidden Markov Models. Belief Propagation introduced in Section 3.5.2 provides the means of effective inference in graphs of arbitrary topology, however, is typically restricted to discrete variables or continuous Gaussian variables for tractable inference. In this section we will introduce *Particle Message Passing* (PAMPAS) [99], a variant of Non-parametric Belief Propagation [220], that is able to perform approximate inference in the graphs of arbitrary topology and makes no explicit assumptions about parametric form for the variables or potential functions. In Particle Message Passing we generalize Particle Filters to work for graphs of arbitrary topology. PAMPAS will underline the inference tasks in this thesis.

As in standard Belief Propagation, introduced in Section 3.5.2, for convenience we will restrict ourselves to inference in pair-wise MRFs. However, similar results can be derived for a general MRF and Bayesian Networks. A simple extension would lead to variant that would work for factor graphs. Given a pair-wise Markov Random Field specified by the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where we have a set of hidden, $\mathcal{V}_{\mathbf{X}}$, and observed, $\mathcal{V}_{\mathbf{Y}}$, nodes corresponding to variables $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ and $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_M\}$ respectively, we

can write the joint distribution as follows,

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{E}, i \in \mathcal{V}_{\mathbf{X}}, j \in \mathcal{V}_{\mathbf{X}}} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \prod_{i \in \mathcal{V}_{\mathbf{X}}} \phi_i(\mathbf{X}_i, \mathbf{Y}). \quad (3.88)$$

If we assume that \mathbf{X}_i 's are continuous random variables, we can write down the recursive message passing and belief estimation equations underlying BP (see Section 3.5.2):

$$\textbf{Continuous message passing:} \quad m_{ij}(\mathbf{X}_j) \propto \int_{\mathbf{X}_i} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i, \mathbf{Y}) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i) d\mathbf{X}_i \quad (3.89)$$

$$\textbf{Belief estimation:} \quad b_i(\mathbf{X}_i) \propto \phi_i(\mathbf{X}_i, \mathbf{Y}) \prod_{k \in A(i)} m_{ki}(\mathbf{X}_i). \quad (3.90)$$

The key observation, underlying both Particle Message Passing and the more general NBP [220], is that integration required to perform message passing can be approximated using Monte Carlo techniques introduced in Section 3.6. For convenience, we will first formulate PAMPAS for a restricted set of MRFs where potentials $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ can be expressed using finite Gaussian mixtures and then address the more general case where some potential functions do not have this convenient form. Notice, that even this more restrictive case, where potentials can be expressed using finite Gaussian mixtures, produces a considerably richer class of models than the purely Gaussian MRFs introduced in Section 3.5.2. In its original form, Particle Message Passing was introduced by Michael Isard in [99]; here we generalize the original formulation of [99] to make the approach appropriate for the applications addressed in this thesis.

As in [99], for convenience we first introduce a probability density function that we will call *message foundation* (where convenient we will use the following shorthand notation for the likelihood $\phi_i(\mathbf{X}_i) = \phi_i(\mathbf{X}_i, \mathbf{Y})$),

$$m_{ij}^F(\mathbf{X}_i) \equiv \frac{1}{Z_{ij}} \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i), \quad (3.91)$$

where Z_{ij} is a normalizing constant. Intuitively, the message foundation approximates the distribution over \mathbf{X}_i , that is then used to derive compatible distribution for \mathbf{X}_j encoded by the message $m_{ij}(\mathbf{X}_j)$. We can use Monte-Carlo integration to approximate the messages by drawing N samples from the message foundation, $\{s_{ij}^{(n)} \sim m_{ij}^F(\mathbf{X}_i) | n \in [1, \dots, N]\}$, and then propagating these samples through a potential function $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$, resulting in the following

$$m_{ij}(\mathbf{X}_j) = \sum_{n=1}^N w_{ij}^{(n)} \psi_{ij}(\mathbf{X}_i = s_{ij}^{(n)}, \mathbf{X}_j) \quad (3.92)$$

mixture approximation to the message, where $w_{ij}^{(n)} = 1/N$ is the weight associated with each sample.

Assuming that $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ can be modeled using a joint distribution represented by the Mixture of M_{ij} Gaussians (MoG), the resulting mixture distribution,

$$m_{ij}(\mathbf{X}_j) = \frac{1}{N} \sum_{n=1}^N \psi_{ij}(\mathbf{X}_i = s_{ij}^{(n)}, \mathbf{X}_j) = \frac{1}{N} \sum_{n=1}^N \psi_{ij}(\mathbf{X}_j | \mathbf{X}_i = s_{ij}^{(n)}), \quad (3.93)$$

for the message will be a Gaussian mixture as well with $M_{ij}N$ components. Notice that by assuming a MoG form for $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ we can model a large class of potential functions. For tractable inference, however, M_{ij} must remain small (on the order of tens of components).

In general, we can sample from any *importance function*, $\{s_{ij}^{(n)} \sim q_{ij}(\mathbf{X}_i) | n \in [1, \dots, N]\}$ so long as we apply importance re-weighting resulting in non-uniform weight $w_{ij}^{(n)} \propto m_{ij}^F(s_{ij}^{(n)})/q_{ij}(s_{ij}^{(n)})$. As with any particle filtering the choice of importance function will effect the convergence properties of the algorithm. Furthermore, samples can be stratified into a number of groups.

To compute the marginal distribution over \mathbf{X}_i , samples can be drawn from the belief distribution $b_i(\mathbf{X}_i)$ directly or using importance sampling. These possibly weighted samples (sum of Dirac functions) serve as an approximate representation of the true marginal. If continuous representation of the marginal is required, kernel density estimation can be used to smooth the particle set (see Section 3.6.2).

3.7.1 Sampling from a Product of Gaussian Mixtures

The key to inference using PAMPAS is sampling from the message foundation $m_{ij}^F(\mathbf{X}_i)$. For the moment, as in previous section, let us assume that both the likelihoods, $\phi_i(\mathbf{X}_i, \mathbf{Y})$, and the potentials, $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$, can be expressed as mixtures of Gaussians. In that case sampling from $m_{ij}^F(\mathbf{X}_i)$ amounts to sampling from a product of Gaussian mixtures. We will consider a more general case, where only a subset of potentials have this form in the next section.

Let us consider a case where we have a product of N mixtures of M_n , $n \in [1, \dots, N]$, components respectively, resulting in the product that can be expressed as a mixture itself with $\prod_{n=1}^N M_n$ Gaussian components. Hence, the brute force approach to sampling would require time exponential in the number of mixtures, $O(\prod_{n=1}^N M_n)$ ($O(M^N)$ if for all mixtures $M_n = M$). This is only tractable for products of few mixtures (typically $N < 3$) having relatively few mixture components. To make the sampling tractable, Sudderth *et al.* [220] propose a Gibbs sampler (see Section 3.6.3), that can produce the unbiased exact samples from the product in $O(KNM^2)$, as the number of iterations $K \rightarrow \infty$. In practice with a relatively small value of K a good sampling can be achieved (we typically use $5 < K < 10$). In cases where $N < 3$ the brute force sampling is tractable, and we use the exact sampler instead.

The Gibbs sampler works by iteratively sampling labels $L = \{l_1, l_2, \dots, l_N\}$, where $l_n \in [1, \dots, M_n]$ corresponding to the Gaussian components in mixture n . Initially L is initialized by randomly sampling the labels. We found that initializing the sampler by sampling l_n 's according to the probability of the mixture components in the mixture n , as in [220], led to slower convergence in some cases. Once we have an initial set of labels L , we pick an integer $k \in [1, \dots, N]$ at random and sample l_k according to the marginal distribution on the labels. The full algorithm introduced in [220] is restated in Algorithm 5 for completeness.

Significant optimizations to the above algorithm can be made for the case where all mixture components have the same covariance. Similarly, for the specific case of mixtures that have diagonal covariance structure, an approximate sampling scheme was introduced in [95] that can sample from the product in $O(KMN)$.

3.7.2 Sampling from More General Forms of Message Foundation

It is impractical to assume that the likelihood $\phi_i(\mathbf{X}_i, \mathbf{Y})$ can be explicitly modeled using a Gaussian mixture, in fact in most cases $\phi_i(\mathbf{X}_i, \mathbf{Y})$ will be too complex to be able to sample from it directly. It is also possible that some sub-set of potentials $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ will not be able to be modeled using a Gaussian mixture effectively.

Input: D Gaussian mixtures, where mixture $d \in [1, \dots, D]$ contains M_d Gaussian components with parameters $\{w_d^{(m)}, \mu_d^{(m)}, \Lambda_d^{(m)}\}$ respectively.

Output: $x \sim \prod_{d=1}^D \sum_{m=1}^{M_d} w_d^{(m)} \mathcal{N}(\mu_d^{(m)}, \Lambda_d^{(m)})$.

1. For each $d \in [1, \dots, D]$, choose a starting label $l_d \in [1, \dots, M_d]$ by sampling $p(l_d = j) \propto 1/M_d$. For convenience, let $L = \{l_1, l_2, \dots, l_D\}$.

2. For each $d \in [1, \dots, D]$,

(a) Calculate the covariance matrix

$$\Lambda^* = \left(\sum_{i=1}^{d-1} [\Lambda_i^{(l_i)}]^{-1} + \sum_{i=d+1}^D [\Lambda_i^{(l_i)}]^{-1} \right)^{-1} \quad (3.94)$$

and the mean vector

$$\mu^* = \Lambda^* \left(\sum_{i=1}^{d-1} [\Lambda_i^{(l_i)}]^{-1} \mu_i^{(l_i)} + \sum_{i=d+1}^D [\Lambda_i^{(l_i)}]^{-1} \mu_i^{(l_i)} \right) \quad (3.95)$$

of the Gaussian resulting from taking a product of Gaussians designated by the set of fixed labels L/l_d .

(b) For each $m \in [1, \dots, M_d]$ calculate

$$\bar{\Lambda}^{(m)} = \left([\Lambda_d^{(l_d)}]^{-1} + [\Lambda^*]^{-1} \right)^{-1}, \quad \bar{\mu}^{(m)} = \bar{\Lambda}^{(m)} \left([\Lambda_d^{(l_d)}]^{-1} \mu_d^{(l_d)} + [\Lambda^*]^{-1} \mu^* \right)$$

(c) Sample a new value for label l_d according to

$$p(l_d = m) \propto w_d^{(m)} \frac{\mathcal{N}(x; \mu_d^{(m)}, \Lambda_d^{(m)}) \mathcal{N}(x; \mu^*, \Lambda^*)}{\mathcal{N}(x; \bar{\mu}^{(m)}, \bar{\Lambda}^{(m)})} \quad (3.96)$$

where x is any convenient point (we use $x = \bar{\mu}^{(m)}$).

3. Repeat step 2 for K iterations.

4. Compute the mean and covariance for the Gaussian component of the product designated by L .

$$\bar{\Lambda} = \left(\sum_{i=1}^D [\Lambda_i^{(l_i)}]^{-1} \right)^{-1}, \quad \bar{\mu} = \bar{\Lambda} \left(\sum_{i=1}^D [\Lambda_i^{(l_i)}]^{-1} \mu_i^{(l_i)} \right)$$

5. Draw sample $x \sim \mathcal{N}(\bar{\mu}, \bar{\Lambda})$.

Algorithm 5: Gibbs Sampler for a Product of Gaussian Mixtures. Original algorithm is introduced in [220].

Hence, we must handle a case where only a sub-set of the terms in the message foundation, $m_{ij}^F(\mathbf{X}_i)$, will have the convenient Gaussian mixture form; for convenience let us call the product of those terms $m_{ij}^{FS}(\mathbf{X}_i)$. The rest of the terms that do not have the convenient form from which we can easily sample can be combined into $m_{ij}^{FE}(\mathbf{X}_i)$, such that $m_{ij}^F(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i)m_{ij}^{FE}(\mathbf{X}_i)$. For example, if the likelihoods $\phi_i(\mathbf{X}_i, \mathbf{Y})$ are not

Gaussian mixtures and potentials $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ are, then $m_{ij}^{FS}(\mathbf{X}_i) = \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i)$ and $m_{ij}^{FE}(\mathbf{X}_i) = \phi_i(\mathbf{X}_i, \mathbf{Y})$.

Of particular interest is the case where some (but not all) potentials can be represented using mixtures of Gaussians (MoGs). Let us consider a case where a graph contains potentials of two distinct types⁵ $\psi_{ij}^{(MoG)}(\mathbf{X}_i, \mathbf{X}_j)$ and $\psi_{ij}^{(\neg MoG)}(\mathbf{X}_i, \mathbf{X}_j)$. Assuming that the messages are obtained as before by propagating samples via these potentials, this will lead to two corresponding message types, $m_{ij}^{(MoG)}(\mathbf{X}_j)$ and $m_{ij}^{(\neg MoG)}(\mathbf{X}_j)$. We can now collect all the terms that have a convenient form for sampling into $m_{ij}^{FS}(\mathbf{X}_i) = \prod_{k \in A(i) \setminus j} m_{ki}^{(MoG)}(\mathbf{X}_i)$, and the rest of the terms into $m_{ij}^{FE}(\mathbf{X}_i) = \phi_i(\mathbf{X}_i, \mathbf{Y}) \prod_{k \in A(i) \setminus j} m_{ki}^{(\neg MoG)}(\mathbf{X}_i)$.

The PAMPAS algorithm of Section 3.7 can then be easily modified to handle this case by setting the importance function $q_{ij}(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i)$. The new NBP variant will then proceed by sampling $s_{ij}^{(n)} \sim m_{ij}^{FS}(\mathbf{X}_i)$ from the importance function and then the importance re-weighting will assign the weight of $w_{ij}^{(n)} \propto m_{ij}^{FE}(s_{ij}^{(n)})$ to the sample. The resulting message will then be obtained as before, by conditioning on \mathbf{X}_i , i.e. $\psi_{ij}(\mathbf{X}_i = s_{ij}^{(n)}, \mathbf{X}_j) = \psi_{ij}(\mathbf{X}_j | \mathbf{X}_i = s_{ij}^{(n)})$.

3.7.3 Choice of Importance Functions

Sections 3.7.1 and 3.7.2 introduce a particular choice of importance function for approximating messages in BP using Monte Carlo,

$$q_{ij}(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i). \quad (3.97)$$

However, this choice of an importance function is not always effective. In particular, consider inference in an undirected chain (e.g. Hidden Markov Model with 3 hidden random variables – $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$). The limitation is that for messages $m_{12}(\mathbf{X}_2)$ and $m_{32}(\mathbf{X}_2)$ the corresponding importance functions $q_{12}(\mathbf{X}_1) = m_{12}^{FS}(\mathbf{X}_1) = \emptyset$ and $q_{32}(\mathbf{X}_3) = m_{32}^{FS}(\mathbf{X}_3) = \emptyset$ are non-informative. While the messages will correctly weigh the non-informative samples, in high-dimensional spaces this will lead to poor approximation to the messages (see discussion in Section 3.6.1). One solution is to use a different importance function that facilitates placement of samples in high probability regions. One natural choice is to use belief as an importance function. In other words, let

$$q_{ij}(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i) m_{ji}(\mathbf{X}_i). \quad (3.98)$$

This choice of importance function will also in some cases facilitate faster *mixing* between messages, leading to overall faster convergence of BP [99].

In order to use either of the two importance functions, however, messages must be *initialized*. In discrete belief propagation messages are often initialized by uniform distributions, that are then refined by Belief Propagation message passing. In the continuous case, and more specifically in high-dimensional continuous case, having uniform messages will lead to non-informative importance functions. This in turn will lead to poor approximation to the true messages, and often will not lead to convergence of NBP. Hence, for the non-parametric BP inference to be effective, some or all messages must be initialized to semi-informative distributions⁶. In most tracking applications [219] this is done by providing an initial pose, or distribution

⁵In both cases we assume that conditional distributions of the form $\psi_{ij}(\mathbf{X}_i = x, \mathbf{X}_j)$ can be derived analytically.

⁶Notice that this is equivalent to having an informative importance function for some of the variables in the graph.

over poses at the first frame. In this thesis we take a very different approach. Instead, we assume that there exists a static discriminative proposal process that provides reasonable starting values or distributions over those values for some of the variables. In other words that we have an informative proposal for some of the messages,

$$q_{ij}(\mathbf{X}_i) = f(\mathbf{X}_i). \quad (3.99)$$

In computer vision applications, having a proposal function of this type is often relatively easy. For example, for articulated body pose estimation, where variables correspond to body parts, finding plausible positions for some variables corresponding to the placement of salient *limbs* or a *face* is a well researched problem (efficient and effective solutions to which exist).

3.7.4 Stratified Sampling

Stratified sampling (a.k.a. *proportional sampling*), involves dividing the samples into a set of homogeneous groups, and sampling within each group according to some function. While stratified sampling is of significant importance in probability and statistics by itself, here we will consider stratified sampling in the context of Monte Carlo methods and PAMPAS. Consider Monte Carlo importance sampling, first introduced in Section 3.6.1. As mentioned before, the efficiency of Monte Carlo approximation depends significantly on the importance function chosen. Sometimes, however, it is unclear which importance function is best suited for the inference task, and a number of alternative importance functions may be available. In particular, in Section 3.7.3 three importance functions have been introduced that are of interest in Particle Message Passing. The key observation is that instead of drawing all samples from one importance function that is believed to be most efficient, we can stratify the sampling procedure to draw samples from multiple importance functions. As a result, the samples will be more diverse overall, yet focused within each group.

Let us assume we have a family of R importance functions, $q_{ij}^{(r)}(\mathbf{X}_i)$, where $r \in [1, \dots, R]$ from which we want to draw N samples. If we also assume that each importance function has an associated sampling fraction γ_r , such that $\sum_{r=1}^R \gamma_r = 1$, then we can define a stratified sampler as follows:

$$\begin{aligned} s_{ij}^{(k)} &\sim q_{ij}^{(1)}(\mathbf{X}_i) && \text{for } k \in [1, \dots, N\gamma_1] \\ s_{ij}^{(k)} &\sim q_{ij}^{(2)}(\mathbf{X}_i) && \text{for } k \in [N\gamma_1 + 1, \dots, N\gamma_1 + N\gamma_2] \\ &\dots && \\ s_{ij}^{(k)} &\sim q_{ij}^{(r)}(\mathbf{X}_i) && \text{for } k \in \left[N \sum_{l=1}^{r-1} \gamma_l + 1, \dots, N \sum_{l=1}^r \gamma_l \right] \\ &\dots && \\ s_{ij}^{(k)} &\sim q_{ij}^{(R)}(\mathbf{X}_i) && \text{for } k \in \left[N \sum_{l=1}^{R-1} \gamma_l + 1, \dots, N \right] \end{aligned}$$

Since the samples, $\{s_{ij}^{(k)} | k \in [1, \dots, N]\}$, will be drawn from different importance functions, importance correction must also be done accordingly. In particular, if $q_{ij}^{(r)}(\mathbf{X}_i)$, $r \in [1, \dots, R]$, are the importance functions

for approximating message, $m_{ij}(\mathbf{X}_j)$, then the following importance correction must be applied,

$$\begin{aligned}
 w_{ij}^{(k)} &= \frac{m_{ij}^F(s_{ij}^{(k)})}{q_{ij}^{(1)}(s_{ij}^{(k)})} && \text{for } k \in [1, \dots, N\gamma_1] \\
 w_{ij}^{(k)} &= \frac{m_{ij}^F(s_{ij}^{(k)})}{q_{ij}^{(2)}(s_{ij}^{(k)})} && \text{for } k \in [N\gamma_1 + 1, \dots, N\gamma_1 + N\gamma_2] \\
 &\dots && \\
 w_{ij}^{(k)} &= \frac{m_{ij}^F(s_{ij}^{(k)})}{q_{ij}^{(r)}(s_{ij}^{(k)})} && \text{for } k \in \left[N \sum_{l=1}^{r-1} \gamma_l + 1, \dots, N \sum_{l=1}^r \gamma_l \right] \\
 &\dots && \\
 w_{ij}^{(k)} &= \frac{m_{ij}^F(s_{ij}^{(k)})}{q_{ij}^{(R)}(s_{ij}^{(k)})} && \text{for } k \in \left[N \sum_{l=1}^{R-1} \gamma_l + 1, \dots, N \right].
 \end{aligned}$$

In the above we assumed that $N\gamma_i$ is an integer for all i , in practice this it is often a fraction and must be rounded. For the stratified sampling to be effective, one must ensure that the number of groups (*strata*), S , is relatively small in relationship to the total number of samples, N . In addition, having widely disproportional fractions of samples may cause sampling artifacts. We found stratified sampling to be effective in PAMPAS. The full stratified sampling PAMPAS procedure is outlined in Algorithm 6.

3.7.5 Differences between PAMPAS and NBP

While the PAMPAS [99] algorithm introduced here and the Non-parametric Belief Propagation (NBP) algorithm introduced in [220] are very similar in nature, there are two key differences that are worth mentioning.

First, in [220] no particular form for the potentials is assumed. Hence, instead of propagating samples from the message foundation, $\{s_{ij}^{(n)} | n \in [1, \dots, N]\}$, via a potential resulting in convenient continuous representation for the message, in [220] $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ is sampled. This results in a particle representation for the message and kernel bandwidth estimation is used to assign equal variance bandwidth to all the samples. This leads to an additional approximation of $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$, where as in our case $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$, modeled using Gaussian mixtures⁷, can be represented exactly.

Second, there is a difference in where the importance sampling takes place due to the inability to represent likelihoods, $\phi_i(\mathbf{X}_i, \mathbf{Y})$, using convenient Gaussian mixture form. In [220] importance sampling and re-weighting is incorporated directly into the Gibbs sampler. This results in a generally better sampling strategy, however, requires an underlying assumption that the kernel width is small relative to the variations in the likelihood function $\phi_i(\mathbf{X}_i, \mathbf{Y})$. As a result, we believe that multiple hypothesis in the message foundation would tend to cause more severe problems in [220], rendering the algorithm of [220] inferior in cases where good initialization is unavailable. In PAMPAS, we need not make any assumptions on the kernel width and can represent the potential exactly, which makes it more convenient for the cases where only weak initialization is available. However, one would expect our approach to degrade as $m_{ij}^F(\mathbf{X}_i)$ and $m_{ij}^S(\mathbf{X}_i)$ become more dissimilar (*i.e.* more terms in the message will not have the convenient Gaussian mixture form), and in such cases NBP [220] may lead to superior performance.

⁷Other potential functions $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ from which conditional distributions of the form $\psi_{ij}(\mathbf{X}_i = x, \mathbf{X}_j)$ can be derived analytically, can also be represented exactly.

<p>Input: Graphical model $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with specified robust potential $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ (consisting of Gaussian mixtures with M_{ij} components and a single Gaussian outlier process) and likelihood $\phi_i(\mathbf{X}_i, \mathbf{Y})$ functions Set of possibly uninitialized messages $m_{ij}(\mathbf{X}_j)$ Number of samples to use for approximating the message, N.</p> <p>Output: Updated message $\tilde{m}_{ij}(\mathbf{X}_j)$</p> <ol style="list-style-type: none"> 1. Collect all terms in the message foundation $m_{ij}^F(\mathbf{X}_i) = \frac{1}{Z_{ij}} \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}(\mathbf{X}_i)$, that have convenient Gaussian mixture form into $m_{ij}^{FS}(\mathbf{X}_i)$ term. 2. Set importance functions and corresponding sampling fractions <ol style="list-style-type: none"> (a) $q_{ij}^{(1)}(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i)$ $q_{ij}^{(2)}(\mathbf{X}_i) = m_{ij}^{FS}(\mathbf{X}_i)m_{ji}(\mathbf{X}_i)$ $q_{ij}^{(3)}(\mathbf{X}_i) = f(\mathbf{X}_i)$ (b) For the first iteration of BP typically $\gamma_1 = 0, \gamma_2 = 0, \gamma_3 = 1$, for the rest typically $\gamma_1 = 0.5, \gamma_2 = 0.5, \gamma_3 = 0$. <p>where $f(\mathbf{X}_i)$ is the static proposal distribution.</p> 3. For each of the importance functions $k \in [1, \dots, 3]$ <ol style="list-style-type: none"> (a) Compute starting sample index, $N_s = \sum_{l=1}^{k-1} \frac{N\gamma_l}{M_{ij}}$ (b) Draw $N_k = (N-1)\gamma_k/M_{ij}$ samples from the proposal function: $s_{ij}^{(N_s+n)} \sim q_{ij}^{(k)}(\mathbf{X}_i), n \in [1, \dots, N_k] \quad (3.100)$ (c) Compute importance correction for $n \in [1, \dots, N_k]$ $w_{ij}^{(N_s+n)} = \frac{m_{ij}^F(s_{ij}^{(N_s+n)})}{q_{ij}^{(k)}(s_{ij}^{(N_s+n)})} \quad (3.101)$ 4. Assuming that we have a robust potential function for which conditional can be derived, e.g. for Gaussian mixture potential $\psi_{ij}(\mathbf{X}_j \mathbf{X}_i) = \lambda_0 \mathcal{N}(\mathbf{X}_j \mu_0, \Lambda_0) + (1-\lambda_0) \sum_{m=1}^{M_{ij}} \delta_{ijm} \mathcal{N}(\mathbf{X}_j F_{ijm}(\mathbf{X}_i), G_{ijm}(\mathbf{X}_i)), \quad (3.102)$ <p>store normalized weights and mixture components for $n \in [1, \dots, (N-1)/M_{ij}]$, $m \in [1, \dots, M_{ij}]$:</p> <ol style="list-style-type: none"> (a) $n' = (n-1)M_{ij} + m$ (b) $\mu_{ij}^{(n')} = F_{ijm}(s_{ij}^{(n)})$ (c) $\Lambda_{ij}^{(n')} = G_{ijm}(s_{ij}^{(n)})$ (d) $\pi_{ij}^{(n')} = (1-\lambda_0) \frac{w_{ij}^{(n)} \delta_{ijm}}{\sum_{l=1}^{N-1} w_{ij}^{(l)}}$ 5. Assign outlier components: $\pi_{ij}^{(N)} = \lambda_0, \mu_{ij}^{(N)} = \mu_0, \Lambda_{ij}^{(N)} = \Lambda_0$ 6. Let $\tilde{m}_{ij}(\mathbf{X}_j) = \sum_{n=1}^N \pi_{ij}^{(n)} \mathcal{N}(\mathbf{X}_j \mu_{ij}^{(n)}, \Lambda_{ij}^{(n)})$.
--

Algorithm 6: PAMPAS Stratified Message Update.

3.7.6 Message Passing Scheduling

While in theory the message passing schedule (order) in BP does not matter, in practice it has been shown that the message passing schedule can effect the convergence properties significantly. It is a well-known empirical observation that asynchronous message passing algorithms, where messages are updated sequentially, generally converge faster and more often than the synchronous variant, where all messages are updated in parallel. In practice, however, synchronous variants are often used, perhaps due to ease of implementation.

In tree-structured graphs the order in which messages should be sent is explicitly defined by the graph. In this case when sequential updating is used, the standard naive schedule is one where a message is propagated as soon as all of its inputs are available or have changed. This results in propagation of messages from the leaves of the tree upward toward the root and then back down.

In general loopy-graphs an explicit message passing schedule must be defined. The message passing schedule can be either synchronous or asynchronous. *Synchronous* message passing amounts to simultaneously sending messages along all edges of the graph. It has been shown, however, that often this results in very slow and inefficient convergence [56]. Alternatively, an *asynchronous* message passing schedule would lead to passing messages in a serial order defined by the schedule. One of the standard asynchronous message schedules can be derived by computing a minimum spanning tree over the graph and updating messages according to the tree-structure rules [239]. The spanning tree, however, may not be unique. In this case one must either choose a tree and a fixed asynchronous schedule for that tree, or for every iteration of BP randomly pick a minimum spanning tree and a corresponding schedule. In this thesis, we use a fixed asynchronous message passing schedule with a minimum spanning tree, for simplicity. In general, however, better convergence may be achieved by randomizing the tree parameterization and the message passing schedule. More recently a new informative message scheduling approach [56] has been proposed that schedules messages in an informed way, that pushes down a bound on the distance from the fixed point.

3.7.7 Simulated Annealing

The Markov chain based method of simulated annealing was developed initially in [116] and later adopted for articulated particle filtering in [52] and [70] as a way of handling multiple modes in a stochastic optimization context. The method employs a series of distributions, with probability densities given by $p_0(\mathbf{X})$ to $p_M(\mathbf{X})$, in which each $p_m(\mathbf{X})$, $m \in [0, \dots, M]$, differs only slightly from $p_{m+1}(\mathbf{X})$. In this context samples need to be drawn from $p_0(\mathbf{X})$ and $p_m(\mathbf{X})$'s are designed such that in $p_M(\mathbf{X})$ the movement between all regions of the search space are allowed. The usual method is to set $p_m(\mathbf{X}) \propto [p_0(\mathbf{X})]^{\beta_m}$, for $1 = \beta_0 > \beta_1 > \dots > \beta_M$.

In the case of Particle Message Passing (PAMPAS) one can anneal the likelihood, the potentials or both. In our experiments, we found that annealing the likelihood as a function of BP iterations worked well. We typically set $\beta_m = \beta_{m+1}\kappa$, where m is the iteration of BP and $0 < \kappa < 1$ is a constant. Simultaneous annealing of potentials is also possible and would lead to stronger joint constraints.

3.7.8 Examples

In this section we illustrate how Particle Message Passing can be used for inference in simple 1-D graphical models (*e.g.* HMMs). All examples have synthetically generated likelihood functions and hand specified

potentials. For experimental convenience and clarity, we use simple Gaussian likelihoods and potential functions (resulting in Gaussian conditionals), though our implementation of Particle Message Passing does not depend or make use of this fact for inference. We used $N = 1000$ samples to approximate messages and beliefs, in all cases. In all examples we are modeling a synthetically generated temporal evolution process of $\mathbf{X}_i \in \mathbb{R}^1, i \in [1, \dots, 5]$.

In Figure 3.15, inference in a directed Hidden Markov Model (illustrated in the top-left corner of the figure) is shown. The likelihoods for variables $\phi_i(\mathbf{Y}_i|\mathbf{X}_i) \equiv \mathcal{N}(\mathbf{X}_i | -7 + 7(i-1), 2) + \eta$, where $i \in [1, \dots, 5]$ and η is a zero mean Gaussian distributed noise with small (relative to the dynamics) variance. These likelihoods are illustrated by red $[\phi_1(\mathbf{Y}_1|\mathbf{X}_1)]$, green $[\phi_2(\mathbf{Y}_2|\mathbf{X}_2)]$, blue $[\phi_3(\mathbf{Y}_3|\mathbf{X}_3)]$, magenta $[\phi_4(\mathbf{Y}_4|\mathbf{X}_4)]$ and black $[\phi_5(\mathbf{Y}_5|\mathbf{X}_5)]$ accordingly. Inference in this model using PAMPAS is equivalent to sequential posterior estimation using a Particle Filter (see Section 3.6.4). Marginals corresponding to beliefs after 0–3 iterations of Particle Message Passing are illustrated in red corresponding to $b(\mathbf{X}_1)$, green – $b(\mathbf{X}_2)$, blue – $b(\mathbf{X}_3)$, magenta – $b(\mathbf{X}_4)$ and black to $b(\mathbf{X}_5)$. Since conditional distributions encoded by the edges between hidden nodes in the graph, illustrated in top-right corner of the figure, are very similar to the true dynamical model expressed by the synthetic observations, $\psi(\mathbf{X}_{i+1}|\mathbf{X}_i) \equiv \mathcal{N}(\mathbf{X}_{i+1}|\mathbf{X}_i + 7, 0.5)$, inference performs well.

In Figure 3.16, an undirected pair-wise MRF version of the graph corresponding to the same problem is shown. Unlike in Figure 3.15, bi-directional potentials (instead of conditional distribution) define evolution of states. In particular, $\psi(\mathbf{X}_i = x, \mathbf{X}_{i+1}) \equiv \mathcal{N}(\mathbf{X}_{i+1}|x + 7, 0.5)$ and $\psi(\mathbf{X}_i = x, \mathbf{X}_{i-1}) \equiv \mathcal{N}(\mathbf{X}_{i-1}|x - 7, 0.5)$, this is illustrated in the top-right corner of Figure 3.16. Similar, to the undirected case inferred distributions well match observations, because dynamics is modeled well. In Figure 3.17, inference with missing observations for \mathbf{X}_3 is shown. The rest of the model is the same as in Figure 3.16. As illustrated, temporal (if we assume that what is illustrated is a temporal process) consistency allows PAMPAS to correctly infer the state of all variable (including \mathbf{X}_3) in the presence of missing observations.

So far, both directed HMM and similar in structure undirected pair-wise MRF were able to produce similar inference results. To illustrate how the two models differ, we construct an example where dynamics embedded in the model is a very poor approximation to the true dynamics of the observed system. In Figure 3.18, the model is adjusted to have conditional distributions that poorly model true dynamics, $\psi(\mathbf{X}_{i+1}|\mathbf{X}_i) \equiv \mathcal{N}(\mathbf{X}_{i+1}|\mathbf{X}_i + 1, 0.5)$. In this case we can see that roughly after 3 time instances the algorithm loses track and the beliefs $b(\mathbf{X}_4)$ and $b(\mathbf{X}_5)$ poorly model the data. Interestingly enough if we try to perform the same inference task with an undirected model that has bi-directional constraints, the result is quite different (see Figure 3.16). In the undirected model, where inference is able to incorporate information from future observations, the distributions over all variables are adjusted to achieve best error averaged over all variables.

3.8 Discriminative Models

Lastly, we would like to introduce a few discriminative models that proved to be useful for articulated pose estimation [1, 2, 4, 206] and tracking [205, 206] (see Section 2.7 for further discussion). In the context of this thesis, these discriminative models will be useful in inference of 3D structure from the 2D pose, within the hierarchical framework that will be introduced in Chapter 6.

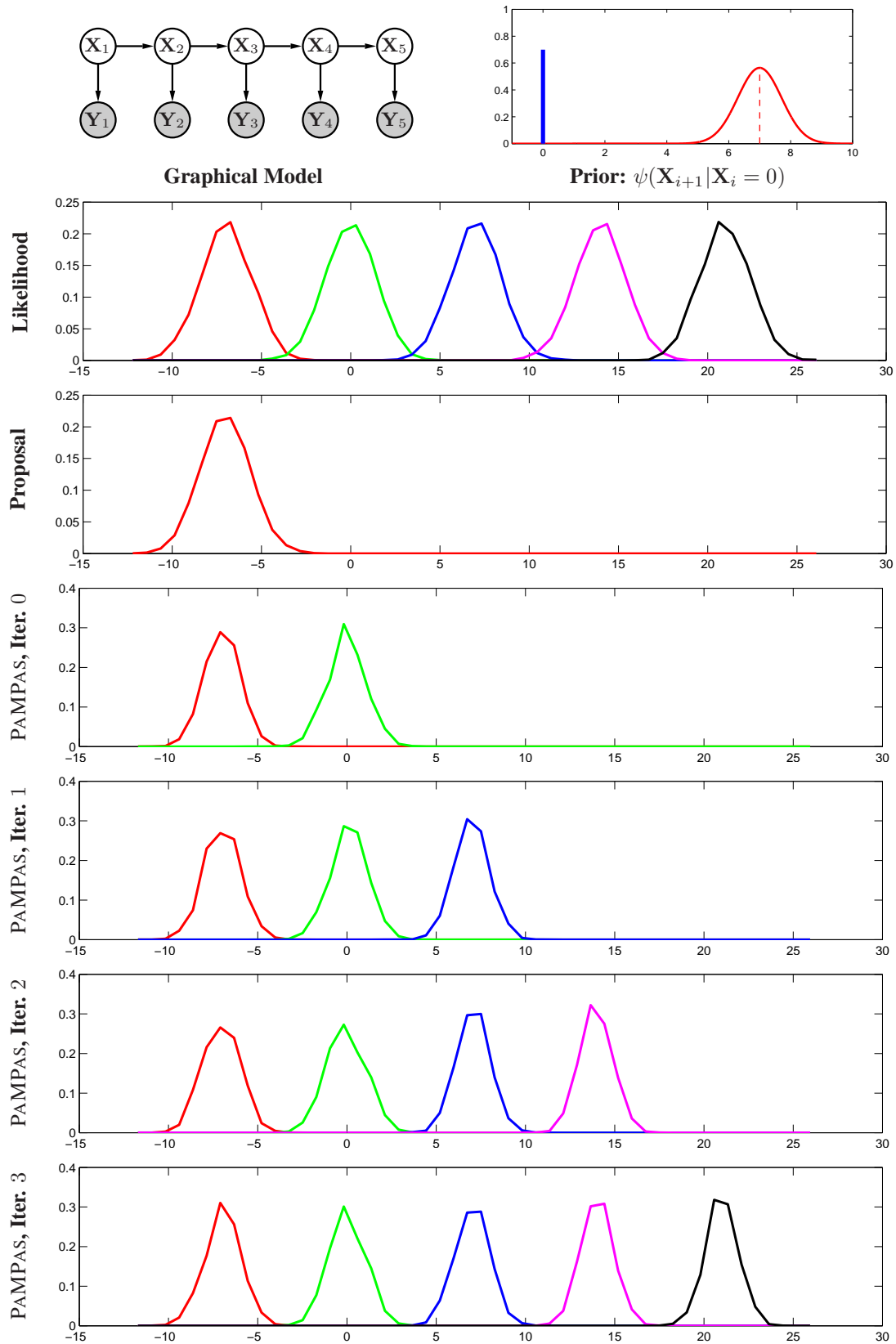


Figure 3.15: Particle Message Passing in Hidden Markov Model. See text for details.

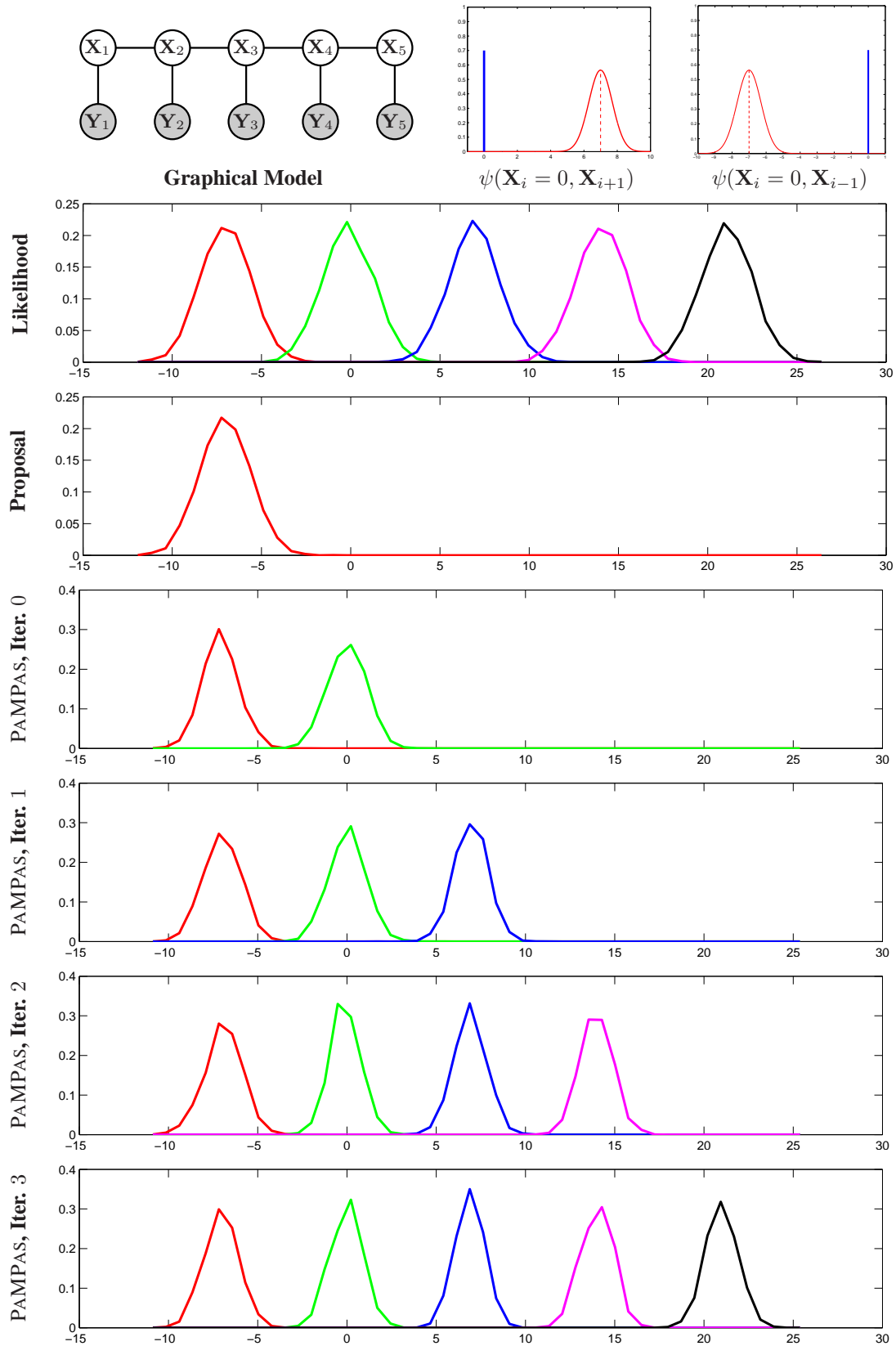


Figure 3.16: Particle Message Passing in pair-wise MRF. See text for details.

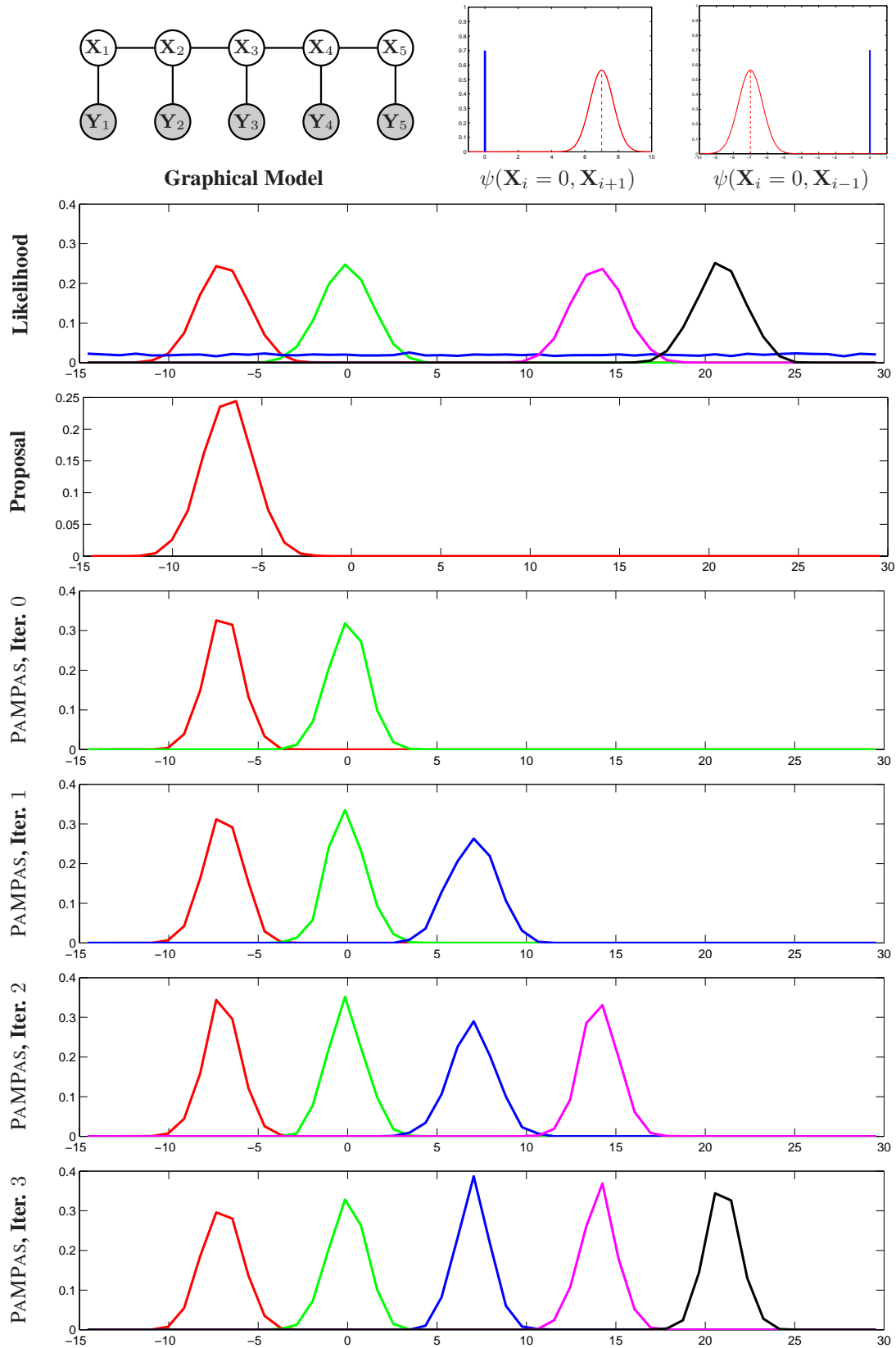


Figure 3.17: Particle Message Passing in pair-wise MRF with missing data. See text for details.

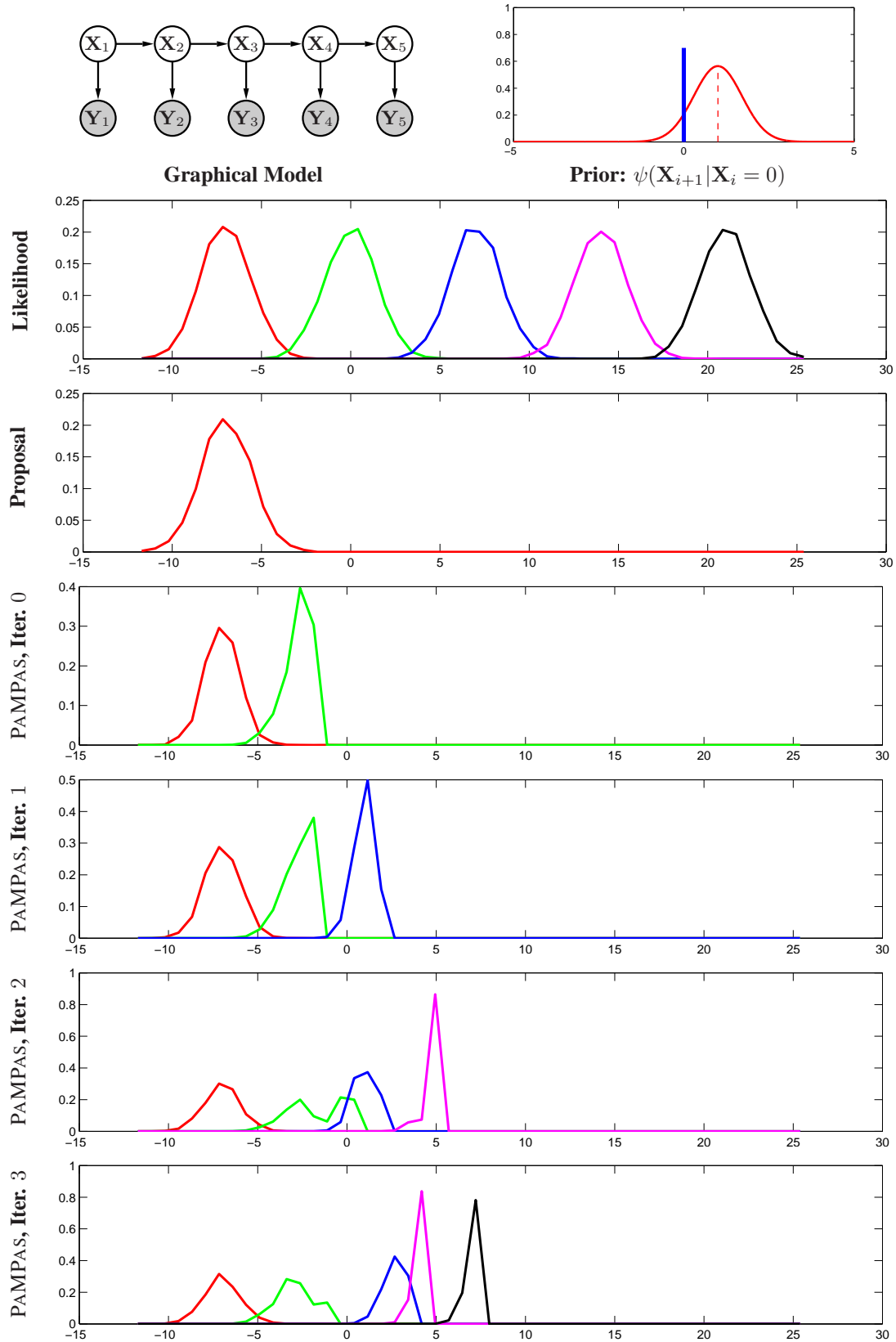


Figure 3.18: **Particle Message Passing in Hidden Markov Model (with poor dynamical prior).** See text for details.

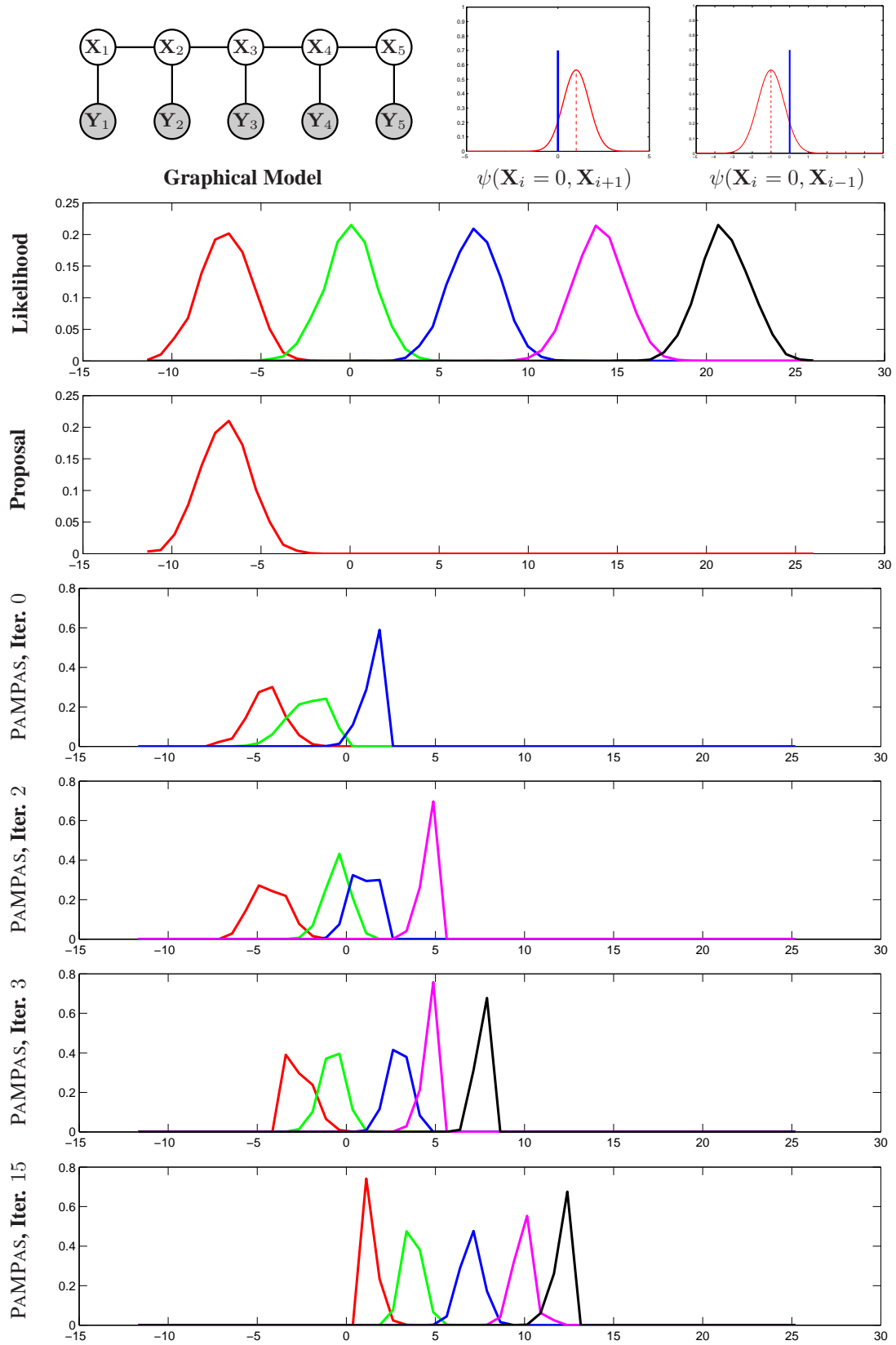


Figure 3.19: Particle Message Passing in pair-wise MRF (with poor dynamical prior). See text for details.

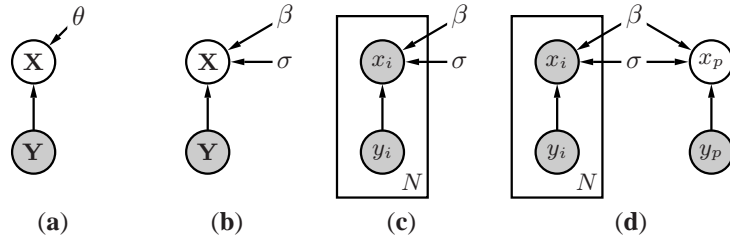


Figure 3.20: **Regression model.** In (a) a graphical model representation for Linear Regression is shown. In the regression model depicted, \mathbf{Y} corresponds to the independent input variable (observation) and \mathbf{X} to the dependent hidden output variable; θ is the set of parameters that are made explicit in (b). In (c) a graphical model representation for N *i.i.d.* input-output pairs of samples (x_i, y_i) drawn from the model is illustrated. In (d) a predictive regression model is shown, where given N *i.i.d.* observations as in (c) the goal is to predict a value for a latent variable x_p given a new observation y_p .

3.8.1 Linear, Ridge and Locally Weighted Regression

Linear Regression is among the simplest discriminative models that attempts to model the conditional $p(\mathbf{X}|\mathbf{Y})$ directly. The model assumes linear (or in case of *polynomial regression*, polynomial) relationship between multivariate random variables \mathbf{X} and \mathbf{Y} , *i.e.* $p(\mathbf{X}|\mathbf{Y}) = \mathcal{N}(\beta\mathbf{Y}, \sigma^2 I)$. In this model $\mathbf{X} \in \mathbb{R}^{d_{\mathbf{X}}}$ is the $d_{\mathbf{X}}$ -dimensional hidden variable and $\mathbf{Y} \in \mathbb{R}^{d_{\mathbf{Y}}}$ is the $d_{\mathbf{Y}}$ -dimensional observation. The relationship between \mathbf{X} and \mathbf{Y} can be expressed as $\mathbf{X} = \beta\mathbf{Y} + \eta$, where β is an $d_{\mathbf{Y}} \times d_{\mathbf{X}}$ matrix of regression coefficients and η is a zero mean normal noise variable with covariance $\sigma^2 I$ (please note that the basic model assumes that the noise across all dimensions is the same). Typically, with a regression model we want to (1) learn the parameters of the model $\theta = \{\beta, \sigma\}$ given a set of input-output paired observations (x_i, y_i) , and (2) given these parameters predict the value of (or distribution over) \mathbf{X} from new observations of \mathbf{Y} (see Figure 3.20 (d)).

In this chapter we take Bayesian approach to regression which is a generalization of the more typical least squares analysis⁸ formulation. Hence, to estimate parameters of a regression model we first must choose the hyper-prior over the parameters themselves. For example, if we choose a non-informative joint prior $p(\beta, \sigma) \propto \frac{1}{\sigma^2}$, the Maximum Likelihood (ML) estimates for the parameters can be obtained by maximizing the likelihood,

$$\mathcal{L}(\theta) = \mathcal{L}(\beta, \sigma) = p(\mathcal{D}|\beta, \sigma) = \prod_{i=1}^N (2\pi\sigma^2)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}(x_i - \beta y_i)^T(x_i - \beta y_i)\right], \quad (3.103)$$

with respect to the parameters $\theta = \{\beta, \sigma\}$ and subject to the training input-output pairs, $\mathcal{D} = \{(x_i, y_i)|i \in [1, \dots, N]\}$. The resulting estimate that can be re-written in terms of matrix notation (with slight abuse of notation where $\mathcal{D}_{\mathbf{X}} = \{x_i|i \in [1, \dots, N]\}$ and $\mathcal{D}_{\mathbf{Y}} = \{y_i|i \in [1, \dots, N]\}$) conform to the least-squares solution often obtained in non-Baysian setting:

⁸Least squares analysis is a method for linear regression that determines the values of unknown quantities in a statistical model by minimizing the sum of the residuals (difference between the predicted and observed values) squared. This method was first described by Carl Friedrich Gauss.

$$\hat{\beta}_{ML} = (\mathcal{D}_{\mathbf{Y}}^T \mathcal{D}_{\mathbf{Y}})^{-1} \mathcal{D}_{\mathbf{Y}}^T \mathcal{D}_{\mathbf{X}}, \quad \hat{\sigma}_{ML}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \beta y_i)^2. \quad (3.104)$$

Given the learned parameters, prediction of the distribution over \mathbf{X} for a new previously unobserved value of $\mathbf{Y} = y_p$ can trivially be computed,

$$p(\mathbf{X} | \mathbf{Y} = y_p) = \mathcal{N}(\mathbf{X}; \beta y_p, \sigma^2 I). \quad (3.105)$$

Ridge Regression

The non-informative prior for β , however, often results in the severe over-fitting and poor generalization. One way of battling this is to have a more informative prior on the parameters (*a.k.a.* regularization). In particular, if in addition to assuming that noise η is normally distributed with mean 0 and variance is $\sigma^2 I$ as before, we also assume that β has a prior distribution that is normal with mean 0 and variance $\frac{\sigma^2}{\lambda} I$, then the Maximum Likelihood estimator for the β becomes,

$$\hat{\beta}_{ML} = (\mathcal{D}_{\mathbf{Y}}^T \mathcal{D}_{\mathbf{Y}} + \lambda I)^{-1} \mathcal{D}_{\mathbf{Y}}^T \mathcal{D}_{\mathbf{X}}, \quad (3.106)$$

which corresponds to the *ridge regression* with regularization parameter λ . Intuitively this can be interpreted as adding smoothness constraints on the learned mapping, where λ is the damped regularization term that penalizes large values in the coefficient matrix β . Larger values of λ will result in overdamping, where the solution will be underestimated, small values of λ will result in overfitting and possibly ill-conditioning.

Ridge regression has been successfully applied, along with a closely related *relevance vector regression*, for discriminative articulated pose estimation by Agarwal *et al.* [1, 4]. The proposed approach uses simple histogram features based on local shape context (see Section 2.5.8) to learn a direct probabilistic mapping from these features to a full 3D pose of a person.

Locally Weighted Regression

In both linear regression and ridge regression all training input-output pairs, (x_i, y_i) , contribute equally to the learning of parameters. Often it is useful, however, to weight these contributions. The weight here can be interpreted as the probability that a particular training input-output pair came from the model (as opposed to the noise or an outlier process). Assuming that we can associate a weight, w_i , with every input-output pair, the equations for linear and ridge regression can be trivially augmented to account for this, *e.g.*

$$\hat{\beta}_{ML} = ([\mathcal{D}_{\mathbf{Y}}]^T W \mathcal{D}_{\mathbf{Y}} + \lambda I)^{-1} [\mathcal{D}_{\mathbf{Y}}]^T W \mathcal{D}_{\mathbf{X}}, \quad (3.107)$$

where $W = \text{diag}(w_1, \dots, w_N)$ is a diagonal matrix with corresponding weights. Notice that efficiency of the learning is a function of the number of non-zero diagonal elements in W .

3.8.2 Bayesian Mixture of Experts

Both linear regression and ridge regression assume that the mapping (conditional distribution) between the inputs (observed variables \mathbf{Y}) and outputs (latent variables \mathbf{X}) is linear and one-to-one. In many realistic

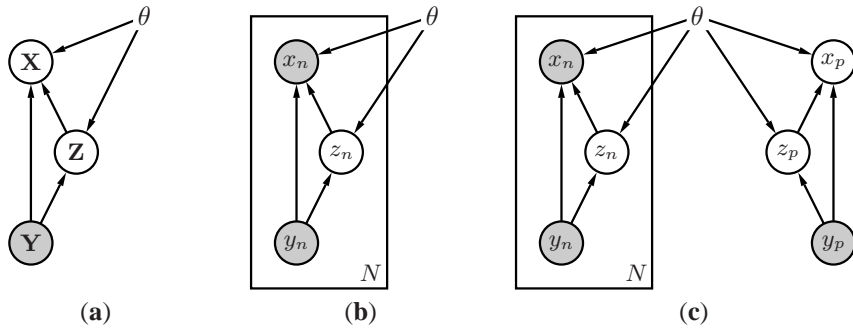


Figure 3.21: **Mixture of experts model.** In (a) a graphical model representation for Mixture of Experts (MoE) is shown. In the MoE model depicted, \mathbf{Y} corresponds to the independent input variable (observation) and \mathbf{X} to the dependent hidden output variable. \mathbf{Z} is the hidden variable corresponding to the activated gate, which is of little interest by itself and often is marginalized out to obtain desired conditional distribution $p(\mathbf{X}|\mathbf{Y})$. In (b) a graphical model representation for N *i.i.d.* input-output pairs of samples (x_i, y_i) drawn from MoE model are shown; corresponding latent gate variables z_i , are also illustrated. Finally in (c) a predictive MoE model is shown where given N *i.i.d.* observations as in (b) the goal is to predict a value for a latent variable x_p given a new observation y_p .

datasets this is not the case. For example, as was discussed in Section 2.9.2 the relationship between 2D features and 3D pose of the person is indeed multi-modal and not one-to-one, due to the projection ambiguities. In fact in many perception problems that involve the recovery of the inverse mapping, multi-modality arises naturally. To represent conditional distributions of this type *Bayesian Mixture of Experts* (BME) was introduced by Jacobs *et al.* in [103, 110] and Waterhouse in [242]. This model has since been used in many applications including human pose estimation [2, 195, 206] and tracking [206].

The key idea in BME is to use a Mixture Model, similar to the one described for Gaussian Mixture in Section 3.4.2, to combine multiple linear (or other type) discriminative models called *experts* into a single coherent probabilistic model. The rationale is that inputs will be assigned to individual experts probabilistically using a *gating* network, where upon each selected expert would be responsible for probabilistically predicting the outputs \mathbf{X} based on learned parameters. As a result some parts of the input space that are complex, would activate multiple experts resulting in the multi-modal distribution over the outputs \mathbf{X} , others that are unambiguous may be assigned to a single expert resulting in the simpler unimodal prediction. Formally the model can be written as follows (caring cunning resemblance to the Gaussian Mixture Model):

$$p(\mathbf{X}|\mathbf{Y}) = \sum_{\mathbf{Z}} p_e(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \theta_e) p_g(\mathbf{Z}|\mathbf{Y}, \theta_g) \quad (3.108)$$

or alternatively for M experts as,

$$p(\mathbf{X}|\mathbf{Y}) = \sum_{m=1}^M p_e(\mathbf{X}|\mathbf{Y}, z_m = 1, \theta_{e,m}) p_g(z_m = 1|\mathbf{Y}, \theta_{g,m}) \quad (3.109)$$

where $\mathbf{Z} = \{z_1, \dots, z_M\}$ is the set of hidden indicator variables that indicate which expert was responsible for generating the data point, $p_g(\mathbf{Z}|\mathbf{Y}, \theta_g)$ is the probabilistic gating network with parameters $\theta_g = \{\theta_{g,1}, \dots, \theta_{g,M}\}$, and $p_e(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \theta_e)$ is the set of experts with parameters $\theta_e = \{\theta_{e,1}, \dots, \theta_{e,M}\}$. This model

is illustrated in Figure 3.21. The choice of distributions for gate and experts is unconstrained by the basic model; the only condition that must hold is that $\sum_{m=1}^M p_g(z_m = 1 | \mathbf{Y}, \theta_{g,m}) = 1$.

Learning of Parameters. The overall parameters of the model including parameters of all experts and gaits can be expressed as $\theta = \{\theta_{g,m}, \theta_{e,m} | m \in [1, \dots, M]\}$. To learn these parameters, as before we must maximize the likelihood of observed data, \mathcal{D} . Let us assume that we have N *i.i.d.* input-output sample pairs $\mathcal{D} = \{(x_n, y_n) | i \in [1, \dots, N]\}$ that are generated by one of the M experts, selected using a set of hidden indicator variables $\{z_m^{(n)} | n \in [1, \dots, N], m \in [1, \dots, M]\}$, where

$$z_m^{(n)} = \begin{cases} 1 & \text{if expert } m \text{ generated } y_n \text{ from } x_n \\ 0 & \text{otherwise} \end{cases} \quad (3.110)$$

The likelihood that encodes the joint density for all N training samples can then be written as the following,

$$\mathcal{L}(\mathcal{D} | \theta) = \prod_{n=1}^N \sum_{m=1}^M p_e(x_n | y_n, z_m^{(n)} = 1, \theta_{e,m}) p_g(z_m^{(n)} = 1 | y_n, \theta_{g,m}). \quad (3.111)$$

Based on the formalism introduced in Section 3.4.1, we know that Maximum Likelihood estimation of parameters will not work in this case, and we will need to resort to Expectation-Maximization (EM) in order to deal with hidden variables, $z_m^{(n)}$.

The EM algorithm in this case would proceed to first estimate the posterior (in the E-step):

$$p(z_m^{(n)} = 1 | x_n, y_n, \theta) = \frac{p_e(x_n | y_n, z_m^{(n)} = 1, \theta_{e,m}) p_g(z_m^{(n)} = 1 | y_n, \theta_{g,m})}{\sum_{i=1}^M p_e(x_n | y_n, z_i^{(n)} = 1, \theta_{e,i}) p_g(z_i^{(n)} = 1 | y_n, \theta_{g,i})}. \quad (3.112)$$

This gives probability that expert m has generated the data pair (x_n, y_n) . In the M-step one must optimize both parameters of the experts and of the corresponding gates. This amounts to first learning the parameters of the experts, that must account for the current expert membership estimates $z_m^{(n)}$; and then learning parameters of the gates. The latter learning step must account for how well a given learned expert can predict the output for an input value.

Notice that so far we have not explored particular choices for the gates or experts and the above EM algorithm is very general. The particular choice of the gate functions and experts will, however, greatly impact the efficiency and overall complexity of the learning tasks in the M-step.

Alternatively, for a restrictive choice of gate and expert functions the desired conditional can be computed *indirectly* from the joint [2, 167, 207], via Bayes' rule. This results in often simpler M-step, at the expense of higher dimensional modeling of the joint distribution. The benefit of the indirect methods is that, for this restricted choice of gate and expert functions, some of the computation (most notably marginalization over the inputs) can be done analytically. Learning of parameters using both direct EM and indirect methods can be made more efficient by enforcing sparsity priors on the inputs [179]. A more detailed discussion of these issues can be found in [179].

In this thesis we will make use of the MoE architecture with a particular simplified choice of gate and expert functions. In particular, we will use Gaussian gates and linear (or ridge) with Gaussian kernel regression experts, *i.e.*

$$p_g(z_m = 1 | \mathbf{Y}, \theta_{g,m}) = \frac{\mathcal{N}(\mathbf{Y} | \mu_m, \Sigma_m)}{\sum_{i=1}^M \mathcal{N}(\mathbf{Y} | \mu_i, \Sigma_i)} \quad (3.113)$$

$$p_e(\mathbf{X} | \mathbf{Y}, z_m = 1, \theta_{e,m}) = \mathcal{N}(\mathbf{X} | \beta_m \mathbf{Y}, \Lambda_m). \quad (3.114)$$

This specific form of the MoE model is often called *Mixture of Regressors* [2, 207]. Both direct and indirect methods for learning parameters of Mixture of Regressors exist [207]. The direct method that uses EM and weighted regression to optimize the expert parameters, will be discussed in detail in Section 6.4. The indirect method as an exercise is presented in the next section.

Prediction. Once the parameters are learned, as with regression we are interested in predicting the output x_p for some previously unobserved input value y_p . This can be formulated as,

$$p(x_p | y_p, \theta) = \sum_{m=1}^M p_e(x_p | y_p, z_m = 1, \theta_{e,m}) p_g(z_m = 1 | y_p, \theta_{g,m}). \quad (3.115)$$

A point estimator can be obtained by taking expected value of the above,

$$\hat{x}_p = \mathbb{E}[p(x_p | y_p, \theta)], \quad (3.116)$$

where \mathbb{E} is conditional expectation.

Examples. In Figures 3.22–3.24 we illustrate the Mixture of Experts architecture on a set of simple 1D examples. Figure 3.22 shows how a mixture of linear kernel regressors can be used to learn a non-linear probabilistic function. Figure 3.23 shows an example where similar architecture is used to learn multi-modal prediction functions. In all examples direct EM-based learning method was used and iterated for a fixed number of steps (10). To initialize the EM learning joint input-output vectors, (x_i, y_i) , were clustered using K-means and a separate expert was assigned to each cluster.

3.8.3 Joint-based Learning for Mixture of Regressors

Let us consider the special case of the Mixture of Experts model, called Mixture of Regressors, where the experts are kernel regressors of the form $p_e(\mathbf{X} | \mathbf{Y}, z_m = 1, \theta_{e,m}) = \mathcal{N}(\mathbf{X} | \beta_m \mathbf{Y}, \Lambda_m)$, where β_m is as before (see Section 3.8.1) the matrix of regression coefficients and Λ_m is the corresponding Gaussian kernel bandwidth (covariance matrix) for expert $m \in [1, \dots, M]$. The joint distribution for this choice of the expert can be written in the following form:

$$p \left(\begin{bmatrix} \mathbf{Y} \\ \mathbf{X} \end{bmatrix} | \theta \right) = \sum_{m=1}^M \delta_m \mathcal{N} \left(\begin{bmatrix} \mathbf{Y} \\ \mathbf{X} \end{bmatrix} \mid \begin{bmatrix} \mu_m \\ \beta_m \mathbf{Y} \end{bmatrix}, \begin{bmatrix} \Sigma_m & \Sigma_m \beta_m^T \\ \beta_m \Sigma_m & \beta_m \Sigma_m \beta_m^T + \Lambda_m \end{bmatrix} \right). \quad (3.117)$$

The conditional distribution can be derived analytically from above using the rules of conditional Gaussian distributions introduced in Section 3.1.2,

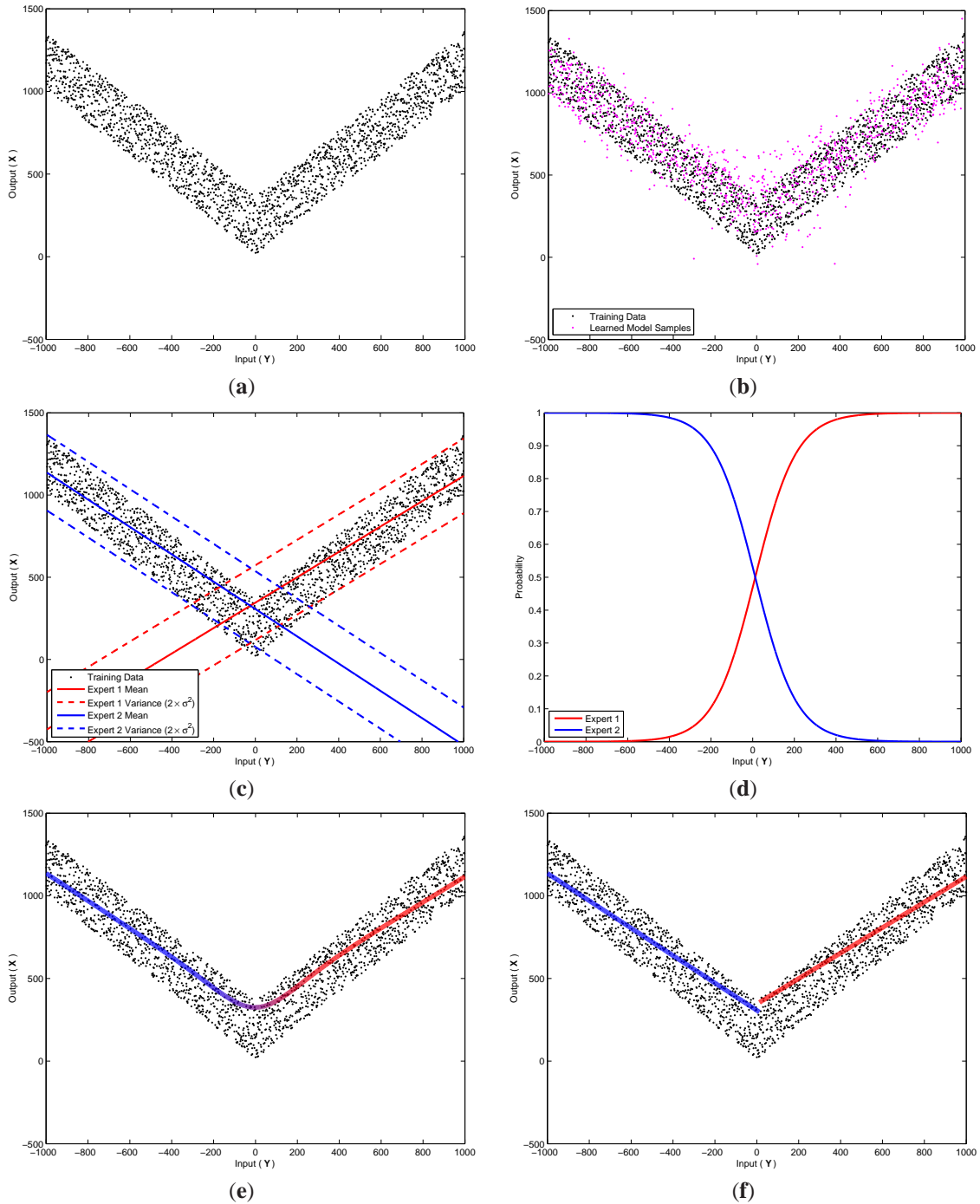


Figure 3.22: **Mixture of kernel regressors example.** A special case of the Mixture of Experts (MoE) model, mixture of linear kernel regressors, is illustrated. The training data, consisting of 1D input (along x -axis) and 1D output (along y -axis) paired samples, is illustrated in (a). Learned model consisting of a mixture of $M = 2$ regressors is illustrated in (b, c, d), where (b) illustrates samples drawn from the model (in magenta); (c) and (d) individual kernel regressor experts and corresponding gates as a function of the input. Point predictions for the range of inputs using the learned model are illustrated in (c) and (d). In (c) weighted prediction corresponding to the conditional expectation in Eq. 3.116 is shown; color designates contribution of individual experts towards the solution. Finally, in (d) prediction based on the most probable expert are shown for the range of inputs; color designates the expert used. Notice that mixture of linear experts in this example are capable of modeling non-linear condition distribution.

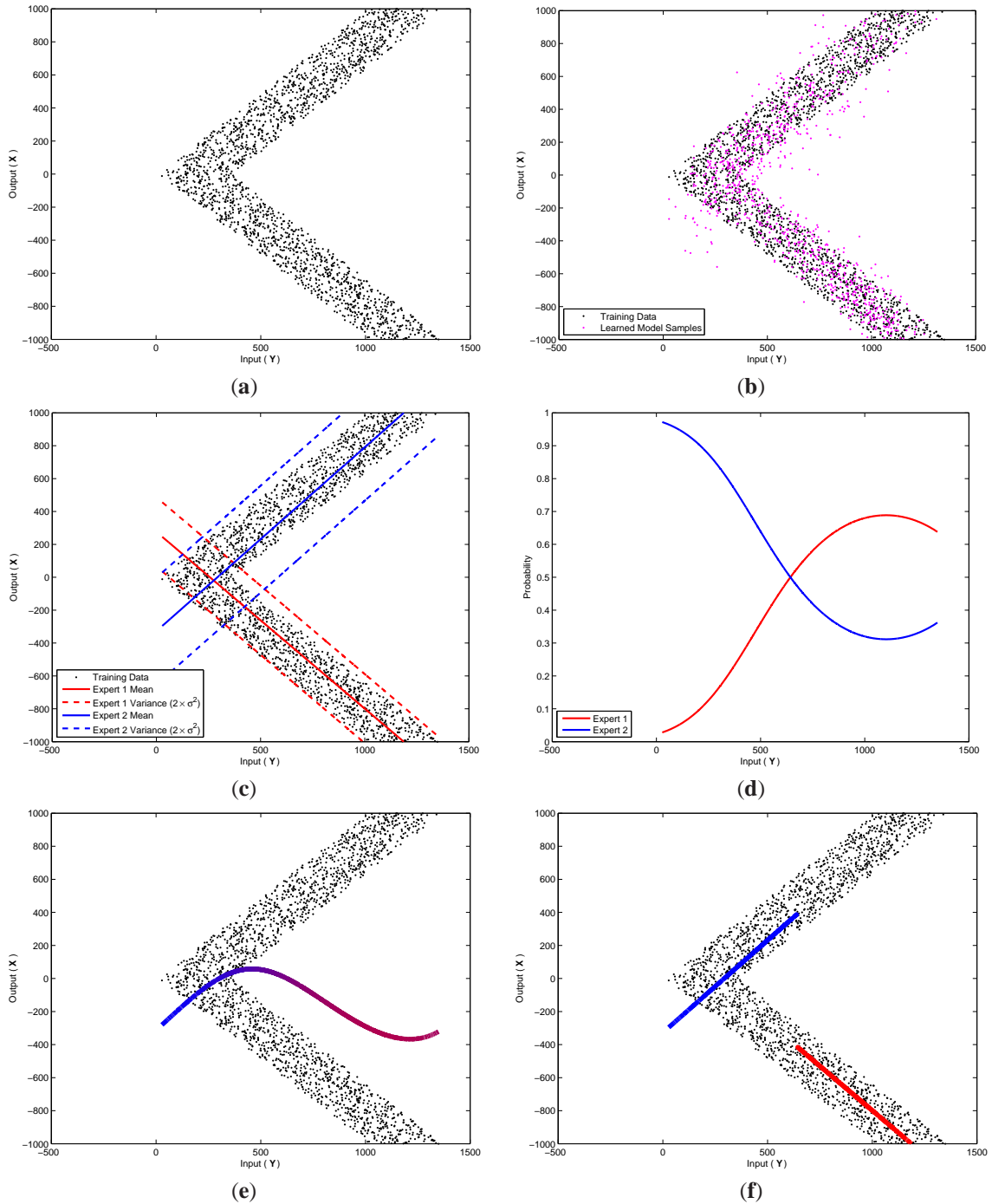


Figure 3.23: **Mixture of kernel regressors example.** A special case of the Mixture of Experts (MoE) model, mixture of kernel linear regressors, is illustrated. The training data, consisting of 1D input (along x -axis) and 1D output (along y -axis) paired samples, is illustrated in (a). Learned model consisting of a mixture of $M = 2$ regressors is illustrated in (b, c, d), where (b) illustrates samples drawn from the model (in magenta); (c) and (d) individual kernel regressor experts and corresponding gates as a function of the input. Point predictions for the range of inputs using the learned model are illustrated in (c) and (d). In (c) weighted prediction corresponding to the conditional expectation in Eq. 3.116 is shown; color designates contribution of individual experts towards the solution. Finally, in (d) prediction based on the most probable expert are shown for the range of inputs; color designates the expert used. Notice that while point estimates cannot deal well with multimodal predictions, multimodality is correctly encoded by the model (see (b)).

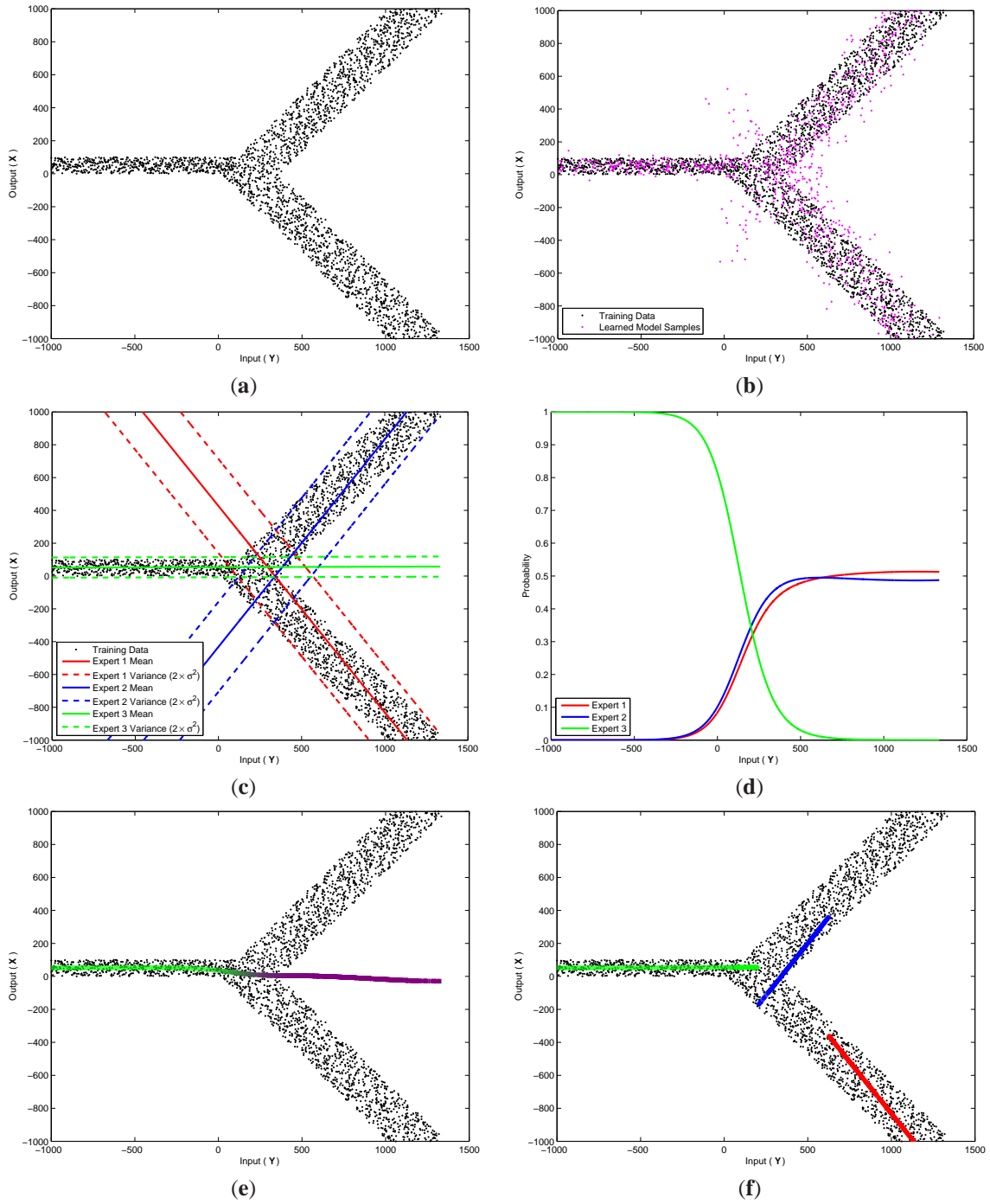


Figure 3.24: **Mixture of kernel regressors example.** A special case of the Mixture of Experts (MoE) model, mixture of kernel linear regressors, is illustrated. The training data, consisting of 1D input (along x -axis) and 1D output (along y -axis) paired samples, is illustrated in (a). Learned model consisting of a mixture of $M = 3$ regressors is illustrated in (b, c, d), where (b) illustrates samples drawn from the model (in magenta); (c) and (d) individual kernel regressor experts and corresponding gates as a function of the input. Notice that different experts have different variances estimated according to the corresponding data. Point predictions for the range of inputs using the learned model are illustrated in (c) and (d). In (c) weighted prediction corresponding to the conditional expectation in Eq. 3.116 is shown; color designates contribution of individual experts towards the solution. Finally, in (d) prediction based on the most probable expert are shown for the range of inputs; color designates the expert used.

$$\begin{aligned}
p(\mathbf{X}|\mathbf{Y}, \theta) = \sum_{m=1}^M \underbrace{\frac{\delta_m \mathcal{N}(\mathbf{Y}|\mu_m, \Sigma_m)}{\sum_{i=1}^M \delta_i \mathcal{N}(\mathbf{Y}|\mu_i, \Sigma_i)}}_{\text{Gate: } p_g(\mathbf{Z}|\mathbf{Y}, \theta_g)} \underbrace{\mathcal{N}(\mathbf{X}|\beta_m \mathbf{Y}, \Lambda_m)}_{\text{Expert: } p_e(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \theta_e)}
\end{aligned} \tag{3.118}$$

where parameters of the gates, $\theta_g = \{\delta_m, \mu_m, \Sigma_m | m \in [1, \dots, M]\}$, and experts, $\theta_e = \{\beta_m, \Lambda_m | m \in [1, \dots, M]\}$, are easily derived from the joint in Eq. 3.117. Full proof of this is given in [207]. Hence, to learn this restricted form of the Mixture of Experts (MoE) model it is sufficient to learn the Mixture of Gaussians (MoG) representation of the joint with the number of mixture component, M , equal to the number of experts required. The MoE model can then be obtained from the Mixture of Gaussians using simple analytic computations.

CHAPTER 4

Graphical Object Models

The previous chapter introduced the general mathematical and computational tools used in this thesis. In this chapter we leverage these tools to address the problem of object detection and tracking. In doing so we introduce a novel probabilistic framework for automatic component-based detection and tracking of generic objects in images and/or video. By combining object detection with tracking in this unified framework we can achieve a more robust solution for both problems. Tracking can make use of object detection for initialization and re-initialization during transient failures or occlusions, while object detection can be made more reliable by considering the consistency of the detection over time. Modeling objects by an arrangement of image-based (possibly overlapping) components, facilitates detection of complex articulated objects, as well as helps in handling partial object occlusions or local illumination changes. For simplicity, in this chapter we first introduce the proposed framework in the context of localization and tracking of simpler rigid objects; we will then extend the proposed framework in Chapters 5 and 6 to deal with pose of more complex articulated objects (people).

Object detection and tracking is formulated as inference in a two-layer graphical model in which the coarse layer node(s) represent(s) the whole object and the fine layer nodes represent multiple component “parts” of the object. Undirected edges between nodes represent learned spatial and temporal probabilistic constraints. Each node in the graphical model corresponds to a position and scale of the component or the object as a whole in an image at a given time instant. Each node also has an associated AdaBoost detector that is used to define the local image likelihood and a proposal process.

In general the likelihoods and dependencies are not Gaussian. There are at least two reasons that can lead to non-Gaussianity of component dependencies: (1) if there is indeed more than one mode for the statistical relationship between components (2) if one wants to reuse parts for efficiency (*e.g.* for a side view of a car we can model both tires using a single ‘tire’ component and then have a bi-modal spatial distribution for a car position and orientation, where bi-modality will get resolved by other spatial constraints). To infer the 2D position and scale at each node we exploit a form of non-parametric belief propagation (NBP), introduced in Section 3.7, that uses a variation of particle filtering and can be applied over a loopy graph [99, 220].

The problem of describing and recognizing categories of objects (*e.g.* faces, people, cars) is central to computer vision. It is common to represent objects as collections of features with distinctive appearance, spatial extent, and position [33, 61, 144, 235, 236]. There is, however, a large variation in how many features one must use and how these features are detected and represented. Most algorithms rely on semi-supervised



Figure 4.1: **Variation within the class of vehicles.** Three instances of vehicles are shown, with two different types of vans on the left and middle and a smaller passenger car on the right. While vehicles shown here have a drastically different appearance as a whole, due to the varying height and type of the vehicle, their components, illustrated by red and green rectangles, tend to be very homogeneous and are easy to model. The components, for convenience, are also illustrated separately to the right of each corresponding vehicle. Notice that components corresponding to the top-left corner of a vehicle, all have distinctive 90 degrees rotated ‘L’ shaped open contour structure; components corresponding to the lower portion of vehicles have a distinctive tire profile in all cases. The relative position of these components is, however, different in each case.

learning [144, 235, 236] schemes where examples of the desired class of objects must be manually aligned, and then learning algorithms are used to automatically select the features that best separate the images of the desired class from background image patches. More recent approaches learn the model in an unsupervised fashion from a set of unlabeled and unsegmented images [33, 61, 204]. In particular, Fergus *et al.* [61] develop a component based object detection algorithm (*a.k.a.* constellation model) that learns an explicit spatial relationship between parts of an object, but unlike our framework assumes Gaussian likelihoods and spatial relationships. In addition, in [61], as in many other approaches [33, 144, 204, 236], temporal consistency is ignored. Also, the computational complexity of the constellation model is exponential in the number of parts encoded by the model, as opposed to the linear complexity of the model proposed here. For further details on the constellation model and analysis of complexity please see Section 2.11.2.

In contrast to part-based representations, simple discriminative classifiers treat an object as a single image region. Boosted classifiers [236], for example, while very successful tend to produce a large set of false positives. This problem can be reduced by incorporating temporal information [235]. Discriminative classifiers based on boosting, however, do not explicitly model parts or components of objects. Such part-based models are useful in the presence of partial occlusions, out-of-plane rotation and/or local lighting variations [59, 144, 249]. Part- or component-based detection is also capable of handling highly articulated objects, for which a single appearance model classifier may be hard to learn. An illustration of the usefulness of component-based detection for vehicles is shown in Figure 4.1.

Murphy *et al.* [152] also use graphical models in the patch-based detection scheme. Unlike our approach they do not incorporate temporal information or explicitly reason about the object as a whole. Also closely related is the work of [157] which uses AdaBoost for multi-target tracking and detection. However, their Boosted Particle Filter [157] does not integrate component-based object detection and is limited to temporal propagation in only one direction (forward in time). In contrast to these previous approaches we combine techniques from discriminative learning, graphical models, belief propagation, and particle filtering to achieve reliable multi-component object detection and tracking.

4.1 AdaBoost

AdaBoost [67] is a supervised machine learning procedure, that given a set of positive and negative example patterns (in our case image regions [236]), learns a binary classification function for the two classes. More recently AdaBoost formulation has been extended to multi-class classification [223] problems. In general, AdaBoost is an algorithm that is used to *boost* classification performance of a simple classifier. This is achieved by combining a collection of weak classifiers to form a (better) strong classifier. A weak classifier (*a.k.a. weak learner*) is a classification function that is not expected to classify the data well even with the best choice of features and parameters. For boosting to work, however, the weak classifier is expected to perform better than chance classification (*i.e.* classify a given image pattern correctly more than 50% of the time). Often weak classifiers are chosen to be simple functions that operate on individual features; AdaBoost is then used to both select the features and train the classifiers based on these features.

The AdaBoost learning procedure works as follows. First, the feature and the weak classifier based on this feature are selected to ensure the best possible separation between positive and negative examples. After this first *round of boosting*, the examples are re-weighted to emphasize those that were misclassified by the selected weak classifier. The second *round of boosting* then selects a weak classifier that performs better on the examples that were misclassified. This can be repeated for K rounds, producing the final strong classifier that is the weighted sum of the responses from the K weak classifiers selected along the way. The relative weighting of the weak classifiers is also estimated, based on the misclassification error.

There are relatively strong guarantees for AdaBoost learning. It has been shown that training error of the strong classifier approaches zero exponentially in the number of boosting rounds [188]. Theoretic bounds on generalization can also be found in [188]. In particular, Schapire *et al.* [188] proved that AdaBoost aggressively reduces the margin of the decision boundary (since it concentrates on examples with smallest margin). It has also been shown theoretically [66] that AdaBoost will overfit if run for too many boosting rounds. It is worth mentioning that there is a strong connection between the theoretic results obtained for boosting and the support-vector machines introduced by Vapnik [229, 230] and others. We refer the reader to [188] for more details on theoretic guarantees of AdaBoost.

The conventional AdaBoost procedure can be interpreted as a greedy feature selection process. In the more general *boosting* framework, the goal is to combine a large set of classification functions using a weighted majority vote. The challenge is to associate the set of good classification functions with large weights and conversely the set of poor classification functions with zero or negligible weights. AdaBoost is a greedy mechanism for selecting a small set of good classification functions (or features) that in combination can be used to classify relatively complex patterns.

AdaBoost performs well when the classification functions are simple, and tends to have little or no benefit (due to overfitting) when the classification functions are complex and can deal with classification task effectively by themselves. Because of this often in AdaBoost simple classifiers that are functions of individual features are used. For the purposes of this thesis we will use weak classifiers similar to the ones introduced in [236]. We define a weak classifier $h_j(I)$ that consists of a feature $f_j(I)$ computed on the sub-window of the image I as

$$h_j(I) = \begin{cases} 1 & \text{if } p_j \sqrt[\beta_j]{[f_j(I)]^{\beta_j}} < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

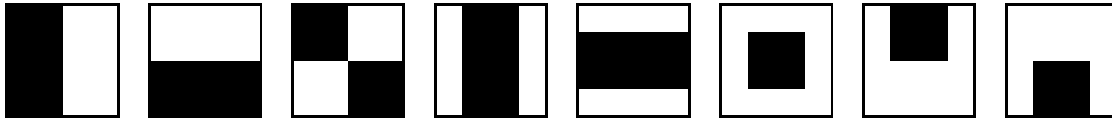


Figure 4.2: **AdaBoost filters.** AdaBoost features are obtained by convolving the image with the Haar-wavelet-like filters, illustrated above, at a given image location (x, y) and scale (w, h) .

where p_j is the polarity indicating the direction of inequality, and $\beta_j \in [1, 2]$ is a parameter allowing for a symmetric two sided pulse classification. The feature $f_j(I)$ is computed by convolving the sub-window¹ of the image I with the delta function over the extent of a spatial template. An over-complete set of spatial templates are defined based on the canonical Haar-wavelet-like features shown in Figure 4.2.

Given a set of labeled patterns the AdaBoost procedure learns a weighted combination of weak classifiers defined by Eq. (4.1),

$$h(I) = \sum_{k=1}^K \alpha_k h_k(I), \quad (4.2)$$

where I is an image, and $h_k(I)$ is the weak classifier chosen for the round k of boosting, and α_k is the corresponding weight. The full AdaBoost procedure is outlined in Algorithm 7. The output of the AdaBoost classifier is a confidence $h(I)$ that the given pattern I is of the desired class. It is customary to consider an object present if $h(I) \geq \frac{1}{2} \sum_{k=1}^K \alpha_k$. In the context of this thesis we use AdaBoost not to classify individual image patterns, but instead to define a rich discriminative likelihood for the patterns as will be further described in Section 4.2.3.

4.1.1 Bootstrapping

The performance of the AdaBoost procedure described in the previous section depends on the positive and negative sets of examples with which the classifier is trained. While collecting good positive examples is at least in principle simple (by supervised labeling), collecting good negative examples is harder. Particularly because the good negative examples we are after are those that visually resemble the object of interest, with respect to the features chosen. Such negative examples will emphasize the distinctions between the object and non-object classes leading to better performance and lower false positive rates (that are common with AdaBoost). In addition, the number of negative examples must be comparable to the number of positive examples collected, to reduce classification bias.

Bootstrapping is an effective iterative two-stage procedure for collecting negative examples. First, a preliminary set of negative examples is collected at random from a set of images that do not contain the object. Based on this preliminary negative set and labeled positive set, a classifier is learned using the AdaBoost algorithm outlined in Section 4.1. This classifier is then run over a collection of images that do not contain the object of desired class. A fixed set of regions that give high response are then collected and amended to

¹The notation used for features, $f_j(I)$, is somewhat of a shorthand. In practice, j ranges over the types of spatial templates $b \in [1, \dots, 8]$ (see Figure 4.2), possible discrete locations, (x, y) , where the template can be applied within an image I and the discrete scale of the template, (w, h) . Hence, $j \in [b, x, y, w, h]^T$, leading to a large collection of features (typically tens or hundreds of thousand).

Input: Example greyscale images $x_i \in \mathbb{R}^{d_w \times d_h}$ with associated binary labels $y_i \in \{0, 1\}$ (i.e. N labeled image patterns $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$)

Output: Strong classifier, $h(x)$, capable of classifying a greyscale image $x \in \mathbb{R}^{d_w \times d_h}$, where d_w is the width and d_h the height of the image in question.

1. Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for positive and negative examples respectively, where m is the total number of positive examples ($y_i = 1$) and l is the total number of negative examples ($y_i = 0$).
2. For each round of boosting $k = 1, \dots, K$

- (a) Normalize the weights for all patterns $i \in [1, \dots, N]$,

$$w_{k,i} = \frac{w_{k,i}}{\sum_{j=1}^N w_{k,j}} \quad (4.3)$$

- (b) For each feature, j , train a weak classifier $h_j(I)$ of the form:

$$h_j(I) = \begin{cases} 1 & \text{if } p_j \sqrt[p_j]{[f_j(I)]^{p_j}} < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

Also, compute the error of classification with respect to the weighted set of examples,

$$\epsilon_j = \sum_{i=1}^N w_{k,i} |h_j(x_i) - y_i|. \quad (4.5)$$

- (c) Choose classifier $h_k(x)$ with the lowest error ϵ_k .
- (d) Update weights for all examples $i \in [1, \dots, N]$ to give higher weight to misclassified examples,

$$w_{k+1,i} = w_{k,i} \left[\frac{\epsilon_k}{1 - \epsilon_k} \right]^{1-e_i} \quad (4.6)$$

where $e_i = 0$ if the example x_i was classified correctly and $e_i = 1$ otherwise.

3. The final strong classifier is defined as follows:

$$h(I) = \begin{cases} 1 & \text{if } \sum_{k=1}^K \alpha_k h_k(I) \geq \frac{1}{2} \sum_{k=1}^K \alpha_k \\ 0 & \text{otherwise} \end{cases} \quad (4.7)$$

where $\alpha_k = \log \frac{1-\epsilon_k}{\epsilon_k}$.

Algorithm 7: AdaBoost Classifier Learning.

the old negative set. A new classifier can then be learned based on the new augmented negative set and the old positive set of examples. As this process is iterated, with every iteration of bootstrapping, the negative examples become more alike the object. This is effective, however, if the number of bootstrapping iterations is high, an inverse effect can be achieved. In particular, as the negative examples become very similar to the images of the object, the decision boundary between the two classes becomes poorly defined. It can be empirically shown that the classification error of weak classifiers approaches 50% as the negative examples

<p>Input: Example positive greyscale images $x^{(p)} = \{x_i^{(p)} i \in [1, \dots, N_p]\}$, $x_i^{(p)} \in \mathbb{R}^{d_w \times d_h}$. Set of greyscale images that do not contain the object $\xi = \{\xi_1, \xi_2, \dots\}$</p> <p>Output: Strong classifier, $h(x)$, capable of classifying greyscale image $x \in \mathbb{R}^{d_w \times d_h}$, where d_w is the width and d_h the height of the image in question.</p> <ol style="list-style-type: none"> 1. Randomly select a set of $N_{n,1}$ negative images, $x^{(n,1)} = \{x_i^{(n,1)} i \in [1, \dots, N_{n,1}]\}$, from ξ. 2. For each bootstrap iteration $m = 1, \dots, M$ <ol style="list-style-type: none"> (a) Let negative set of examples be $x^{(n)} = \{x^{(n,i)} i \in [1, \dots, m]\}$, where the total number of negative examples is $N_n = \sum_{i=1}^m N_{n,i}$. For convenience, we also can refer to the individual elements of the negative set using the following notation $x_i^{(n)}$, where $i \in [1, \dots, N_n]$. (b) Train AdaBoost strong classifier $h(x)$ based on the following examples: $\{(x_1^{(p)}, 1), \dots, (x_{N_p}^{(p)}, 1), (x_1^{(n)}, 0), \dots, (x_{N_n}^{(n)}, 0)\} \quad (4.8)$ using procedure outlined in Algorithm 7. (c) Run classifier $h(x)$ on images in ξ and obtain a set, $x^{(n,m+1)} = \{x_i^{(n,m+1)} i \in [1, \dots, N_{n,m+1}]\}, \quad (4.9)$ of $N_{n,m+1}$ false positives. 3. Return the last trained $h(x)$.
--

Algorithm 8: Bootstrap Learning of AdaBoost Classifier.

become more like the object of interest, and therefore boosting stops being effective in practice. The full bootstrap procedure is outlined in Algorithm 8.

4.2 Graphical Object Models

In our framework we model an object using a spatio-temporal undirected graphical model. Each node in the graph represents either the object or a component of the object at time t . Nodes have an associated state vector $\mathbf{X} = [x, y, s]^T \in \mathbb{R}^3$ defining the component's real-valued position, (x, y) , and scale, s , within an image. The joint probability distribution for this spatio-temporal graphical object model with N components and over T frames can be written as:

$$\begin{aligned}
 p(\mathbf{X}_1^O, \mathbf{X}_1^{C_1}, \mathbf{X}_1^{C_1}, \dots, \mathbf{X}_1^{C_N}, \dots, \mathbf{X}_T^O, \mathbf{X}_T^{C_1}, \mathbf{X}_T^{C_1}, \dots, \mathbf{X}_T^{C_N}, I_1, \dots, I_T) = & \quad (4.10) \\
 \frac{1}{Z} \underbrace{\prod_{t=2}^T \psi(\mathbf{X}_t^O, \mathbf{X}_{t-1}^O)}_{\text{Temporal Prior}} \underbrace{\prod_{ti} \psi_i(\mathbf{X}_t^O, \mathbf{X}_t^{C_i}) \prod_{tij} \psi_{ij}(\mathbf{X}_t^{C_i}, \mathbf{X}_t^{C_j})}_{\text{Spatial Prior}} \underbrace{\prod_t \phi(\mathbf{X}_t^O, I_t) \prod_{ti} \phi_i(\mathbf{X}_t^{C_i}, I_t)}_{\text{Image Likelihood}}
 \end{aligned}$$

where \mathbf{X}_t^O and $\mathbf{X}_t^{C_i}$ is the state of the object, O , and object's n -th component, C_i , at time t respectively ($i \in$

$[1, \dots, N]$ and $t \in [1, \dots, T]$; $\psi(\mathbf{X}_t^O, \mathbf{X}_{t-1}^O)$ is the temporal compatibility of object state between frames t and $t - 1$; $\psi_i(\mathbf{X}_t^O, \mathbf{X}_t^{C_i})$ is the spatial compatibility of the object and its components at frame t ; $\psi_{ij}(\mathbf{X}_t^{C_i}, \mathbf{X}_t^{C_j})$ is the spatial compatibility between object components at frame t ; and $\phi(\mathbf{X}_t^O, I_t)$ and $\phi_i(\mathbf{X}_t^{C_i}, I_t)$ denote the local likelihoods for the object and component states respectively, where I_t corresponds to the image at time t . Notice that we assume stationary likelihoods and priors, *i.e.* likelihood functions and priors do not change over time.

Notice that since our component and the object image regions are not independent the above formulation is only an approximation. However, since the overlap tends to be small and the features that operate in the different regions tend to be different (selected independently using AdaBoost procedure), this approximation works well in practice.

Our framework can be viewed as having five distinct components: **(i)** a graphical model, **(ii)** an inference algorithm that provides the ability to infer a state of each node in the graph, **(iii)** a local evidence distribution (or image likelihood), **(iv)** a proposal process for some or all nodes in a graphical model, and **(v)** a set of spatial and/or temporal constraints corresponding to the edges in a graph. We will now discuss each one of these in turn.

4.2.1 Building the Graphical Model

In a single frame we represent objects using a two-layer spatial graphical model. The fine, component, layer contains a set of loosely connected “parts.” The coarse, object, layer corresponds to an entire appearance model of the object and is connected to all constituent components. Examples of such models for pedestrian and vehicle detection are shown in the shaded regions of Figure 4.3 **(a)** and **(b)** respectively. In both cases objects are modeled using four overlapping image components. For the vehicle, the components are: top-left (TL), top-right (TR), bottom-right (BR) and bottom-left (BL) corners; while for the pedestrian, they are: head (HD), left arm (LA), right arm (RA) and legs (LG). The corresponding image regions are illustrated in Figure 4.3 **(left)** in both cases.

To integrate temporal constraints we extend the spatial graphical models over time to an arbitrary length temporal window. The resulting spatio-temporal graphical models are shown in Figure 4.3 **(a)** and 4.3 **(b)**. Having a two-layer graphical model, unlike the single component layer model of [197], allows the inference process to reason explicitly about the object as a whole, as well as helps reduce the complexity of the graphical model, by allowing the assumption of the conditional independence of components over time given the overall object pose. Alternatively, one can also imagine building a single object layer model, which would be similar to the Boosted Particle Filter [157] (with bi-directional temporal constraints). Hence, the proposed model can be interpreted as an extension of [157].

Depending on an object one may or may not have a likelihood or a proposal process for the object layer nodes. For example, if the whole appearance is indeed too complicated to model as a whole (*e.g.* arbitrary size vehicles) and can only be modeled in terms of components, we can simply assume uniform likelihood over the entire object state space. In such cases the second, object, layer nodes simply fuse the component information to produce estimates for the object state that are consistent over time.

4.2.2 Learning Spatial and Temporal Constraints

Each undirected edge between components i and j has an associated potential function $\psi_{ij}(\mathbf{X}_t^{C_i}, \mathbf{X}_t^{C_j}) = \psi_{ji}(\mathbf{X}_t^{C_j}, \mathbf{X}_t^{C_i})$ that encodes the compatibility between pairs of node states. Similar potentials are defined between the components and the object, $\psi_i(\mathbf{X}_t^O, \mathbf{X}_t^{C_i})$, and across time, $\psi(\mathbf{X}_t^O, \mathbf{X}_{t-1}^O)$. Since in our framework the state space for the object and components is one and the same, we also make no distinction between the different potential functions. In this section we formulate potentials for the components, but same equations apply to $\psi_i(\mathbf{X}_t^O, \mathbf{X}_t^{C_i})$ and $\psi(\mathbf{X}_t^O, \mathbf{X}_{t-1}^O)$.

The potential $\psi_{ij}(\mathbf{X}_t^{C_i}, \mathbf{X}_t^{C_j})$ is modeled using a robust mixture of M_{ij} Gaussians, which gives a convenient form for the conditional distributions,

$$\psi_{ij}(\mathbf{X}_t^{C_i}, \mathbf{X}_t^{C_j}) = \lambda^0 \mathcal{N}(\mathbf{X}_t^{C_j}; \mu_{ij}, \Lambda_{ij}) + (1 - \lambda^0) \sum_{m=1}^{M_{ij}} \delta_{ijm} \mathcal{N}(\mathbf{X}_t^{C_j}; F_{ijm}(\mathbf{X}_t^{C_i}), G_{ijm}(\mathbf{X}_t^{C_i}))$$

where λ^0 is a fixed outlier probability, μ_{ij} and Λ_{ij} are the mean and covariance of the Gaussian outlier process, and $F_{ijm}(\cdot)$ and $G_{ijm}(\cdot)$ are functions that return the mean and covariance matrix respectively of the m -th Gaussian mixture component; δ_{ijm} is the relative weight of an individual component and $\sum_{m=1}^{M_{ij}} \delta_{ijm} = 1$. For experiments in this chapter we used $M_{ij} = 2$ mixture components.

Given a set of labeled images, where each component is associated with a single reference point, we use standard iterative Expectation-Maximization (EM) algorithm (see details in Section 3.4.2) with K-means initialization to learn $F_{ijm}(\cdot)$ and $G_{ijm}(\cdot)$ directly (a discussion on learning conditionals directly versus deriving them analytically from joint distribution encoded by the potential function can be found in Section 5.3.1) of the form:

$$F_{ijm}(\mathbf{X}_i) = \mathbf{X}_i + \begin{bmatrix} \frac{\mu_{ijm}^x}{\mu_{ijm}^s}, \frac{\mu_{ijm}^y}{\mu_{ijm}^s}, \mu_{ijm}^s \end{bmatrix}^T \quad (4.11)$$

$$G_{ijm}(\mathbf{X}_i) = \begin{bmatrix} \sigma_{x,ijm}^2 & 0 & 0 \\ 0 & \sigma_{y,ijm}^2 & 0 \\ 0 & 0 & \sigma_{s,ijm}^2 \end{bmatrix}^T \quad (4.12)$$

where $\mu_{ijm}^x, \mu_{ijm}^y, \mu_{ijm}^s$ is the mean position and scale of component or object j relative to i . $G_{ijm}(\cdot)$ is assumed to be diagonal matrix, representing the variance in relative position and scale. Examples of the learned conditional distributions can be seen in Figure 4.4 (a), (b), and (c).

4.2.3 AdaBoost Image Likelihoods

The likelihood, $\phi_i(\mathbf{X}_t^{C_i}, I_t)$ models the probability of observing the image region I_t conditioned on the state $\mathbf{X}_t^{C_i}$ of the component i , and ideally should be robust to partial occlusions and the variability of image statistics across many different inputs. To that end we build our likelihood model using a boosted classifier. As with potentials, we make no explicit distinction between component, $\phi_i(\mathbf{X}_t^{C_i}, I_t)$, and object, $\phi(\mathbf{X}_t^O, I_t)$, likelihoods.

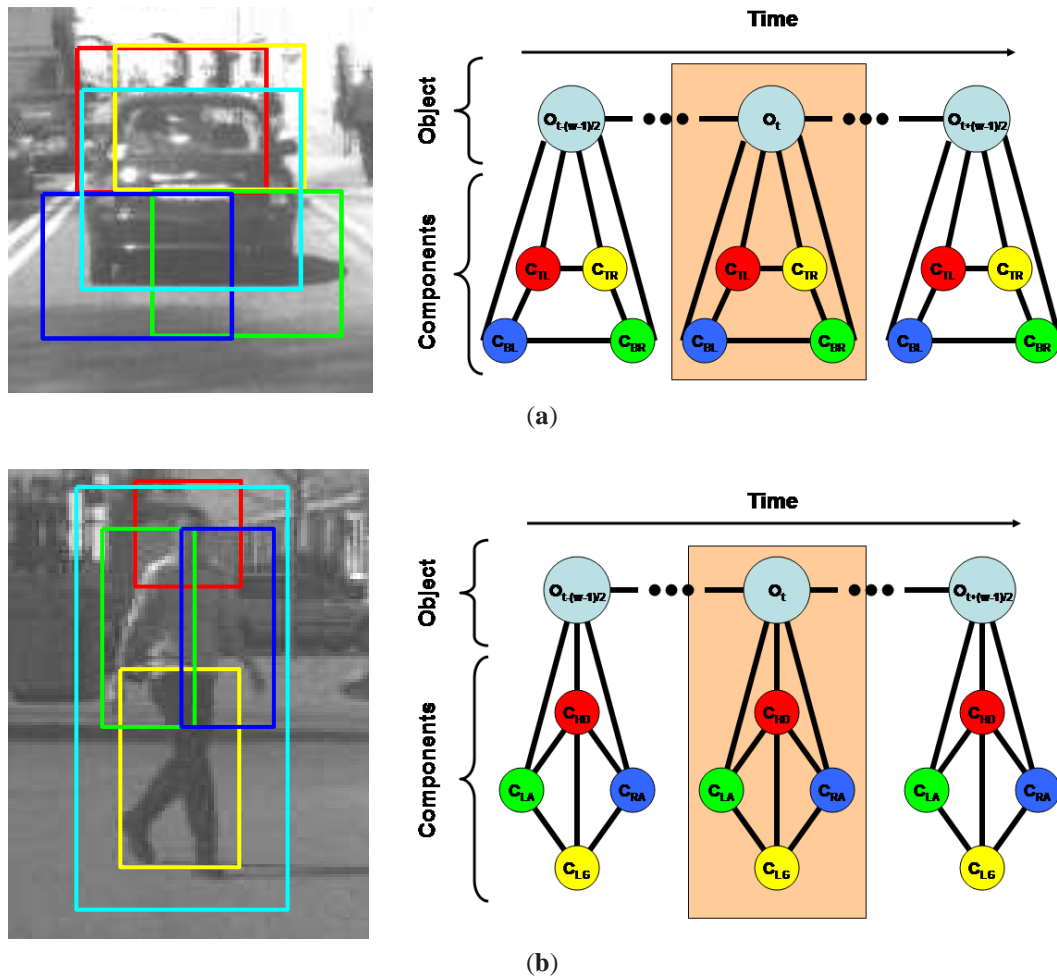


Figure 4.3: **Graphical models for the pedestrian and vehicle detection and tracking.** Graphical models for the vehicle and pedestrian objects are illustrated in (a) and (b) respectively. The graphical structure for the model is shown on the right; corresponding color coded image components are illustrated on the left. Entire appearance model in both cases is in cyan, the components are in red, yellow, blue and green. The shaded region of the model on the right corresponds to a single frame model that can be used for object detection in an image. Spatio-temporal models are obtained by replicating this spatial model along the temporal domain to a w -length window and then connecting the object layer nodes across time. The resulting spatio-temporal models are able to both detect and tract the corresponding objects in video.

Following the framework described in Section 4.1 we train boosted detectors, $h^{C_i}(I)$, for each component C_i , $i \in [1, \dots, N]$, and the corresponding overall object appearance, $h^O(I)$, where appropriate. For simplicity we use AdaBoost [236] without a cascade (training with a cascade would likely improve the computational efficiency of the system). In order to reduce the number of false positives produced by the detectors, we use a bootstrap procedure that iteratively adds false positives that are collected by running the trained strong classifier over the set of background images (not containing the desired object) and then re-training the detectors using the old positive and the new extended negative sets. Examples of the positive and negative samples and the resulting features selected by the AdaBoost learning for components of pedestrian graphical object model

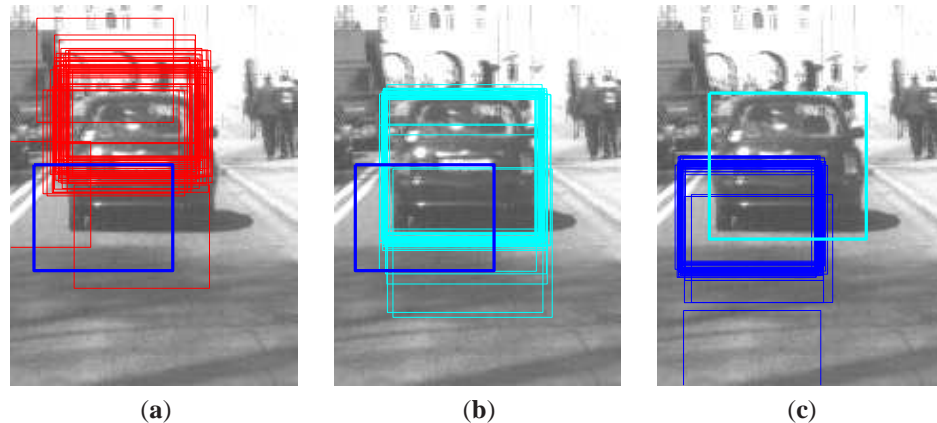


Figure 4.4: **Modeling spatial constraints.** Illustrated are learned conditional distributions from (a) Bottom-Left (BL) to Top-Left (TL) component, (b) Bottom-Left (BL) to the whole appearance model, and (c) whole appearance model to the Bottom-Left (BL) component of the vehicle. In each case the conditional distribution is visualized by conditioning on one of the parts and sampling location and scale for the other part based on the corresponding learned distribution.

can be seen in Figures 4.5–4.8.

The output of the AdaBoost classifier for each component C_i is a confidence $h^{C_i}(I_{t, \mathbf{X}_t^{C_i}})$ that the given image pattern $I_{t, \mathbf{X}_t^{C_i}}$ (obtained by selecting portion of the image I_t according to the hypothesized state $\mathbf{X}_t^{C_i}$) is of the desired class (see Section 4.1). We convert this confidence into a likelihood function by first normalizing the α_k 's, so that $h^{C_i}(I_{t, \mathbf{X}_t^{C_i}}) \in [0, 1]$, and then exponentiating

$$\phi_i(\mathbf{X}_t^{C_i}, I_t) = \phi_i(I_t | \mathbf{X}_t^{C_i}) \propto \exp \left[\frac{h^{C_i}(I_{t, \mathbf{X}_t^{C_i}})}{\mathcal{T}} \right]. \quad (4.13)$$

Similarly, for the entire object likelihood, where appropriate,

$$\phi(\mathbf{X}_t^O, I_t) = \phi(I_t | \mathbf{X}_t^O) \propto \exp \left[\frac{h^O(I_{t, \mathbf{X}_t^O})}{\mathcal{T}} \right], \quad (4.14)$$

where \mathcal{T} is a temperature parameter that controls the smoothness of the likelihood function, with smaller values of \mathcal{T} leading to a more peaked distribution. Similar likelihoods can be derived for object nodes as well where appropriate. Consequently we can also anneal the likelihood by deriving a schedule with which \mathcal{T} changes. We found an exponential annealing schedule $\mathcal{T} = \mathcal{T}_0 v^\kappa$, where \mathcal{T}_0 is the initial temperature, v is a fraction $\in (0, 1)$, and κ is the annealing iteration, to work well in practice. AdaBoost classifiers are learned using a database of 861 vehicles and 662 pedestrians [144]. The number of negative examples after bootstrapping tends to be on the order of 2000 to 3000.

Depending on an object one may or may not have a likelihood or a proposal process for the object layer nodes. For example, if the whole appearance of an object is indeed too complicated to model as a whole (*e.g.* arbitrary size vehicles) and can only be modeled in terms of components, we can simply assume a uniform likelihood over the entire state space. In such cases the object layer nodes simply fuse the component information to produce estimates for the object state that are consistent over time.

Note that while our state space is continuous, the AdaBoost likelihood can only be evaluated at discrete

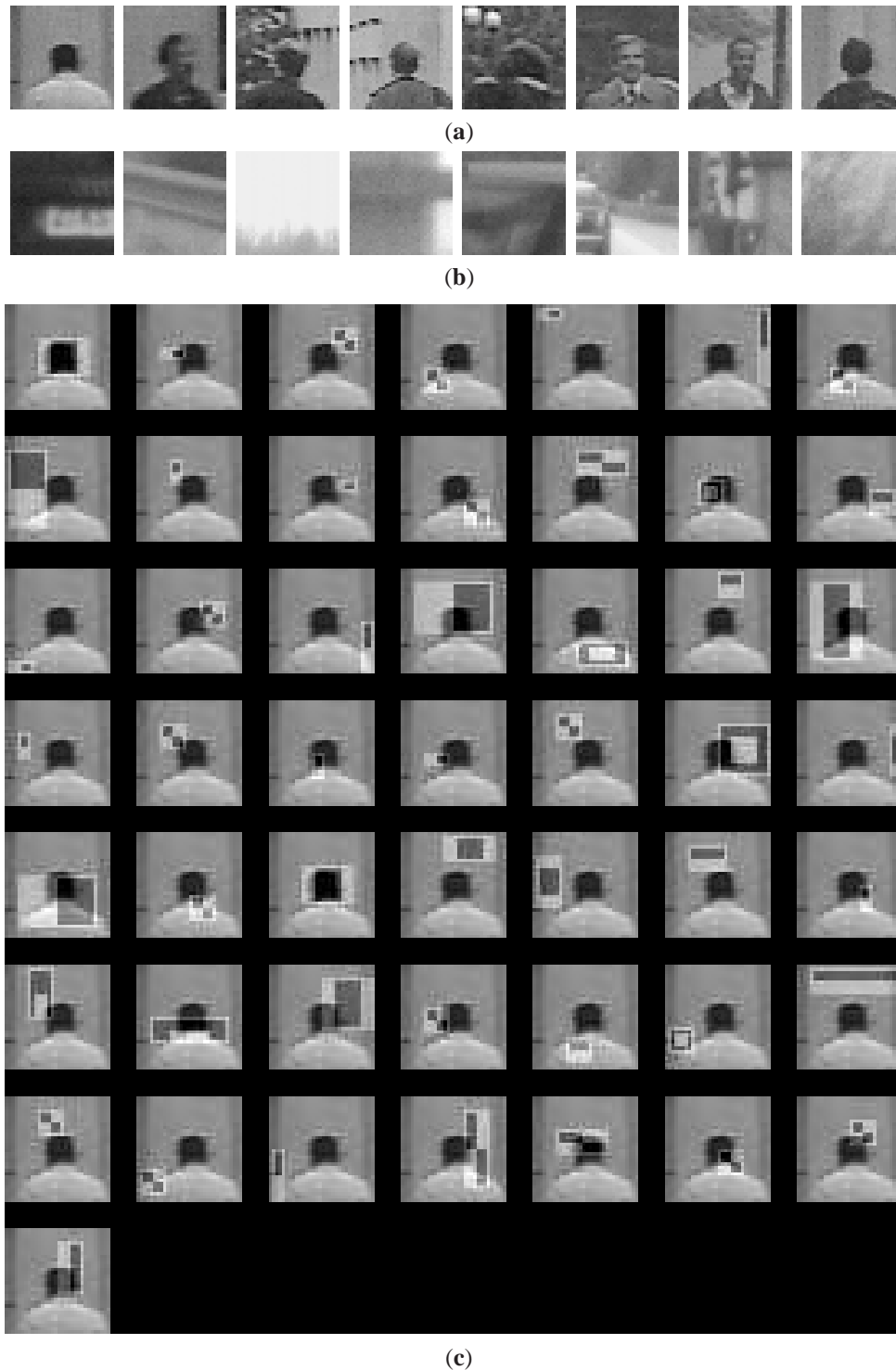


Figure 4.5: **AdaBoost detector for the head.** Typical positive and negative examples are shown in (a) and (b) respectively. All positive examples were selected in a supervised fashion and scaled to a canonical size of 32×32 pixels. Negative examples were picked at random from a set of street images that did not contain people. The negative set was then refined using bootstrap procedure. The features selected by AdaBoost, overlaid on an example image of the head, are shown in (c). The final strong classifier learned consisted of 50 features shown in (c) weighted by corresponding α_k 's (not illustrated).



Figure 4.6: **AdaBoost detector for the left side of an upper body.** Typical positive and negative examples are shown in (a) and (b) respectively. All positive examples were selected in a supervised fashion and scaled to a canonical size of 28×60 pixels. Negative examples were picked at random from a set of street images that did not contain people. The negative set was then refined using bootstrap procedure. The features selected by AdaBoost, overlaid on an example image of the upper body, are shown in (c). The final strong classifier learned consisted of 50 features shown in (c) weighted by corresponding α_k 's (not illustrated).

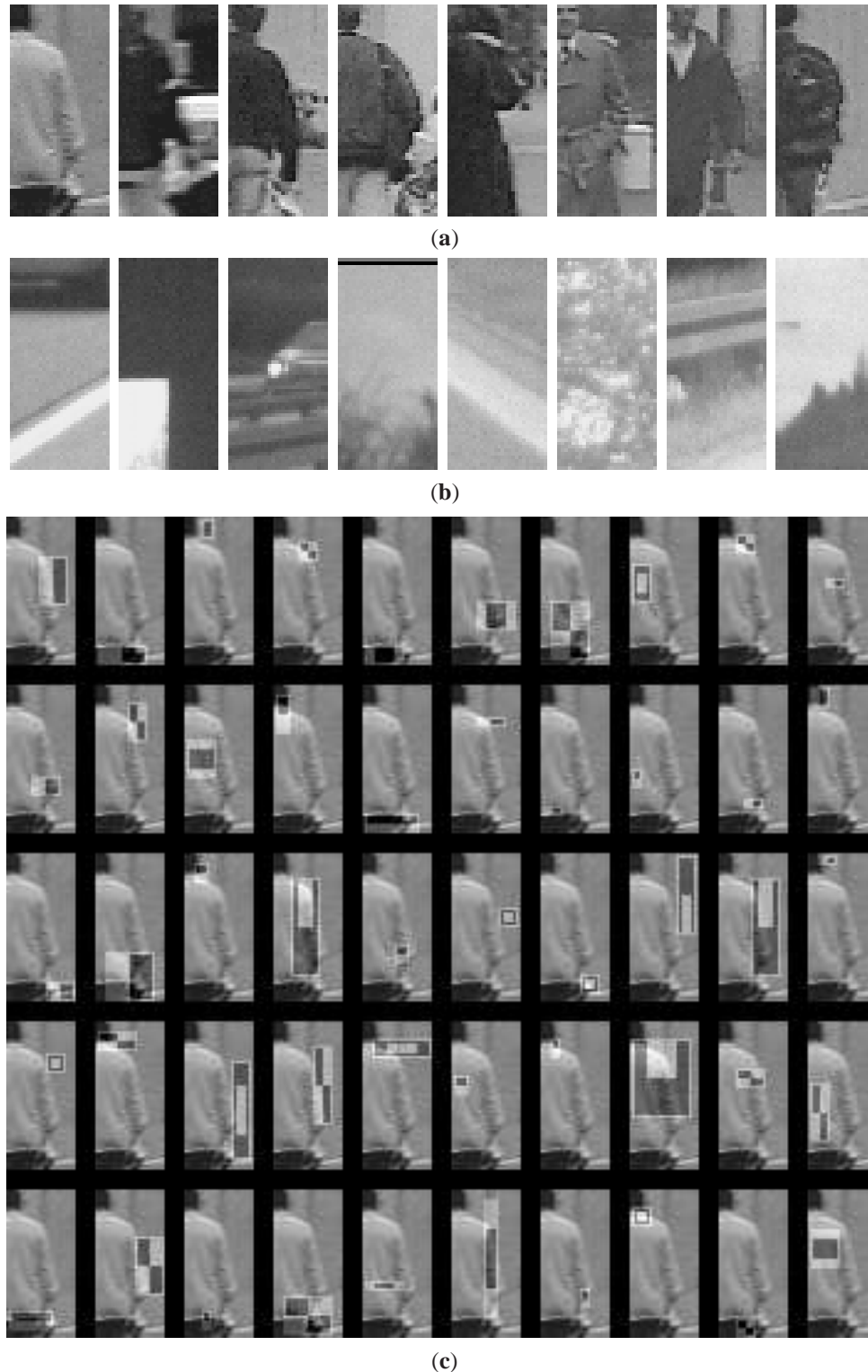


Figure 4.7: **AdaBoost detector for the right side of an upper body.** Typical positive and negative examples are shown in (a) and (b) respectively. All positive examples were selected in a supervised fashion and scaled to a canonical size of 28×60 pixels. Negative examples were picked at random from a set of street images that did not contain people. The negative set was then refined using bootstrap procedure. The features selected by AdaBoost, overlaid on an example image of the upper body, are shown in (c). The final strong classifier learned consisted of 50 features shown in (c) weighted by corresponding α_k 's (not illustrated). Notice that the first feature selected is symmetric to the one chosen by the left side detector in Figure 4.6.

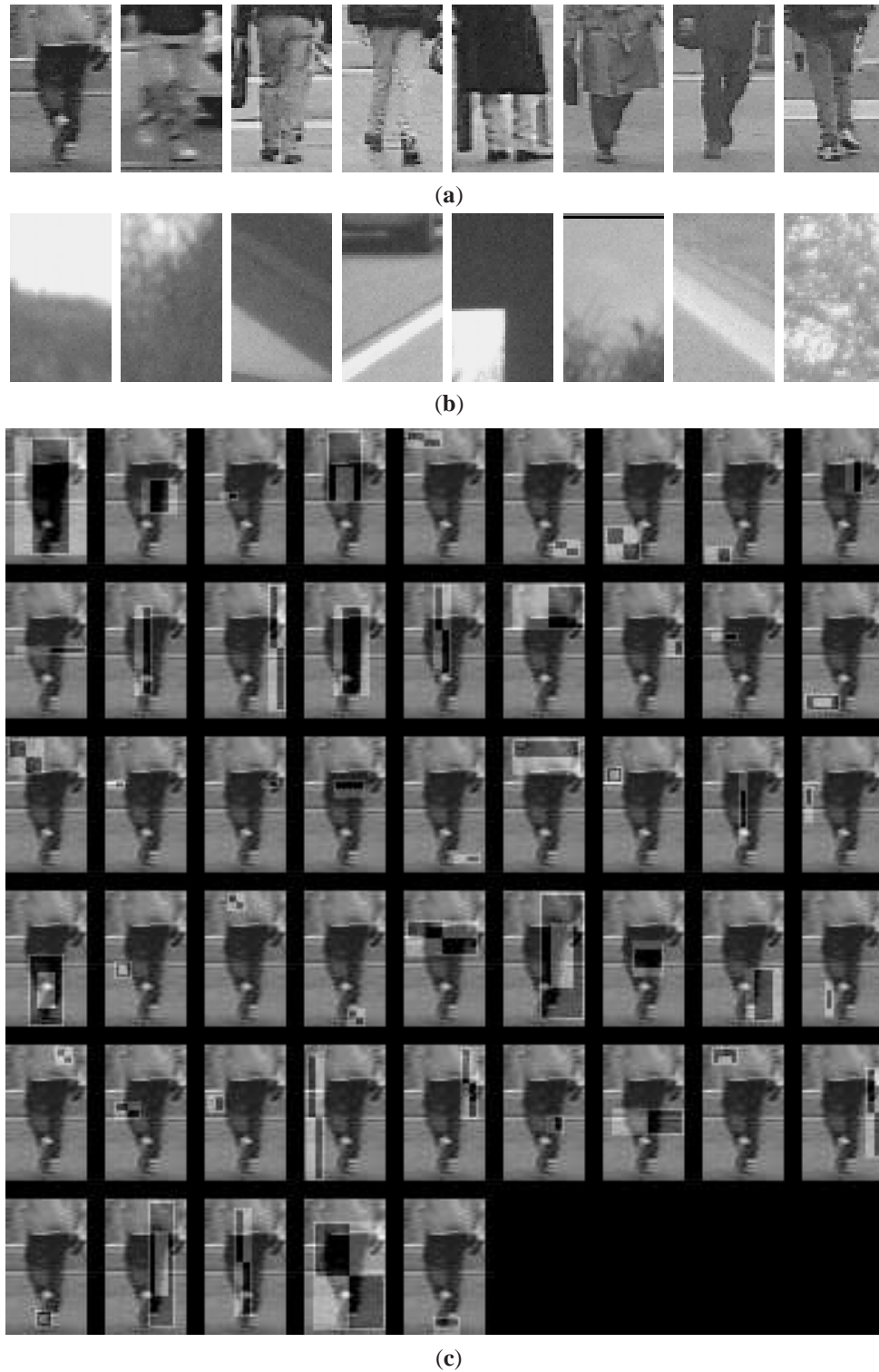


Figure 4.8: **AdaBoost detector for the lower body.** Typical positive and negative examples are shown in (a) and (b) respectively. All positive examples were selected in a supervised fashion and scaled to a canonical size of 36×60 pixels. Negative examples were picked at random from a set of street images that did not contain people. The negative set was then refined using bootstrap procedure. The features selected by AdaBoost, overlaid on an example image of the lower body, are shown in (c). The final strong classifier learned consisted of 50 features shown in (c) weighted by corresponding α_k 's (not illustrated).

pixel positions. To enable sub-pixel accuracy one can use interpolation. In addition, it may be prohibitively expensive to compute the likelihood due to need for scaling imposed by the continuous scale space. In practice we discretized the scale space, and use bi-cubic interpolation to approximate the continuous likelihood function. This is done by precomputing the likelihood for each pixel at a set of scales and interpolating, using bi-cubic interpolation, a likelihood for any continuous state \mathbf{X}_i .

4.2.4 Inference using Belief Propagation

Inferring the state of the object and its components in our framework is defined as estimating belief in a graphical model. We use Particle Message Passing described in detail in Section 3.7 to deal with this task. The approach is a generalization of particle filtering [54] which allows inference over arbitrary graphs rather than a simple chain. In this generalization the “message” used in standard belief propagation is approximated with a kernel density (formed by propagating a particle set through a mixture of Gaussians density), and the conditional distribution used in standard particle filtering is replaced by product of incoming messages. Most of the computational complexity lies in sampling from a product of kernel densities required for message passing and belief estimation; we use efficient sequential multiscale Gibbs sampling and epsilon-exact sampling [95] to address this problem.

Individual messages may not constrain a node well, however, the product over all incoming messages into the node tends to produce a very tight distribution in the state space. For example, any given component of a vehicle is incapable of estimating the height of the vehicle reliably, however, once we integrate information from all components in the object layer node, we can get a very reliable estimate for the overall object size.

More formally a message m_{ij} is written as

$$m_{ij}(\mathbf{X}_j) = \int \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A_i \setminus j} m_{ki}(\mathbf{X}_i) d\mathbf{X}_i, \quad (4.15)$$

where A_i is the set of neighbors of node i and $\phi_i(\mathbf{X}_i) \equiv \phi_i(\mathbf{X}_i, I_i)$ is the local evidence (or likelihood) associated with the node i , and $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ is the potential designating the compatibility between the states of node i and j . The details of how the message updates can be carried out by stratified sampling from belief and proposal function see Section 3.7.

While it is possible and perhaps beneficial to perform inference over the spatio-temporal model defined for the entire image sequence, there are many applications for which this not an option due to the lengthy off-line processing required. Hence, we use a w -frame windowed smoothing algorithm where w is an odd integer ≥ 1 (see Figure 4.3). There are two strategies one can employ when performing windowed smoothing in the proposed framework: (1) object-detection centric strategy or a (2) tracking-centric strategy. In the former we re-initialize all nodes every time we shift a window, hence the inference is memoryless and temporal integration is only applied within the window of size w . In the latter we only initialize the nodes associated with a new frame; this tends to enforce temporal consistency from before $t - (w - 1)/2$. While the tracking-centric strategy tends to converge faster and produce more consistent results over time, it is also less sensitive to objects entering and leaving the scene. Note that with $w = 1$, the algorithm resembles single frame component-based fusion [249].

The BP message update equation can be executed either by updating all the messages m_{ij} in a batch or by updating certain messages before others. The latter, also called focused message updating [44], can

significantly speed up BP given some *a priori* knowledge of the object model. For example, if an object layer node of the graph has no associated likelihood or proposal process (as is in our car model), then it would make sense to update the incoming messages to it before updating the outgoing ones. For generality and ease, we choose not to do focused message updating here.

4.2.5 Proposal Process

To reliably detect and track the object, non-parametric BP makes use of a bottom-up proposal process, that constantly looks for and suggests alternative hypothesis for the state of the object and components. We model the proposal distribution using a weighted particle set. To form a proposal particle set for a component, we run the corresponding AdaBoost detector over an image at a number of scales to produce a set of detection results that score above the $\frac{1}{2} \sum_{k=1}^K \alpha_k$ threshold. The weight for these particles is set to be proportional to the likelihood. While this set tends to be manageable for the entire appearance model, it is usually large for non-specific component detectors (a few thousand locations can easily be found in any given image). To reduce the dimensionality we only keep the top P scoring detections, where P is on the order of a 100 to 200. To achieve breadth of search we importance sample particles from the proposal using a uniform distribution.

4.3 Experiments

We tested the proposed approach on a set of images collected with a single car-mounted grayscale camera. As mentioned in Section 4.2.3 we trained the AdaBoost likelihoods off-line using a database of 861 vehicles collected by ourselves and 662 pedestrians taken from [144]. In both cases, testing images were collected using a different camera, under different imaging conditions and in different location from those in the training set. Hence, none of the test images appeared in the training set. To normalize the size and location of components for training we manually labeled a set of (4–5) landmark points corresponding to a target object and components in each training image.

4.3.1 Multi-frame Single Target Detection and Tracking

The result of vehicle detection and tracking over a sequence of 55 consecutive frames can be seen in Figure 4.9. A 3-frame spatio-temporal object model was used and was shifted, using tracking-centric strategy, discussed in Section 4.2.4, over time. The position and scale of components and the object as a whole are shown in the respective colors. In Figure 4.9 (a) a set of considered proposals for the components and object as a whole are shown. Neither of these proposal processes are able to reliably locate the object. By combining the proposals from various components using our inference framework, we are able to reliably and consistently detect and localize the object, and its parts. We ran BP with 30 particles for 10 iterations at every frame. For comparison, we implemented a simple fusion scheme that blindly averages the best detection result from each of the four components (Figure 4.9 (b) ‘Best Avg.’) to produce an estimate for the vehicle position and scale independently at every frame. The performance of the simple fusion detection is very poor suggesting that the noisy component detectors often do not have the global maximum at the correct position and scale. In contrast, the spatio-temporal object model consistently combines the evidence for accurate estimates throughout the sequence.

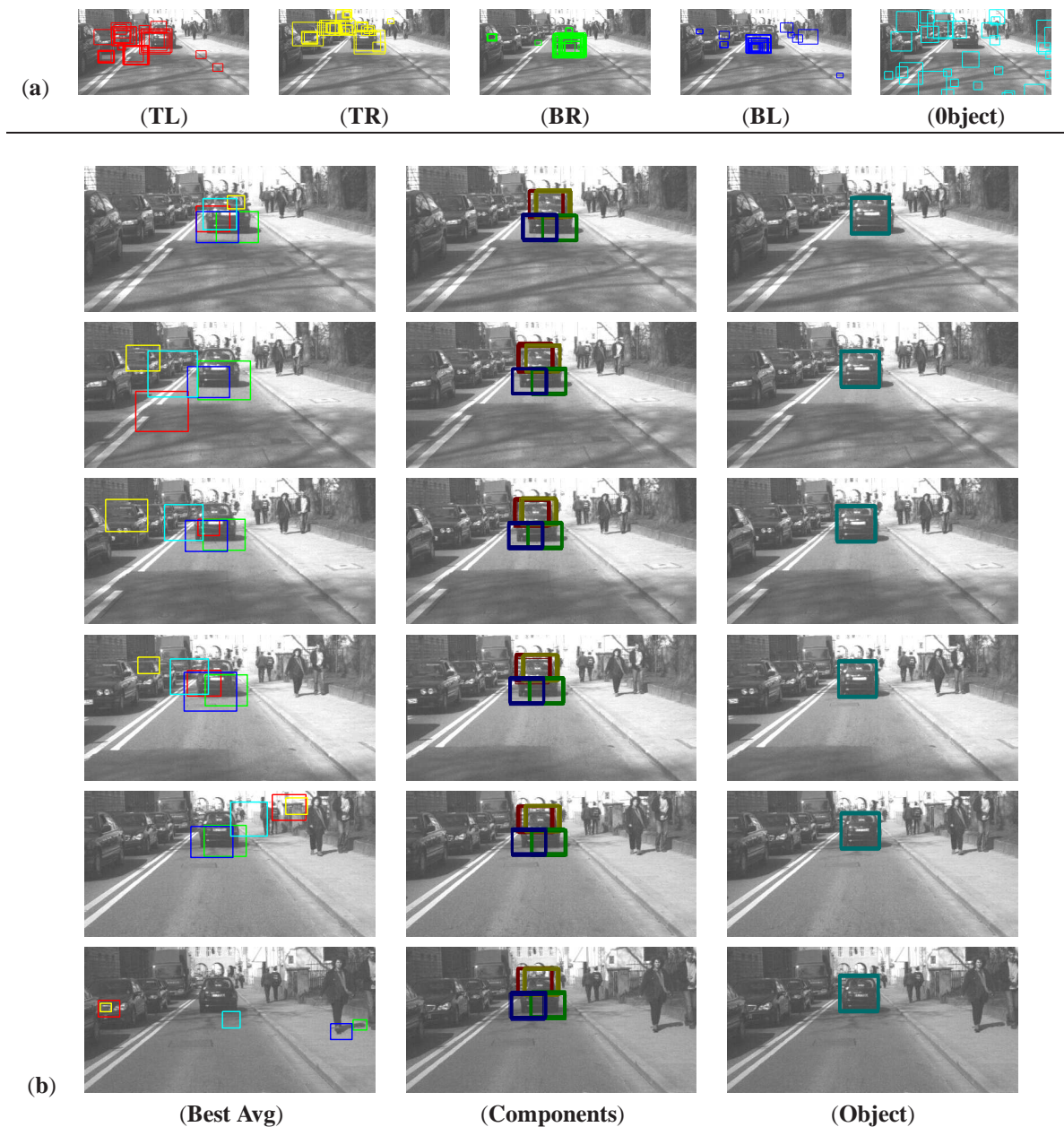


Figure 4.9: **Vehicle component-based spatio-temporal object detection and tracking.** (a) shows the initialization/proposal distribution, and (b) 30 samples taken from the belief for each of the four components (middle) and an object (right). The detection and tracking was conducted using a 3-frame smoothing window. Frames 2 through 52 are shown (top to bottom respectively) at 10 frame intervals. For comparison (b) (left) shows the performance of a very simple fusion algorithm, that fuses the best result from each of the components by blind averaging.

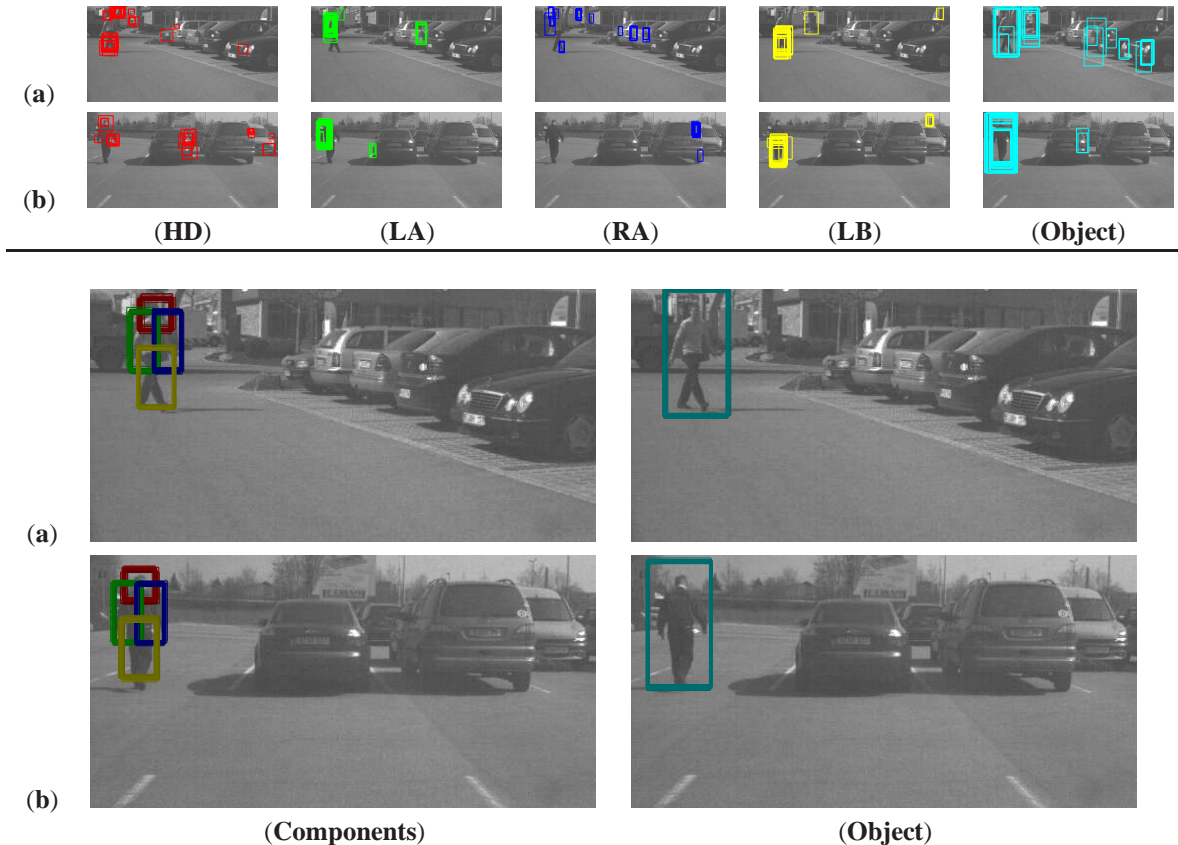


Figure 4.10: **Pedestrian component-based spatio-temporal object detection** for two subjects (a) and (b); (top) shows the initialization/proposal distribution, and (bottom) 30 samples taken from the belief for each of the four components and the object. The detection was conducted using a 3-frame temporal smoothing window.

The performance of the pedestrian spatio-temporal detector is shown in Figure 4.10. A 3-frame spatio-temporal object model is run at a single instance in time for two pedestrians in two different scenes. Similar to the vehicle detection we run BP with 30 particles for 10 iterations. For both experiments the initial temperature of the likelihood was fixed at $\mathcal{T}_0 = 0.2$.

4.3.2 Single Frame Multi-target Detection

While in general the algorithm presented here is capable of detecting multiple targets, by converging to multimodal posterior distributions for components and objects, in practice this tends to be very difficult. It is well known that particle filters have problems tracking multimodal distributions [157]. This framework that further extends particle filtering, and requires message update routine to take products over particle sets, makes this problem even more apparent. We postulate that given a large number of particles this can be done, but would be prohibitively expensive. Instead we use a peak suppression scheme, where we look for the modes of the posterior one at a time, and suppress the response of our likelihood function in the regions where peaks have already been found. An example of this that is produced by running a purely spatial graphical model over the image is shown in Figure 4.11.

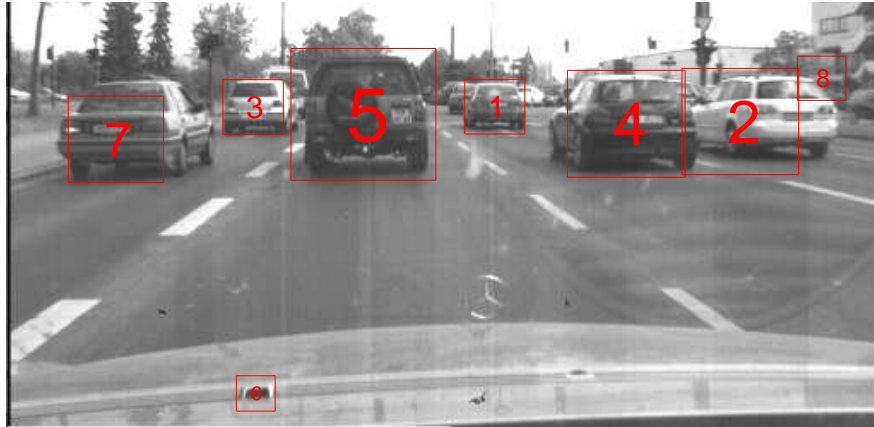


Figure 4.11: **Multiple target detection.** Example of detecting multiple instances of a vehicle object in the same image is shown. The greedy approach employed detects individual instances of vehicle class (8 in total) by searching for each instance in succession. Once detected, the most prominent mode of the posterior (red) is labeled as a detection, and the associated image evidence are suppressed from consideration in the future runs. The modes that correspond to instances with highest confidence are found in early stages of this greedy search strategy as is designated by the labels. For further discussion please see text.

In Figure 4.11 vehicle detection was administered 8 times in succession. After each run the most prominent mode, shown in red, of the resulting object posterior distribution was labeled as an instance of the vehicle object and image evidence in the corresponding region were suppressed for subsequent detections. This scheme tends to pull out instances of the object class that have high confidence first, followed by instances where confidence is lower. This can be seen from the labels assigned to the object instances in Figure 4.11. For the example shown, we manually choose the number of objects expected (8), however, this can be done automatically as well by looking at the overall likelihood for the given object instance. Notice, that we are able to quite reliably pull out all 6 real instances of the object at roughly correct position and scale; we also pull out two false positives. The false positive labeled ‘6’, which corresponds to the blemish on the windshield of the car recording the scene, indeed looks very similar to the back of the car profile at a much smaller scale. In both cases, the false positives had a much lower confidence than real instances as is illustrated by the labels given by our greedy search algorithm. Lastly, it is also worth noting that we observed that our approach that explicitly encodes the spatial relationships between components is better capable of handling variations in orientation of the object (see various instances of detected cars in Figure 4.11).

4.4 Conclusion and Discussion

In this chapter we show how the mathematical tools presented in the previous chapter can be leveraged to build a class of models for generic object detection and localization. Experiments presented in this chapter are a proof of concept that continuous-state graphical models provide effective means of modeling and drawing inferences about objects in a visual detection task. Presented architecture can be interpreted as an extension of the constellation model [61], where the spatial constraints are non-parametric rather than Gaussian. However, we believe that the true power of the architecture presented here is that it can be extended to deal with complex articulated objects as will be shown in the next chapter.

We present a novel object detection and tracking framework exploiting boosted classifiers and non-parametric belief propagation (NBP). The approach provides component-based detection and integrates temporal information over an arbitrary size temporal window. We illustrate the performance of the framework with two classes of objects: vehicles and pedestrians. In both cases we can reliably infer position and scale of the objects and their components.

While the proposed approach is effective in detecting and tracking generic objects in images and video, it has two shortcomings that need to be addressed in the future: **(1)** it is fully supervised, and **(2)** it cannot effectively deal with multi-target detection and interactions. The fully supervised nature of the algorithm is the Achilles' heel of the proposed approach. Clearly means of automatically learning the structure and parameters of graphical object models from weakly labeled or un-labeled data is necessary to make it widely applicable to generic object detection and localization. Doing this manually, as is done in this chapter, will not scale to large sets of complex models for which expert knowledge may not be available. The greedy approach developed here for multi-target detection, while effective, requires multiple runs of the basic algorithm, resulting in complexity that is linear in the number of object instances. Ideally, a better solution that can jointly estimate all instances and is sub-linear in the number of such instances can be developed. One way of addressing this would be by explicitly maintaining multi-modal predictions for the posterior/beliefs and messages in the NBP framework, either using mixture tracking [233] and/or hybrid MCMC methods.

CHAPTER 5

Loose-limbed Body Model

In this chapter we present a fully automatic method for estimating the pose and tracking the human body in 3D. We introduce a novel representation for modeling the body that we call *loose-limbed body model*. This new model, in which limbs are connected via learned probabilistic constraints, facilitates initialization and failure recovery. The tracking and pose estimation problem is formulated as one of inference in a graphical model and belief propagation is used to estimate the pose of the body at each image frame. Each node in the graphical model represents the 3D position and orientation of a limb (Figure 5.1). Undirected edges between nodes represent statistical dependencies and these constraints between limbs are used to form messages that are sent to neighboring nodes in space and time. Additionally, each node has an associated likelihood defined over a set of image features. The combination of highly non-Gaussian likelihoods and a six-dimensional continuous parameter space (3D position and orientation) for each limb makes standard belief propagation algorithms infeasible. Consequently we exploit a form of non-parametric belief propagation [99, 220] that uses a variation of particle filtering and can be applied over a loopy graph, initially described in Section 3.7 and used for generic object detection and tracking in the previous chapter.

There are a number of significant advantages to this approach as compared to traditional methods for tracking human motion. Most current techniques model the body as a kinematic tree in 2D [111], 2.5D [34], or 3D [30, 52, 193, 210] leading to a high-dimensional parameter space (25–50 dimensions is not uncommon). Searching such a high-dimensional space directly is impractical and so current methods typically rely on manual initialization of the body model. Additionally, they often exploit strong priors characterizing the types of motions present. When such algorithms lose track (as they eventually do), the dimensionality of the state space makes it difficult to recover.

While the full body pose is hard to recover directly, the location and pose of individual limbs is much easier to compute. Many good face/head detectors exist [20, 115, 236] and limb detectors have been used for some time (e.g. [20, 147, 173, 187]). The approach we take here can use bottom up information from feature detectors of any kind and consequently should generalize to a rich variety of input images. In our implementation we exploit background/foreground separation and color coherency for computational simplicity but part detectors that perform well against arbitrary backgrounds are becoming standard [173, 236].

With a kinematic tree model, exploiting this partial, “bottom-up” information is challenging. If one could definitively detect the body parts, then inverse kinematics could be used [256] to solve for the body pose, but in practice low-level part detectors are noisy and unreliable. The use of a loose-limbed model and belief

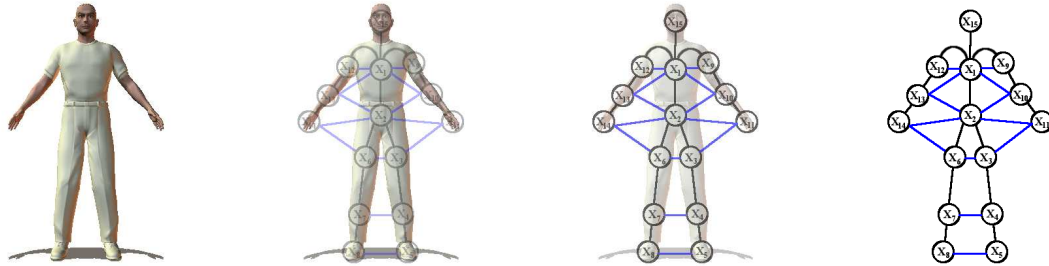


Figure 5.1: **Graphical model for a person.** Nodes represent limbs and arrows represent statistical dependencies between limbs. Black edges correspond to the kinematic constraints, and blue to the interpenetration constraints.

propagation provides a principled framework for incorporating information from part detectors. Because the inference algorithm operates over a general graph rather than a forward chain as in traditional particle filter trackers, it is also straightforward to perform temporal forward–backward smoothing of the limb trajectories without modifying the basic approach.

A loose-limbed body model requires a specification of the probabilistic relationships between joints at a given time instant and over time. We represent these non-Gaussian relationships using mixture models that are learned from a database of motion capture sequences. It is worth noting that these models encode information about joint limits and represent a relatively weak prior over human poses, which is appropriate for tracking varied human motions.

The model also requires an image likelihood measure for each limb. We formulate our likelihood model based on foreground silhouette and edge features. The likelihoods for different features are defined separately and combined using independence assumptions across views and feature types. It should be noted, however, that our framework is general and can use any and all available features.

We test the method by tracking subjects viewed from a number (4 to 7) calibrated cameras in an indoor environment with no special clothing. There is nothing restricting this approach to multiple cameras and Chapter 6 will explore its use for monocular pose-estimation and tracking. Quantitative evaluation is performed using the HumanEva [194] dataset that contains synchronized motion capture data and multi-view video. The motion capture data obtained using a commercial Vicon (Vicon Motion Systems Inc., Lake Forest, CA) motion capture system serves as a “ground truth” in the quantitative comparison.

5.1 Previous Work

There has been significant work in recovering the full body pose from images and video in the last 10-15 years. The literature on the human pose estimation and tracking has been reviewed in detail in Chapter 2. Here, for completeness, we will briefly review only the most relevant literature to motivate our model.

As was discussed in Section 2.7, *discriminative approaches* attempt to learn direct mapping from image features to 3D pose from either a single image [1, 179, 181, 189, 206] or multiple approximately calibrated views [77]. These approaches tend to use silhouettes [1, 77, 179, 181] and sometimes edges [205, 206] as image features and learn probabilistic mapping in the form of Nearest Neighbor (NN) search [189], regression [1], mixture of Bayesian experts [206], or specialized mappings [179]. While such approaches are fast

	Centralized Models	Disaggregated Models	
	Kinematic-tree	Pictorial Structures	Loose-Limbed Body Model
Inference	Local Stochastic Search	Belief Propagation	Particle Massage Passing
State-space	Continuous	Discrete	Continuous
Constraints	Kinematic Penetration Occlusion Temporal	(<i>simple</i>) Kinematic	Kinematic Penetration Occlusion ¹ Temporal
Applications	Tracking	Pose Estimation	Pose Estimation/Tracking
Model	3D/2D	2D	3D/2D ¹
Complexity	Exponential	Linear	Linear

Table 5.1: **Comparison of loose-limbed body model to other generative approaches.** The approach presented in this chapter is illustrated in the grayed column on the right. The loose-limbed body model approach advocated in this thesis allows for continuous pose estimation and tracking with rich set of constraints, while having a tractable inference complexity that is linear in the number of body-parts in the model.

and have been shown to work reliably in restricted domains, overall they tend to deal poorly with missing or corrupt image data. Consequently, due to their discriminative nature, they tend to generalize poorly to recovering poses that are uncommon or unaccounted for during training. Furthermore, it is hard to embed prior knowledge into such approaches that ensures that the resulting articulations are plausible (*e.g.* parts of the body are not penetrating, or joint limits are preserved).

Generative approaches are much better equipped to deal with these issues, since they attempt to model the image generation process². For the time being we will concentrate on relevant generative approaches in this section; a broader discussion of related work can be found in Chapter 2. Generative approaches typically rely on a kinematic tree [139] representation of the body in 2D [111], 2.5D [34], or 3D [30, 52, 193, 210]. In such approaches the pose is defined by a set of parameters representing the global position and orientation of the root, usually a torso, and the joint angles representing the state of each limb with respect to the neighboring part higher up in the tree. Such centralized models are very expressive and are able effectively encode prior knowledge that can both reduce the ambiguities in the observed pose and ensure that recovered pose is to some extent realistic. The inference in these models typically amounts to generating a number of hypothesis for the pose, and evaluating the likelihood that a given hypothesis gives rise to the image evidence observed. Inference in such models, however, often requires stochastic search for the parameters in the high dimensional, 25-50 dimensional, state-space. Many specialized inference approaches have been developed to reduce the exponential complexity of the search in this high-dimensional space. Such inference methods typically take into account the structure [52, 136] of these models and/or dynamics [193] of human motion. However, none, can tractably infer the articulated pose without effective initialization that is relatively close to the solution. For this reason, these models are particularly valuable for tracking but have little consequence in the pose estimation task.

To address the complexity of inference in generative models a new class of disaggregated models has emerged. Disaggregated models for finding or tracking articulated objects date back to Fischler and Elschlager’s

¹This will be addressed in Chapter 6.

²While generative models employed for human pose and motion estimation are typically very weak (*i.e.* they cannot generate realistic images of articulated human motion) they still tend to be very effective for inference.

pictorial structures [62]. Various variations on this type of the model in the context of articulated and generic objects have been discussed in Sections 2.4.3 and 2.11.2 respectively. The main idea behind this class of models is that one can model a body as a collection of independent body parts that are constrained at the joints (ensuring proper articulated structure of the body). Based on this notion Ioffe and Forsyth [96, 97] first find body parts and then group them into figures in a bottom-up fashion. The approach exploits the fact that they have a discrete set of poses for parts that need to be assembled, but it prevents them from using rich likelihood information to “co-operate” with the body model when estimating the pose. Consequently this also prevents them from effectively dealing with partial occlusions of the body.

An alternative way of formulating probabilistic disaggregated models is via undirected graphical models described in Section 3.3. Assuming existence of conditional independencies between body parts (*e.g.* pose of right arm is conditionally independent of the left given the torso), one can model the body using a corresponding undirected graphical model and formulate tracking and pose estimation as inference in this graph. Felzenszwalb and Huttenlocher [59] introduced a clever inference scheme that allowed linear³ complexity exact inference in such graphical models using standard Belief Propagation. This method was then successfully illustrated on recovering mostly frontal 2D articulated poses. Their inference algorithm, however, requires a tree-structured topology for the graph, a particular form of potential functions (that encode connectivity at the joints), and discretization of the state-space (see full discussion of this in Section 3.5.2). As a result, efficiency comes at the cost of expressiveness and resulting models cannot account for occlusions, temporal constraints or long-range correlations between body parts, all of which will introduce loops into the graphical structure; expressive joint constraints are also disallowed. Furthermore, the inference algorithm relies on the fact that the 2D model has a relatively low-dimensional state-space for each body part, making it impractical to scale the approach to 3D inference. While later extended to deal, to some extent, with correlations between body parts in [122] and to jointly learn appearance in [173] the basic method still struggles with limitations discussed above.

The *loose-limbed body model* introduced in this chapter can be viewed as the “best of both worlds”, permitting expressiveness similar to that of kinematic tree models and allowing linear inference complexity similar to [59]. Our method makes no explicit assumptions about the topology of conditional independence properties of the graph (*i.e.* it can deal with cyclic graphs), allows for a richer class of potential functions, and can deal with continuous pose in 3D. To achieve tractable inference, however, we resort to approximate, instead of exact, inference using a variant of Non-parametric Belief Propagation, Particle Message Passing (PAMPAS). The comparison with closely related prior work discussed above is compactly summarized in Table 5.1.

A similar approach to ours was developed at roughly the same time for articulated hand tracking by Sudderth *et al.* [219]. However, in [219] authors only dealt with tracking and have not addressed the pose estimation problem. Another closely related approach was developed more recently by Rodgers *et al.* [177] for estimating articulated pose of people from range scan data. A similar in spirit approach to ours has also been adopted in [248] for tracking a 2D human motion using a dynamic Markov network and later in [93] using data-driven Belief Propagation. A much simplified observation model, that relied solely on silhouettes, was adopted in [248] and their system does not deal with pose estimation. In [93] a much richer observation model was used, but the approach is still limited to 2D pose inference in roughly frontal body orientations;

³Linear in the number of parts and exponential in the number of degrees of freedom for each part.

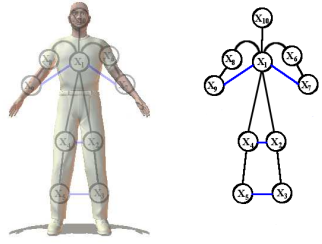
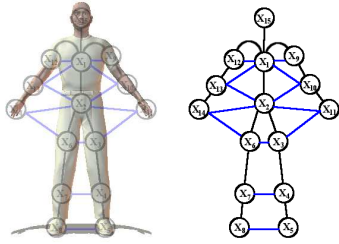
	10-part model	15-part model
Graph		
Number of nodes	10	15
Number of edges		
kinematic	9	14
interpenetration	4	13
Max node degree	7	8
Avg. node degree	2.6	3.6

Figure 5.2: 10-part and 15-part loose-limbed body models for a person. Graphical models corresponding to 10-part and 15-part model of a person are illustrated in (left) and (right) columns respectively. In both cases nodes represent limbs and edges represent statistical dependencies between limbs. Black edges correspond to the kinematic constraints, and blue to the interpenetration constraints. The degree of the node is defined as the number of edges incident on the that node. Node degree is one of the measures for corresponding graphical model complexity.

the subject is assumed to be facing towards the camera and wearing distinct clothes. All of these methods, while closely related, use somewhat different inference algorithms and a more direct comparison between them merits future research.

5.2 Loose-limbed Body Model

Following the framework that we first introduced in Chapter 4 the body is represented by a graphical model in which each graph node corresponds to a body part (upper leg, torso, *etc.*). We test our approach with two such models consisting of 10 and 15 body parts (see Figure 5.2), corresponding to a “coarse” and “fine” body representation respectively. The latter, in addition to modeling all major limbs of the body, also models hands and feet. The 15-part model also contains a more realistic parameterization of the torso that is modeled using 2 segments (pelvis and thorax with abdomen), allowing independent twist of upper and lower body.

Each part has an associated configuration vector defining the part’s position and orientation in 3-space. Placing each part in a global coordinate frame enables the part detectors to operate independently while the full body is assembled by inference over the graphical model. Edges in the graphical model correspond to position and angle relationships between adjacent body parts in space and possibly time, as illustrated in Figure 5.2.

To describe the body by a graphical model, we assume that variables in each node are conditionally independent of those in non-neighboring nodes given the values of the node’s neighbors⁴. Each part/limb is

⁴Self-occlusions of body parts in general violate this assumption. For that purpose, in the next chapter, we introduce occlusion-sensitive likelihoods and edges to model occlusion relationships in addition to other constraints presented here. However, in the case

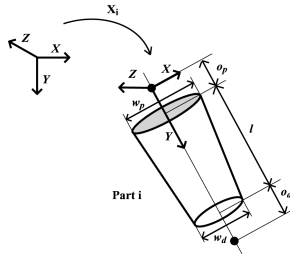


Figure 5.3: **Parameterization of a 3D body part.**

modeled by an elliptical cross-section tapered cylinder having 6 fixed and 6 estimated parameters. The fixed parameters $\Phi_i = [l_i, w_i^p, w_i^d, o_i^p, o_i^d, \epsilon_i^d]$ correspond respectively to the part length, width at the proximal and distal ends, the offset of the proximal and distal joints along the axis of the limb, and eccentricity as shown in Figure 5.3. The offsets, o_i^p and o_i^d , are only used to limit the extent of where the likelihood function is evaluated. In vicinity of a joint, assumptions typically made by the likelihood function are often violated [52].

The estimated parameters $\mathbf{X}_i = [\mathbf{x}_i, \mathbf{q}_i]^T$ represent the configuration of the part i in a global coordinate frame where $\mathbf{x}_i = [x_{x,i}, x_{y,i}, x_{z,i}] \in \mathbb{R}^3$ and $\mathbf{q}_i \in \text{SO}(3)$ are the 3D position of the proximal joint and the angular orientation of the part respectively. The rotations are represented by unit quaternions $\mathbf{q}_i = [q_{x,i}, q_{y,i}, q_{z,i}, q_{w,i}]$, such that $\|\mathbf{q}_i\| = 1$. As a result, $\mathbf{X}_i \in \mathbb{R}^7$, lies on a 6D manifold. The overall pose of the body, \mathbf{X} , for the model with N parts is expressed by the collection of individual part locations and orientations, $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$. This somewhat redundant representation, facilitates distributed inference using Belief Propagation.

Each undirected edge between parts i and j has an associated potential function $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ that encodes the compatibility between pairs of part configurations and intuitively can be thought of as the probability of configuration \mathbf{X}_j of part j conditioned on the \mathbf{X}_i of part i (and vice versa). We introduce two types of potential functions $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ and $\psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j)$, corresponding to kinematic and penetration constraints between parts respectively. In general, these constraints are complex and non-Gaussian. While we only introduce kinematic and penetration potential functions, the framework is general and can handle a variety of other constraints (*e.g.* occlusions [196] and/or motion specific kinematics).

5.3 Constraints

The key to modeling the body using a *loose-limbed body model* is the ability to formulate local spatial (and temporal) coherence constraints for the body parts. To this end, in this section we define the local constraints (*a.k.a.* potential compatibility functions) used in our model.

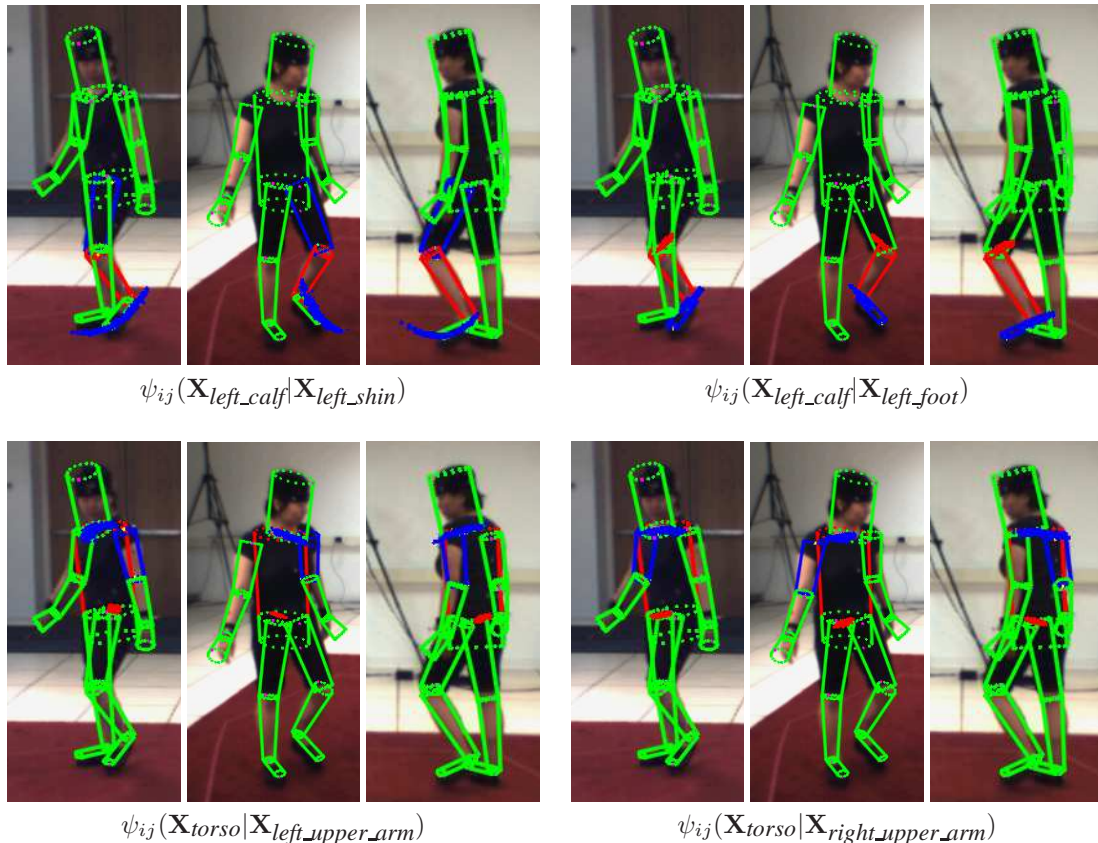


Figure 5.4: **Learned kinematic potentials.** Kinematic potentials are illustrated by sampling from corresponding conditional distributions. The potentials for the left lower leg and subset of potentials for the torso are shown. The figure illustrates distribution of limb positions and orientations conditioned on the ground truth pose for the neighboring limb shown in blue. Blue spheres indicate the proximal joint position of a limb encoded by the sample, while the red spheres indicate the distal end of the limb for each sample. The spread of these samples illustrates the variance of the learned distribution. The ground truth pose for the limb is shown in red.

5.3.1 Kinematic Constraints

The kinematic potentials $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ are in general non-Gaussian and in our framework are approximated by a robust mixture of M_{ij} Gaussian kernels. This modeling choice is motivated by the convenient form for the resulting conditional distributions⁵,

$$\psi_{ij}^K(\mathbf{X}_j|\mathbf{X}_i) = \lambda^0 \mathcal{N}(\mathbf{X}_j|\mu_{ij}, \Lambda_{ij}) + (1 - \lambda^0) \sum_{m=1}^{M_{ij}} \delta_{ijm} \mathcal{N}(\mathbf{X}_j|F_{ijm}(\mathbf{X}_i), G_{ijm}(\mathbf{X}_i)), \quad (5.1)$$

where λ^0 is a fixed outlier probability, μ_{ij} and Λ_{ij} are the mean and covariance of the Gaussian outlier process, and $F_{ijm}(\cdot)$ and $G_{ijm}(\cdot)$ are functions that return the mean and covariance matrix respectively of

of multiple views we found that kinematic and penetration constraints are sufficient to reliably infer the pose. As the number of views decreases, or views become more degenerate, additional occlusion ambiguities will arise and occlusion constraints described in Chapter 6 would have to be added.

⁵Notice that for inference using PAMPAS, we only need conditional distributions, not the full potentials or joint distributions that give rise to these conditionals.

the m -th Gaussian mixture component; $\delta_{ijm} \geq 0$ is the relative weight of an individual component and $\sum_{m=1}^{M_{ij}} \delta_{ijm} = 1$.

One of the challenges in modeling $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ is that part of the state-space $\mathbf{X}_i = [\mathbf{x}_i, \mathbf{q}_i]^T$ corresponding to rotation in 3D lies on Riemannian manifold. A proper model of distribution on the $\text{SO}(3)$ can be achieved using *von Mises-Fisher* distribution [15] (or mixture thereof), which is a generalization of a Gaussian to an arbitrary dimensional spherical shell. For a distribution on a 3-dimensional sphere embedded into \mathbb{R}^4 ,

$$\mathcal{M}(\mathbf{q}_i; \mu, \kappa) = \frac{\kappa}{(2\pi)^2 I_1(\kappa)} \exp(\kappa \mu^T \mathbf{q}_i), \quad (5.2)$$

where μ is the mean direction, $\kappa \geq 0$ is the concentration parameter (similar to variance) and I_1 denotes the modified Bessel function of the first kind of order 1. As with Gaussians the product of the von Mises-Fisher distributions is in itself a von Mises-Fisher distribution. This means that NBP can easily be modified to take into account these distributions on angles. This, however, would lead to additional implementation complexity.

Instead, following [197, 219], we use a linearized approximation for densities that involve \mathbf{q}_i . Hence any distribution over rotations is modeled as a mixture of Gaussian distributions $\in \mathbb{R}^4$. Any sampled orientations from such a distribution may be projected back to $\text{SO}(3)$ by normalizing the corresponding 4-dimensional vector. This approximation works well for points that are tightly concentrated, and tends to over-estimate the variance as they become more spread out over the sphere. Conveniently, since we model the distribution over orientation using a Gaussian mixture, the distribution over the entire state is jointly a Gaussian mixture as well and $F_{ijm}(\mathbf{X}_i) \in \mathbb{R}^7$, $G_{ijm}(\mathbf{X}_i) \in \mathbb{R}^{7 \times 7}$. It remains to be shown how the functions $F_{ijm}(\mathbf{X}_i)$ and $G_{ijm}(\mathbf{X}_i)$ are modeled or learned.

Deriving Kinematic Conditionals from Joint Distributions

Given a set of S ground truth paired state vectors $\mathcal{D} = \{(x_i^{(1)}, x_j^{(1)}), (x_i^{(2)}, x_j^{(2)}), \dots, (x_i^{(S)}, x_j^{(S)})\}$ for neighboring nodes i and j respectively, we can learn the potential compatibility function between the two nodes directly by simply learning the joint distribution $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j) = p(\mathbf{X}_i, \mathbf{X}_j)$. Since $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ is modeled using a Gaussian mixture, we can derive the corresponding conditional distributions needed by PAMPAS analytically. In particular, we can analytically derive $F_{ijm}(\mathbf{X}_i)$ and $G_{ijm}(\mathbf{X}_i)$ functions that give means and covariances of conditional mixture components in Eq. 5.1. For example, assuming that we learn parameters of joint distribution (using for example Expectation-Maximization (EM) procedure for Gaussian Mixture Models (GMMs) described in Algorithm 1), such that

$$p(\mathbf{X}_i, \mathbf{X}_j) = \sum_{m=1}^{M_{ij}} \delta_{ijm} \mathcal{N} \left(\begin{bmatrix} \mathbf{X}_i \\ \mathbf{X}_j \end{bmatrix} \mid \begin{bmatrix} \bar{\mu}_{im} \\ \bar{\mu}_{jm} \end{bmatrix}, \begin{bmatrix} \bar{\Lambda}_{iim} & \bar{\Lambda}_{ijm} \\ \bar{\Lambda}_{jim} & \bar{\Lambda}_{jjm} \end{bmatrix} \right) \quad (5.3)$$

we can define $\psi_{ij}(\mathbf{X}_j | \mathbf{X}_i)$ by analytically deriving $F_{ijm}(\mathbf{X}_i) = \bar{\Lambda}_{jim} [\bar{\Lambda}_{iim}]^{-1} (\mathbf{X}_i - \bar{\mu}_{im})$ and $G_{ijm}(\mathbf{X}_i) = \bar{\Lambda}_{jjm}^{-1} - \bar{\Lambda}_{jim} [\bar{\Lambda}_{iim}]^{-1} \bar{\Lambda}_{ijm}$.

This method of learning potentials, however, has two disadvantages. First, learning the joint distribution, $p(\mathbf{X}_i, \mathbf{X}_j)$, in high dimensional ($\in \mathbb{R}^{14}$) space is hard. Secondly, the joint distribution will undoubtedly encode the prior information for both \mathbf{X}_i and \mathbf{X}_j . Hence, if we train on upright postures, for example, we

would never be able to infer the pose of the person lying down (even if we have observed the full range of motion for all the joints). This is concerning, because instead of natural joint range of motion constraints that kinematic potentials should encode, they will also encode training data specific information that will make it hard for the algorithm to generalize. Instead, what we would like to do is assume a uniform prior over both $p(\mathbf{X}_i)$ and $p(\mathbf{X}_j)$ and learn potentials that will only encode the kinematic joint constraints. This amounts to learning conditional distributions $p(\mathbf{X}_j|\mathbf{X}_i)$ and $p(\mathbf{X}_i|\mathbf{X}_j)$ directly, which is allowed, so long as there exists a common joint distribution that gives rise to the two conditionals. Ideally such learning procedure would ensure that learned conditionals would be symmetric, *i.e.* $p(\mathbf{X}_j|\mathbf{X}_i) \propto p(\mathbf{X}_i|\mathbf{X}_j)$.

Learning Kinematic Conditionals Directly

For convenience, let us first define re-parameterization of the state, \mathbf{X}_i , in terms of homogenized 3D object-to-world matrix transformation,

$$\mathbb{X}_i = H(\mathbf{X}_i) = \begin{bmatrix} 1 - 2q_{y,i}^2 - 2q_{z,i}^2 & 2q_{x,i}q_{y,i} - 2q_{w,i}q_{z,i} & 2q_{x,i}q_{z,i} + 2q_{w,i}q_{y,i} & \mathbf{x}_{x,i} \\ 2q_{x,i}q_{y,i} + 2q_{w,i}q_{z,i} & 1 - 2q_{x,i}^2 - 2q_{z,i}^2 & 2q_{y,i}q_{z,i} + 2q_{w,i}q_{x,i} & \mathbf{x}_{y,i} \\ 2q_{x,i}q_{z,i} - 2q_{w,i}q_{y,i} & 2q_{y,i}q_{z,i} - 2q_{w,i}q_{x,i} & 1 - 2q_{x,i}^2 - 2q_{y,i}^2 & \mathbf{x}_{z,i} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The corresponding inverse transformation $H^{-1}(\cdot)$ that maps back from the 3D object-to-world matrix to our state-space parameterization is somewhat more involved. If the trace, $tr(\mathbb{X}_i) \equiv a_{1,1} + a_{2,2} + a_{3,3} + 1$, where $a_{i,j}$ is the i -th row and j -th column of a 4×4 homogenized matrix \mathbb{X}_i , is ≥ 0 , then the following simple calculation would define the inverse,

$$H^{-1}(\mathbb{X}_i) = \begin{bmatrix} a_{1,4} & a_{2,4} & a_{3,4} & \frac{(a_{3,2} - a_{2,3})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{1,3} - a_{3,1})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{2,1} - a_{1,2})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{\sqrt{tr(\mathbb{X}_i)}}{2} \end{bmatrix}.$$

Otherwise, if $tr(\mathbb{X}_i) \leq 0$, one must look at the major diagonal element and apply the respective inverse transform as follows, $\mathbf{X}_i =$

$$H^{-1}(\mathbb{X}_i) = \begin{cases} \begin{bmatrix} a_{1,4} & a_{2,4} & a_{3,4} & \frac{\sqrt{tr(\mathbb{X}_i)}}{2} & \frac{(a_{1,2} - a_{2,1})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{1,3} - a_{3,1})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{2,3} - a_{3,2})}{2\sqrt{tr(\mathbb{X}_i)}} \end{bmatrix}^T & \text{if } a_{1,1} \geq a_{2,2} \text{ and } a_{1,1} \geq a_{3,3} \\ \begin{bmatrix} a_{1,4} & a_{2,4} & a_{3,4} & \frac{(a_{1,2} - a_{2,1})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{\sqrt{tr(\mathbb{X}_i)}}{2} & \frac{(a_{2,3} - a_{3,2})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{1,3} - a_{3,1})}{2\sqrt{tr(\mathbb{X}_i)}} \end{bmatrix}^T & \text{if } a_{2,2} \geq a_{1,1} \text{ and } a_{2,2} \geq a_{3,3} \\ \begin{bmatrix} a_{1,4} & a_{2,4} & a_{3,4} & \frac{(a_{1,3} - a_{3,1})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{(a_{2,3} - a_{3,2})}{2\sqrt{tr(\mathbb{X}_i)}} & \frac{\sqrt{tr(\mathbb{X}_i)}}{2} & \frac{(a_{1,2} - a_{2,1})}{2\sqrt{tr(\mathbb{X}_i)}} \end{bmatrix}^T & \text{if } a_{3,3} \geq a_{1,1} \text{ and } a_{3,3} \geq a_{2,2} \end{cases}$$

It can be shown that indeed $\mathbf{X}_i = H^{-1}(H(\mathbf{X}_i))$. We can further define relative states \mathbf{X}_{ij} , such that

$$\mathbf{X}_{ij} = H^{-1}([H(\mathbf{X}_i)]^{-1} \times H(\mathbf{X}_j)). \quad (5.4)$$

Intuitively \mathbf{X}_{ij} is the parameterized pose of the part j in part i 's coordinate frame for a particular pair of co-ocurred states. The conditional $\psi_{ij}(\mathbf{X}_j|\mathbf{X}_i)$ can then be expressed as a transformed distribution over \mathbf{X}_{ij} .

We can learn a Gaussian mixture distribution for \mathbf{X}_{ij} using the Expectation-Maximization procedure. As a result,

$$p(\mathbf{X}_{ij}) = \sum_{m=1}^{M_{ij}} \delta_{ijm} \mathcal{N}(\mathbf{X}_{ij}; \mu_{ijm}, \Lambda_{ijm}) \quad (5.5)$$

we learn parameters $\{\delta_{ijm}, \mu_{ijm}, \Lambda_{ijm}\}_{m=1}^{M_{ij}}$, where μ_{ijm} is the mean, Λ_{ijm} a covariance, and $\sum_{m=1}^{M_{ij}} \delta_{ijm} = 1$ are weights for the mixture components of the transformed distribution. We can then define the transformation functions explicitly $F_{ijm}(\mathbf{X}_i) = R(\mathbf{X}_i) * \mu_{ijm} + T(\mathbf{X}_i)$ and $G_{ijm}(\mathbf{X}_i) = (\Lambda_{ijm}^{-1} * R(\mathbf{X}_i))^{-1}$, where $T(\mathbf{X}_i) = [\mathbf{x}_i, 0, 0, 0]^T$ and

$$R(\mathbf{X}_i) = \begin{bmatrix} 1 - 2q_{y,i}^2 - 2q_{z,i}^2 & 2q_{x,i}q_{y,i} - 2q_{w,i}q_{z,i} & 2q_{x,i}q_{z,i} + 2q_{w,i}q_{y,i} & 0 & 0 & 0 & 0 \\ 2q_{x,i}q_{y,i} + 2q_{w,i}q_{z,i} & 1 - 2q_{x,i}^2 - 2q_{z,i}^2 & 2q_{y,i}q_{z,i} + 2q_{w,i}q_{x,i} & 0 & 0 & 0 & 0 \\ 2q_{x,i}q_{z,i} - 2q_{w,i}q_{y,i} & 2q_{y,i}q_{z,i} - 2q_{w,i}q_{x,i} & 1 - 2q_{x,i}^2 - 2q_{y,i}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{w,i} & -q_{x,i} & -q_{y,i} & -q_{z,i} \\ 0 & 0 & 0 & -q_{x,i} & q_{w,i} & -q_{z,i} & q_{y,i} \\ 0 & 0 & 0 & -q_{y,i} & -q_{z,i} & q_{w,i} & -q_{x,i} \\ 0 & 0 & 0 & -q_{z,i} & q_{y,i} & -q_{x,i} & q_{w,i} \end{bmatrix}.$$

Intuitively, for a given value of $\mathbf{X}_i = [\mathbf{x}_i, \mathbf{q}_i]^T$, the top-left block will transform the translation component of the mean and covariance via a rotation matrix defined by the \mathbf{q}_i and the bottom-right block will transform the quaternion rotation component of the mean and covariance via the Grassman product.

While our learning algorithm is general enough to learn distributions that have couplings between positional and rotational components of the state space, resulting in full-covariance matrices, for computational purposes we restrict ourselves to the block-diagonal covariance distributions.

Figure 5.4 shows a few of the learned conditional distributions. Samples are shown from several limb-to-limb conditionals. For example, the lower leg distribution is shown conditioned on the pose of the upper leg. The proximal end of the shin (green circle) is predicted with high confidence given the thigh location, but there is a wide distribution over possible ankle locations, as expected.

5.3.2 Penetration Constraints

Another important constraint that needs to be modeled is interpenetration between limbs. Since the body consists of convex solid parts, they cannot physically penetrate each other. To model this we define a set of pair-wise constraints between the parts that are most likely to penetrate, given the kinematics of the body. In the limit we could consider all pairs of parts, which would result in an inference algorithm that is quadratic instead of linear in the number of parts. Instead, as a simplification, we only allow for most likely penetration scenarios that arise in upright motions such as walking, running, dancing and *etc.*

Let us consider the penetration constraints we want to encode. Given a configuration of part i , \mathbf{X}_i , we want to allow potentially penetrating part j to be anywhere so long as it does not penetrate part i in it's current configuration. This means that non-penetration constraints are hard to model using a Mixture of Gaussians [197], since we need to model equal probability over the entire state space, and zero probability in some local region around the pose \mathbf{X}_i . Instead we model the penetration potentials using the following unnormalized distribution

$$\psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j) \propto 1 - \varrho(\mathbf{X}_i, \mathbf{X}_j) \quad (5.6)$$

where $\varrho(\mathbf{X}_i, \mathbf{X}_j)$ is the probability that part i in configuration \mathbf{X}_i penetrates part j in configuration \mathbf{X}_j and is defined to be 1 if and only if i penetrates j in their respective configurations (0 otherwise). Notice that we can encode soft-penetration constraints by allowing $\varrho(\mathbf{X}_i, \mathbf{X}_j)$ to assume any value from 0 to 1 as a function of the overlap between parts. In our experiments, however, hard penetration constraints proved to be more effective.

There are a number of ways one can detect and measure 3D overlap between two body parts. Constructive solid geometry (CSG) [63, 245] can use boolean operators applied on a set of truncated cone primitives, that we use for modeling body parts, to principally do this. CSG methods, however, tend to be relatively expensive and tricky to implement. Instead, we experimented with two simple approximations: spherical and voxel. Spherical approximation, approximates truncated cones with a sparse set of spherical⁶ shells with corresponding non-constant radii. The set of shells approximating part i are then exhaustively intersected with the shells modeling part j . Since intersection of the two spheres can be computed using a simple euclidian distance operator between the centroids, this process tends to be very efficient. However, this approximation is only well suited for determining the presence or absence of the intersection between two parts, not the amount of intersection. If one needs to compute the amount of intersection, one alternative is to partition the space occupied by one of the limbs into a set of 3D voxels and compute the approximate volume of intersection by checking whether each voxel grid point lies within potentially penetrating limb. Since we found hard penetration constraints to be more robust, we employ the simpler spherical approximation that avoids additional computational complexity of the latter method.

5.4 Image Likelihoods

The inference algorithm, the details of which will be outlined in the next section, combines the body model described above with a probabilistic image likelihood model. We define $\phi_i(\mathbf{X}_i) \equiv \phi_i(I|\mathbf{X}_i)$ to be the likelihood of observing the image measurements conditioned on the pose of limb i . Ideally this model would be robust to partial occlusions, the variability of image statistics across different input sequences, and variability among subjects. To that end, we combine a variety of generic non-clothes specific cues including silhouettes and edges.

5.4.1 Foreground Likelihood

Most algorithms that deal with 3D human motion estimation [1, 14, 50, 52, 59, 196, 197] rely on silhouette information for image likelihoods. Indeed this is a very strong cue [14] that should be taken into account when available. Here, as in most prior work, we assume that a foreground/background separation process exists that computes a binary mask $FG_c(x, y)$, where $FG_c(x, y) = 1$ if and only if pixel (x, y) in an image I belongs to the foreground for a given camera view $c \in [1, \dots, C]$.

⁶3D ellipsoids can be used instead, for parts that have elliptical cone cross section, with similar complexity.

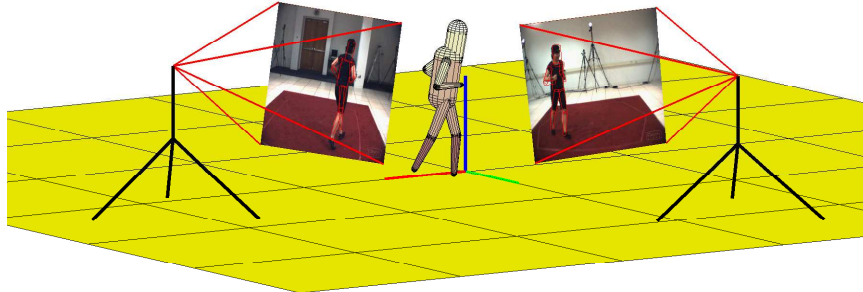


Figure 5.5: **Backprojecting the 3D body model.** Illustrated is the process used to project the 3D body model (consisting of a set of connected limbs) into a number of calibrated image views. For clarity, only 2 out of a total of 7 views are shown.

Formally, we assume that pixels in the image (and hence foreground binary mask) can be partitioned into three disjoint sub-sets (see Figure 5.6 (c)), $\Omega_{c,1}(\mathbf{X}_i) \cup \Omega_{c,2}(\mathbf{X}_i) \cup \Omega_{c,3}(\mathbf{X}_i)$; where $\Omega_{c,1}(\mathbf{X}_i)$ is the set of pixels enclosed by the projection of the part i at pose \mathbf{X}_i onto camera view c ; $\Omega_{c,2}(\mathbf{X}_i)$ contains pixels slightly outside part i that are statistically correlated with the part; and $\Omega_{c,3}(\mathbf{X}_i)$ are pixels that are not correlated with part i in any way. Assuming pixel independence and independence of observations across camera views we can write the likelihood of the image given the pose of the part as

$$\phi_{fg}(I|\mathbf{X}_i) \propto \prod_{c=1}^C \left[\prod_{(x,y) \in \Omega_{c,1}(\mathbf{X}_i)} p_1(FG_c(x,y)) \prod_{(x,y) \in \Omega_{c,2}(\mathbf{X}_i)} p_2(FG_c(x,y)) \prod_{(x,y) \in \Omega_{c,3}(\mathbf{X}_i)} p_3(FG_c(x,y)) \right], \quad (5.7)$$

where p_i , $i \in \{1, 2, 3\}$ are the region-specific probabilities learned from the set of labeled images. In general, $p_1(FG_c(x,y) = 1) > 0.5$, $p_2(FG_c(x,y) = 1) < 0.5$ and $p_3(FG_c(x,y) = 1) = 0.5$, corresponding to the observation that pixels enclosed by projection of the part tend to be segmented as part of foreground silhouette and pixels slightly outside typically correspond to background. Reasoning about pixels that are outside of the immediate vicinity of the part's projection is often hard, because other parts or foreground objects may be present in the scene. To deal with this we assume equal probability for these regions, *i.e.* $p_3(FG_c(x,y) = 1) = 0.5$. Furthermore, to simplify our likelihood model for all our experiments in this chapter we used the following learned values for all limb likelihoods (avoiding learning separate values for each part),

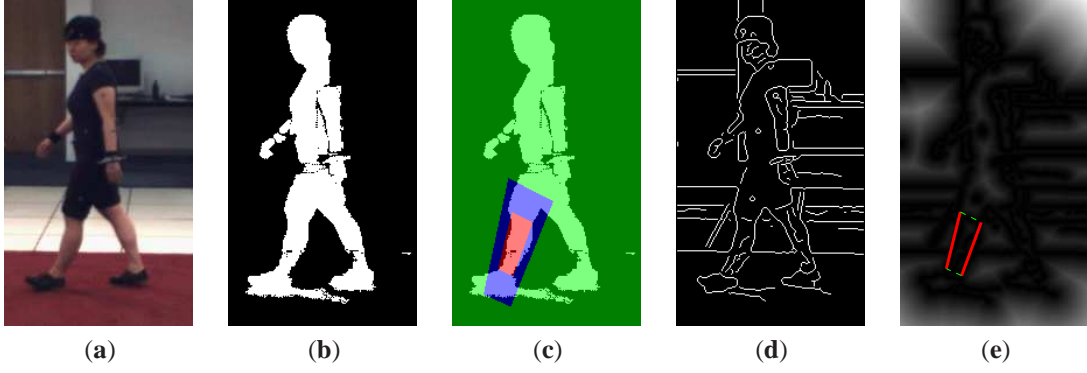


Figure 5.6: **Image likelihoods.** Illustrated is the original image (a) and the likelihood features for computing the left lower leg likelihood; (b) illustrates the silhouette obtained by background subtraction; (c) shows the partition of the silhouette image pixels into three disjoint sub-sets where red, blue and green pixels correspond to $\Omega_{c,1}$, $\Omega_{c,2}$, and $\Omega_{c,3}$ regions respectively. The edge image obtained by Canny edge detector and the corresponding distance transform for the edge image are shown in (c) and (d) respectively. In (d) the projected model edge pixels for which the edge likelihood is computed are shown in solid red.

$$\begin{aligned} p_1(FG_c(x, y) = 1) &= 0.8 \\ p_2(FG_c(x, y) = 1) &= 0.3 \\ p_3(FG_c(x, y) = 1) &= 0.5. \end{aligned}$$

Notice that since $FG_c(x, y)$ is binary, $p_i(FG_c(x, y) = 0) = 1 - p_i(FG_c(x, y) = 1)$ for $i \in \{1, 2, 3\}$.

5.4.2 Edge Likelihood

Even with perfect background subtraction, the silhouettes alone are ambiguous. For example, motion or position of occluding parts may not be observed (this was already illustrated in Figure 2.2). This ambiguity is reduced as the number of views increase, but with few cameras such as the case here the effects are still significant. Hence, to reduce ambiguity and better localize parts, we also use an edge-based likelihood measure.

We start by computing an edge distance transform, $EDGE_c(x, y)$, by first running the Canny edge detector on the image (obtained from camera c) and then computing a distance transform based on the resulting binary edge image. The edge based likelihood measure is then defined as follows, once again assuming independence across pixels and camera views,

$$\phi_{edge}(I|\mathbf{X}_i) \propto \prod_{c=1}^C \left[\prod_{(x,y) \in \Gamma_c(\mathbf{X}_i)} \exp(1/EDGE_c(x, y)^2) \right], \quad (5.8)$$

where $\Gamma_c(\mathbf{X}_i)$ is the set of pixels on the left and right edge of part's projection in camera view c . Particularly, since we model parts in 3D using tapered right elliptical cones, $\Gamma_c(\mathbf{X}_i)$ will correspond to the two opposite edges of trapezoid obtained by 2D projection of the cone cross-section onto the image plane. This is illustrated in Figure 5.6 (e).

5.4.3 Combining Features

To produce the final likelihood measure $\phi_i(I|\mathbf{X}_i)$, that takes into account both foreground and edge features, we must fuse the two likelihood terms. However, we must also account for different *a priori* confidence exhibited by the two features. In particular, foreground features are in general much more reliable than edge features [14] (assuming a reasonably reliable foreground/background separation process). Taking this into account, results in the following weighted likelihood measure,

$$\phi_i(I|\mathbf{X}_i) = [\phi_{fg}(I|\mathbf{X}_i)]^{1-w_e} [\phi_{edge}(I|\mathbf{X}_i)]^{w_e}, \quad (5.9)$$

where w_e is the relative confidence weight for the edge term. In practice we found $w_e = 0.1$ worked reasonably well.

5.5 Bottom-up Part Detectors

Occlusion of body parts, changes in illumination, and a myriad of other situations may cause a person tracker to lose track of some, or all, parts of a body. We argue that reliable tracking requires bottom-up processes that constantly searches for body parts and suggest their location and pose to the tracker; we call these “shouters”⁷. This bottom-up process is also useful in bootstrapping the inference, by providing initial distributions over locations of a sub-set of parts. Further discussion of this in the context of Particle Message Passing can be found in Section 3.7.3.

One expects shouters to be noisy in that they will sometimes fail to detect parts or will find spurious parts. Furthermore they will probably not be able to differentiate between left and right extremities of the body. Both of these behaviors can be seen in Figure 5.8. However, even these noisy “guesses” provide valuable low-level cues, and our belief propagation framework is designed to incorporate this bottom-up information in a principled way. As will be described in detail in Section 5.6, we use a stratified sampler for approximating messages originating at graph node i and being sent to node j at time t . This sampler draws some fraction of samples from a static importance function $q_{ij}(\mathbf{X}_i) = f(\mathbf{X}_i)$. This importance function is constructed by the node’s shouter process, that we denote by $f(\mathbf{X}_i)$, and draws samples from locations in pose space (3D location and orientation) near the detected body parts.

5.5.1 Head Detection

We build head detector based on the Viola and Jones face detector [236]. We use two models for frontal and profile faces, and apply them in multiple-views to produce plausible estimates for the position and orientation of the head (see Figure 5.7).

We first detect a set of 2D face candidates in all views, by running the two detectors at a number of scales (Figure 5.7 (top)). We then try to pair up candidates from different views, assuming known extrinsic calibration estimated off-line for all cameras. The pose of the head can then be estimated by intersecting the frustums mapped by the two face candidates in the 3D space. The orientation about the head axis is refined, to about 45° precision, by considering the types of the faces found in the two views. For example, frontal face observed from one camera paired with profile face found in the other, will result in the overall

⁷The idea of “shouters” came about through discussions with A. Jepson and D. Fleet.

head orientation pointing toward the camera that observed the frontal view in the first place; a frontal face observed from two different cameras will result in a pose of the head where face is pointing between the two cameras considered.

Once the 3D candidates are estimated, they are pruned by checking to ensure that the size is plausible (within limits for a human head) and that candidates project to (mostly) foreground regions in all the views. As a result of this process a set of plausible candidate poses for the head are constructed, $\{x_{head}^{(1)}, x_{head}^{(2)}, \dots, x_{head}^{(N_{head})}\}$, where N_{head} is the total number of plausible head candidates selected. The proposal function, $f(\mathbf{X}_{head})$, for the head is constructed simply by formulating kernel density based on the candidates,

$$f(\mathbf{X}_{head}) = \sum_{n=1}^{N_{head}} \mathcal{N}(\mathbf{X}_{head} | x_{head}^{(n)}, \Lambda_{head}), \quad (5.10)$$

where covariance Λ_{head} is a function of the overall head detector’s precision. In general, Λ_{head} should be estimated from training data, however, since a labeled dataset with ground truth 3D head positions is not readily available, instead we set Λ_{head} by hand. We set diagonal elements of Λ_{head} , that account for variance in the estimated position and tilt of the head, relatively small and twist (rotation about the head axis) is set to a considerably larger value to account for 45° uncertainty discussed above; all off diagonal elements of Λ_{head} are set to 0.

5.5.2 Limb Detection

Unlike faces, limbs lack distinctive 3D shape and texture structure that is consistent across people and clothing. We attempt to build limb proposals based on color information [147], by assuming that limbs have roughly uniform color⁸. To this end, we first segment foreground regions of each view into a set of coherent color blobs using a mean-shift image segmentation procedure [42]. We then fit ellipses to these regions and intersect frustums produced by the elliptical image regions in 3D. The intersection gives a rough estimate for the position and orientation of the limb (modulo the twist of the limb along its axis of symmetry, which is typically unobservable at standard video resolutions). Similar to head detection, we use the sizes of the estimated 3D limbs to prune the number of candidates to a set of plausible limb positions and orientations $\{x_{limb}^{(1)}, x_{limb}^{(2)}, \dots, x_{limb}^{(N_{limb})}\}$. Also, similar to the head, we form the proposal function for the limbs using a kernel density,

$$f(\mathbf{X}_{limb}) = \sum_{n=1}^{N_{limb}} \mathcal{N}(\mathbf{X}_{limb} | x_{limb}^{(n)}, \Lambda_{limb}). \quad (5.11)$$

As a result all limbs have the same proposal function and it is up to the inference and spatial (and possibly temporal) consistency constraints to interpret their identity in the context of the human body. While the inference algorithm proposed here can deal with this task, we found that this often requires many samples and results in slow convergence. Instead, since we typically are interested in dealing with mostly upright poses we modify the above proposal function as follows,

⁸Clearly this assumption can easily be violated by the various types and textures of clothing, however, one would hope that it will hold for at least some sub-set of limbs considered.

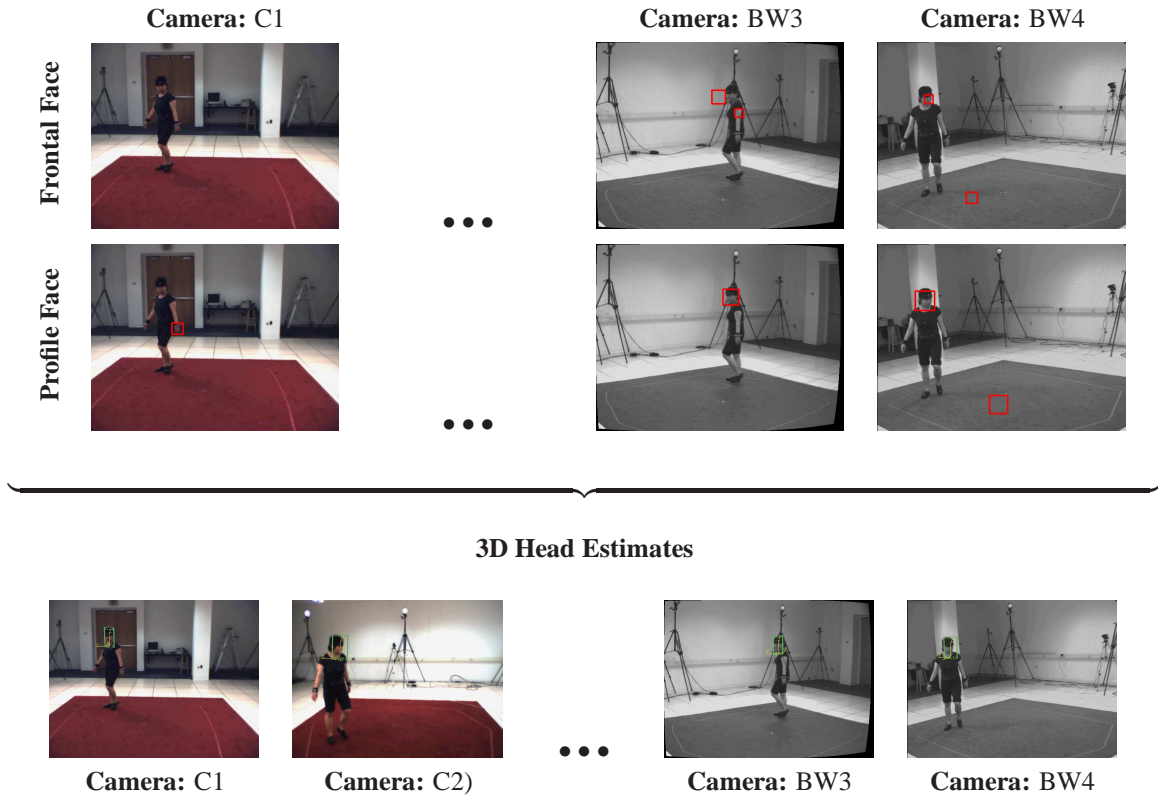


Figure 5.7: **Head detection.** Top two rows show results of the Viola and Jones [236] frontal and profile face detectors respectively, run in high precision low recall modality. We use pre-trained detectors distributed with Intel’s OpenCV library [159]. Bottom row shows 3D head estimates obtained by combining the face detection results from multiple views. The red bounding boxes on images in the top two rows illustrate detected faces. The green bounding boxes on the bottom row are projections of the 3D hypotheses for the head position and orientation; in yellow are the corresponding coordinate frames.

$$f(\mathbf{X}_i) = \sum_{n=1}^{N_{limb}} \mathcal{N}(\mathbf{x}_{z,i}|z_i, \Lambda_i) \mathcal{N}(\mathbf{X}_i|x_{limb}^{(n)}, \Lambda_{limb}), \quad (5.12)$$

where $\mathcal{N}(\mathbf{x}_{z,i}|z_i, \Lambda_i)$ can be interpreted as the weighting function that weighs detections as belonging to one of the body parts based on the vertical distance from the floor⁹, z_i . Notice that the proposed weighting is simply a bias that helps to identify which proposed part positions are likely to belong to upper or lower extremities. These biases are the same for left and right sides of the body and hence result in equivalent proposal functions for the two sides.

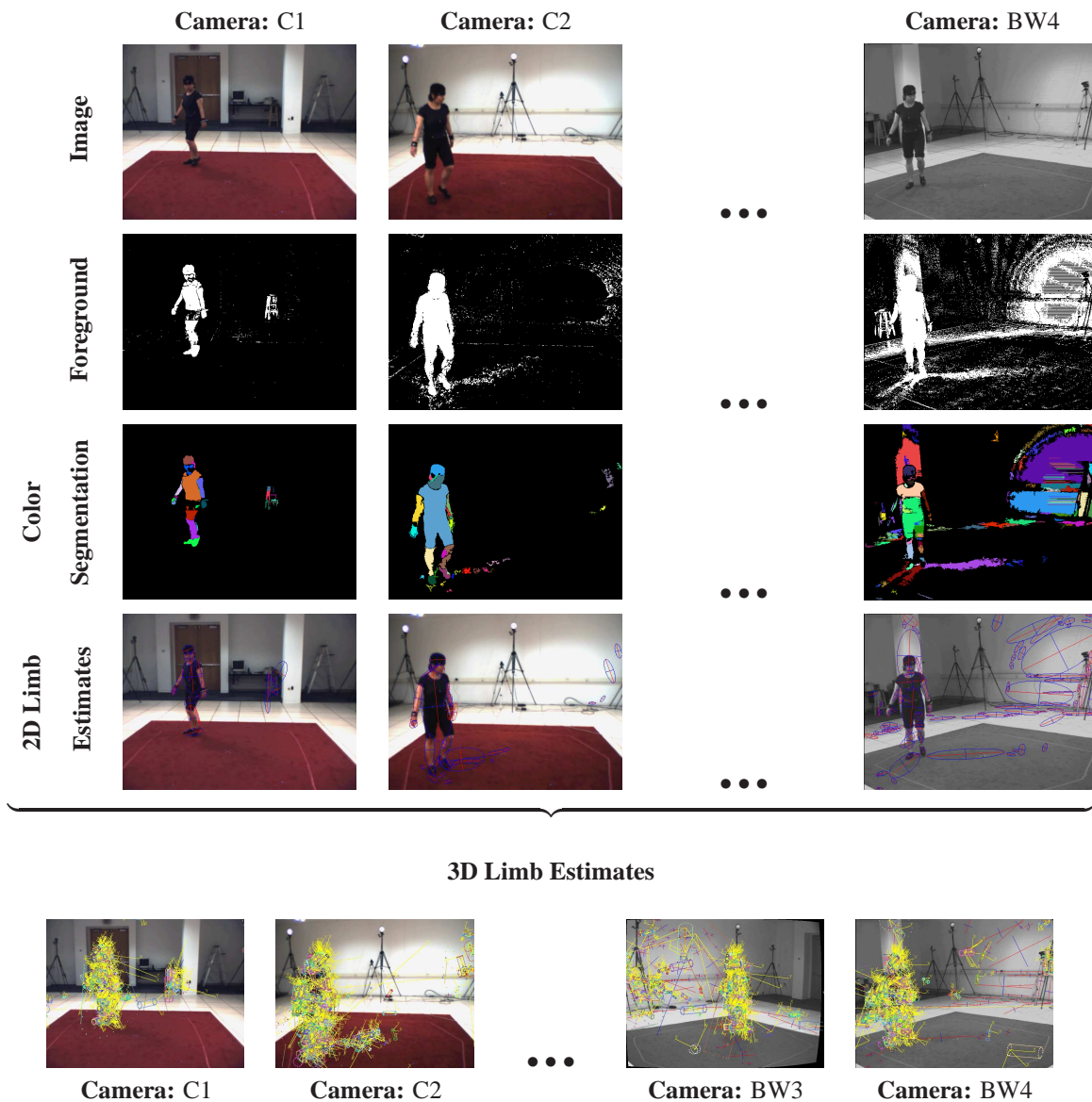


Figure 5.8: **Limb detection.** Top row shows the original images from 3 out of 7 camera views. Results of foreground/background segmentation and mean-shift clustering for color segmentation of foreground regions are shown in second and third rows respectively. Colors are assigned to the region segments at random. Fourth row shows an elliptical 2D limb fit to the regions detected; last row shows the resulting 3D limb estimates produced by combining the 2D estimates across different views.

5.6 Inference

The joint distribution over all variables in our model, defined by the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with vertices \mathcal{V} , $|\mathcal{V}| = N$, corresponding to body parts and edges \mathcal{E} corresponding to constraints, can be written as follows:

⁹This implicitly assumes the world coordinate system is either aligned with the floor or is known. This assumption, while improves the efficiency and performance of our algorithm, is not strictly necessary. One can use the more general form of the proposal function from Eq. 5.11 that assumes no knowledge of terrain.

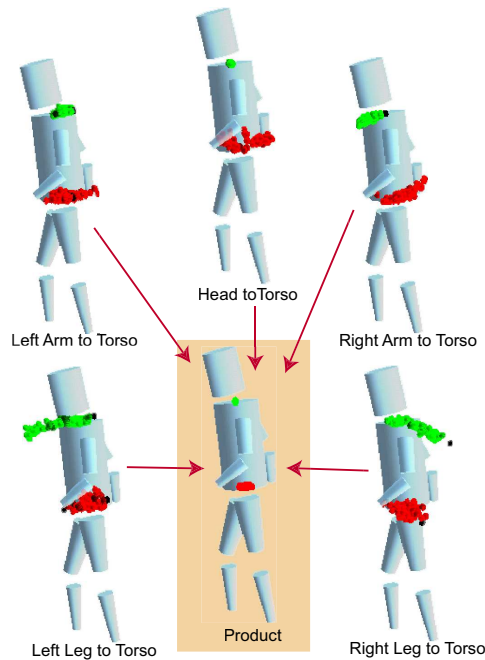


Figure 5.9: **Message product for the torso.** In the 10-part body model, the head, upper arms, and upper legs send messages to the torso. Samples from these messages are illustrated by showing the predicted torso location with green balls. The distribution over the orientation of the torso is illustrated by showing a red ball at the distal end of the torso for each sample. While any single message represents uncertain information about the torso pose, the product of these messages tightly constrains the torso position and orientation.

$$p(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N | I) = \prod_{i \in \mathcal{V}} \phi_i(\mathbf{X}_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j) \psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j). \quad (5.13)$$

Pose estimation and tracking can then be formulated as inference in this graphical model and estimated using Belief Propagation. To cope with the continuous parameter space of each limb, the non-Gaussian conditionals between nodes, and the non-Gaussian likelihood, we use a form of non-parametric belief propagation [99, 220] described in Section 3.7. The approach is a generalization of particle filtering [54] which allows inference over arbitrary graphs rather than a simple chain. In this generalization the “message” used in standard belief propagation is approximated with a smoothed particle set, and the conditional distribution used in standard particle filtering is replaced by a product of incoming message sets. The two formulations of [99] and [220] have different strengths discussed in Section 3.7.5; we adopt the PAMPAS algorithm because it maps better to our models where the potentials are small mixtures of Gaussians and the likelihoods are simple to evaluate up to an unknown normalization. NBP [220] is more suitable for applications with complex potential functions.

The message passing framework is illustrated in Figure 5.9 where the head, upper arms and upper legs all send messages to the torso. These messages are distributions that are represented by a set of weighted samples as in particle filtering (smoothed with a Gaussian kernel). Belief propagation requires forming the product of these incoming messages. As Figure 5.9 shows, the individual limbs may not constrain the torso very precisely. The product over all the incoming messages, however, produces a very tight distribution over the torso pose. The challenge in Particle Message Passing (and non-parametric belief propagation in general)

is to compute the product of multiple such sampled distributions efficiently. If each sample is represented by a small Gaussian kernel, then the explicit product of D messages each containing N kernels requires $O(N^D)$ time to compute, which is impractical in most cases where $D > 2$. Hence, for $D > 2$ we use the Gibbs sampler described in Section 3.7.1 (Algorithm 5) that can draw approximate and asymptotically unbiased samples from the product in $O(DN^2)$. For $D \leq 2$ we compute the message product explicitly.

In PAMPAS the belief propagation messages are approximated using Monte-Carlo importance sampling. This is achieved by sampling from the product of messages and then propagating these samples through an appropriate potential function. Since in our *loose-limbed body model* formulation we have two types of potential functions, $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ and $\psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j)$ that have different representations, the messages in the two cases will be different as well,

$$m_{ij}^K(\mathbf{X}_j) = \int \psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}^K(\mathbf{X}_i) m_{ki}^P(\mathbf{X}_i) d\mathbf{X}_i \quad (5.14)$$

$$m_{ij}^P(\mathbf{X}_j) = \int \psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j) \phi_i(\mathbf{X}_i) \prod_{k \in A(i) \setminus j} m_{ki}^K(\mathbf{X}_i) m_{ki}^P(\mathbf{X}_i) d\mathbf{X}_i \quad (5.15)$$

where $A(i)$ is the set of neighbors of node i and $\phi_i(\mathbf{X}_i)$ is the local likelihood associated with node i . Note that $m_{ij}^K(\mathbf{X}_j)$ is represented using a mixture of Gaussian kernel densities and $m_{ij}^P(\mathbf{X}_j)$ by a mixture of continuous unnormalized functions. These representations stem from the choice of potential functions discussed in Sections 5.3.1 and 5.3.2 respectively. The stratified PAMPAS algorithm (see Algorithm 6) can be modified to handle this case by choosing proper importance functions. In particular,

$$q_{ij}^{(1)}(\mathbf{X}_i) = \prod_{k \in A(i) \setminus j} m_{ki}^K(\mathbf{X}_i) \quad (5.16)$$

$$q_{ij}^{(2)}(\mathbf{X}_i) = \prod_{k \in A(i)} m_{ki}^K(\mathbf{X}_i) \quad (5.17)$$

$$q_{ij}^{(3)}(\mathbf{X}_i) = f(\mathbf{X}_i). \quad (5.18)$$

The messages, $m_{ij}^K(\mathbf{X}_j)$ and $m_{ij}^P(\mathbf{X}_j)$, are hence generated by sampling from the above importance functions, applying a proper re-weighting scheme, and propagating these weighted samples that intuitively account for distribution over \mathbf{X}_i through a potential function $\psi_{ij}^K(\mathbf{X}_i, \mathbf{X}_j)$ or $\psi_{ij}^P(\mathbf{X}_i, \mathbf{X}_j)$ respectively. For further details see Section 3.7.

Illustration of PAMPAS being utilized for pose estimation with 10-part loose-limbed body model can be seen in Figure 5.10. In Figure 5.10 marginals are illustrated in terms of the most likely sample drawn from the marginal, on the left, and the full distribution visualized by overlapping samples on the right. In all images the dark and light green illustrate parts belonging to the left and right sides of the body respectively; yellow illustrates coordinate frames for the torso and the head. One can clearly see how the marginals converge to the desired solution within the first 5-6 iterations.

The basic algorithm outlined in Section 3.7 leaves open questions of **(1)** what proportions to use in the stratified sampler, **(2)** how many particles to use for Monte Carlo approximation of each message and **(3)** what order to use in updating the messages.

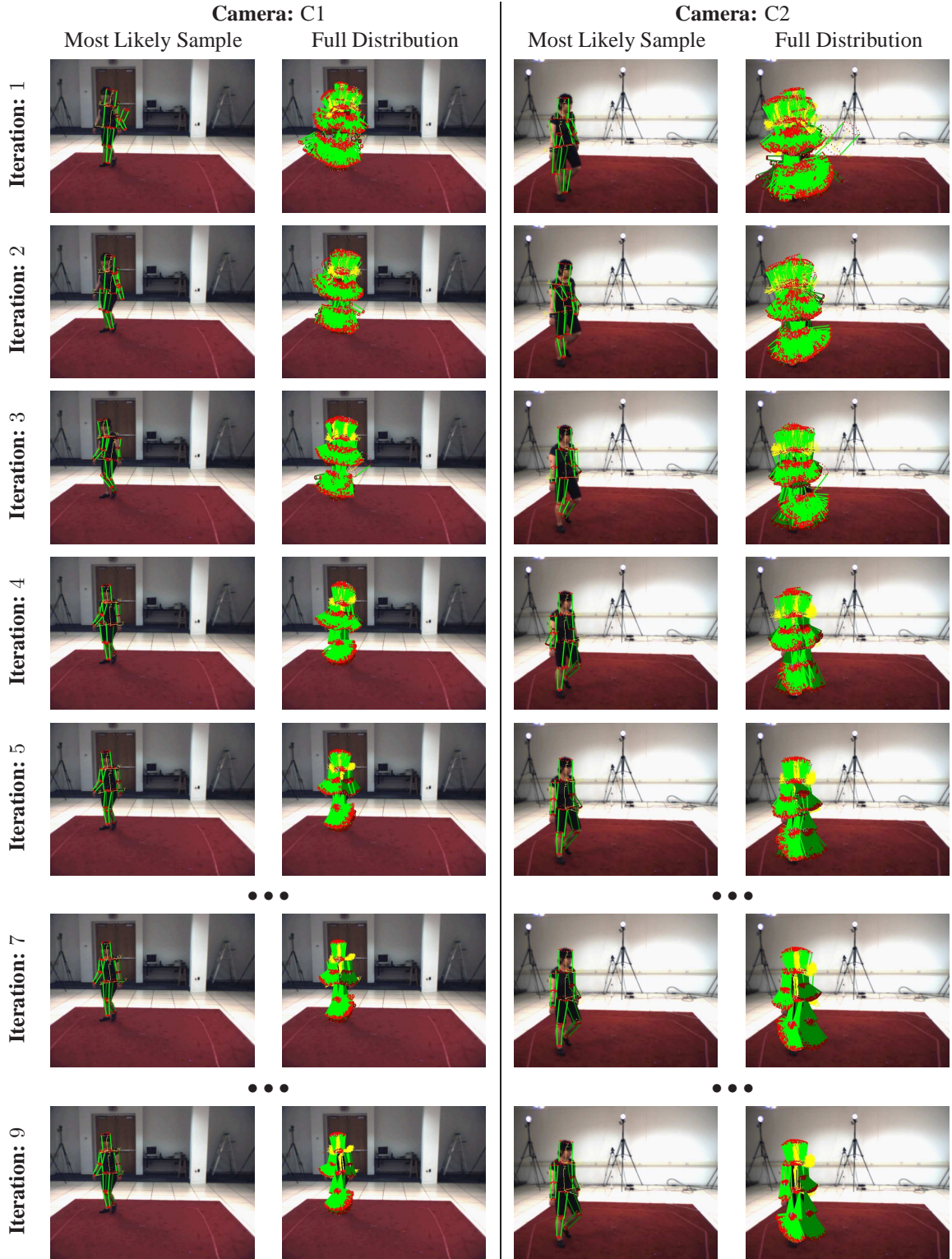


Figure 5.10: **Illustration of convergence of loose-limbed body model during pose estimation.** Resulting marginals for each limb estimated by PAMPAS are illustrated after 1-5, 7 and 9 message passing iterations. As can be observed, marginals converge to a desired solution in roughly 5 iterations in this case, after which point the marginals are refined without significantly affecting the mode of the marginal distribution. The error curve as a function of PAMPAS iterations for this frame can be found in Figure 5.13 (Frame 450).

Implementation Details

Sampling Proportions. The stratified sampler we use, samples all the samples from $q_{ij}^{(3)}(\mathbf{X}_i)$ for the first message passing iteration and then samples half of samples from $q_{ij}^{(1)}(\mathbf{X}_i)$ and half from $q_{ij}^{(2)}(\mathbf{X}_i)$ for the remaining iterations. We found the sampling from $q_{ij}^{(2)}(\mathbf{X}_i)$ sometimes lead to faster convergence, where as sampling from $q_{ij}^{(1)}(\mathbf{X}_i)$ often leads to better results when the solution is close to convergence. Consequently we have also experimented with adapting sampling proportions, using an annealing schedule based on the number of message passing iterations. The idea being, that while in the beginning where there is a large uncertainty about the solution, we should sample equally from $q_{ij}^{(1)}(\mathbf{X}_i)$ and $q_{ij}^{(2)}(\mathbf{X}_i)$, as the solutions starts to converge (assuming it is converging with iterations of BP) we should sample more from $q_{ij}^{(1)}(\mathbf{X}_i)$. While this proved to be useful in some instances, it also sometimes introduced biases particularly when the stratum corresponding to the $q_{ij}^{(2)}(\mathbf{X}_i)$ was small. Hence, for simplicity, for all experiments here we use equal sampling fractions for $q_{ij}^{(1)}(\mathbf{X}_i)$ and $q_{ij}^{(2)}(\mathbf{X}_i)$.

Number of samples. The number of particles/samples used to approximate messages has a significant effect on the runtime of the algorithm. While the basic Particle Message Passing algorithm assumes that all messages are approximated using the same number of N samples, we found this to be sub-optimal. In particular, we found that messages going out of the nodes that are highly connected (*e.g.* torso) are often more compact and require fewer samples to represent adequately; alternatively, messages that correspond to outer nodes in the graph, that have fewer connections, need more samples to be adequately represented. Hence, we derived an ad-hoc adaptive procedure for the number of samples required to represent the message based on the degree of the node sending the message. In particular, for all experiments we used the following number of samples to approximate messages sent from node i :

Node i	# of samples	Mixtures in potential	Message representation
torso	50	Kinematic: 4	$m_{ij}^K(\mathbf{X}_j)$ = mixture of 201 Gaussian kernels
		Penetration: 1	$m_{ij}^P(\mathbf{X}_j)$ = 1 - mixture of 50 Gaussian kernels
head, shins, upper arms	200	Kinematic: 4	$m_{ij}^K(\mathbf{X}_j)$ = mixture of 801 Gaussian kernels
		Penetration: 1	$m_{ij}^P(\mathbf{X}_j)$ = 1 - mixture of 200 Gaussian kernels
calfs, lower arms	800	Kinematic: 4	$m_{ij}^K(\mathbf{X}_j)$ = mixture of 2401 Gaussian kernels
		Penetration: 1	$m_{ij}^P(\mathbf{X}_j)$ = 1 - mixture of 800 Gaussian kernels
(In addition for 15-part model)			
hands, feet	800	Kinematic: 4	$m_{ij}^K(\mathbf{X}_j)$ = mixture of 2401 Gaussian kernels
		Penetration: 1	$m_{ij}^P(\mathbf{X}_j)$ = 1 - mixture of 800 Gaussian kernels

Note that penetration messages due to their non-Gaussian form are simply treated in the PAMPAS framework as continuous functions. Deriving automatically the number of samples required for each message would clearly be of benefit, however, this is hard to do in general, since the number of samples must be a function of the overall graph topology, importance functions employed for Monte Carlo integration, and distributions of all involved variables.

Message passing schedule. As mentioned in Section 3.7 in order to perform inference in a loopy graphical model, one needs to define the message update schedule. We use a fixed message update schedule that sends messages from the outer extremities inward toward the torso and then back out (from the torso to outer

extremities). We also first propagate the kinematic messages and then the penetration messages. For pose estimation we run PAMPAS for 10 message passing iterations per frame (with convergence often achieved in 5–6 iterations), and for tracking (that will be discussed in the next section) for only 2 message passing iterations per frame. The underlying assumption being that in the tracking framework the pose is likely to be relatively well constrained by the estimate from the previous time frame, and hence PAMPAS often converges faster.

5.6.1 Tracking

Thus far we have only addressed the problem of pose estimation and have not dealt with incorporating temporal consistency into our model. This is partly due to the fact that ability to automatically estimate the pose is one of the key benefits of the *loose-limbed body model*. However, our model can also incorporate temporal consistency. Temporal consistency (or tracking) can be performed in at least two different ways within our framework.

Tracking using a spatio-temporal model

The most direct way of extending the proposed pose estimation framework to tracking, is by replicating and chaining the spatial loose-limbed body model across time. This methodology was already introduced in somewhat different context in Chapter 4. The new spatio-temporal graphical model requires additional temporal constraints between limbs at time $t - 1$ and t , that we denote by $\psi^T(\mathbf{X}_{i,t-1}, \mathbf{X}_{i,t})$. Typically a single Gaussian potential is sufficient to model these temporal constraints. For example,

$$\psi^T(\mathbf{X}_{i,t-1}, \mathbf{X}_{i,t}) = \mathcal{N}(\mathbf{X}_{i,t} - \mathbf{X}_{i,t-1} | 0, \Lambda_T), \quad (5.19)$$

is equivalent to a zero velocity assumption. With this type of temporal constraint, inference can be performed as before using Particle Message Passing in either batch or sliding window fashion. We have explored this alternative in [197]. A similar approach has also been discussed in the context of generic object tracking in Chapter 4.

The benefit of this type of spatio-temporal model is that temporal consistency is well maintained, the disadvantage is the additional computational complexity incurred by this more complex model. In addition, if tracking fails, the spatio-temporal model is often harder to re-initialize, because of the tight coupling to the pose at the previous time instants.

Tracking using importance sampling

An alternative approach, that we take in this chapter, is to propagate the temporal consistency via an importance function. This approach does not alter the model already introduced, and hence does not require additional computation. In essence, it assumes that we are solving the pose estimation problem at every frame, and the pose from the previous time step is only used as an initialization (or guess) for where to start the inference at the next frame. As such, this approach is well suited for re-initialization if the pose estimate at the previous timeframe is wrong. The disadvantage is that temporal consistency is only loosely propagated, and the results often exhibit interframe jitter.

In particular, we can define another importance function,

$$q_{ij}^{(4)}(\mathbf{X}_{i,t}) = \mathcal{N}(\mathbf{X}_{i,t} | \mathbf{X}_{i,t-1}, \Lambda_T). \quad (5.20)$$

Sampling from this importance function places the samples in the vicinity of the solution obtained at the previous time step. This is then refined using the observations from the current frame and the message passing. Altering the fraction of samples that come from the different importance functions in the stratified sampling will have an affect on the diversity of poses considered at any given time instant. Ultimately the optimal importance sampling procedure would have to rely on knowledge of the scene and human postures considered. For experiments presented in this chapter, we make no such assumptions and use a simple generic importance sampling scheme discussed previously.

5.7 Experiments and Evaluation

5.7.1 HumanEva-I Dataset

To test performance of our articulated pose estimation and tracking approach we collected the novel dataset¹⁰ that we call HUMANEVA-I. In HUMANEVA-I we simultaneously captured 3D motion and mutioocular video using a calibrated marker-based motion capture system¹¹ and multiple high-speed video cameras. We collected video data using 3 color and 4 greyscale cameras at 60 Hz. The video and motion capture streams were synchronized in software using a direct optimization method. The HUMANEVA-I database consists of 4 subjects performing a set of 6 predefined actions three times (twice with video and motion capture, and once with motion capture alone). The dataset is partitioned into training, validation and testing sub-sets. A more detailed description of the dataset, data collection and processing can be found in [194].

To simultaneously capture video and motion information, our subjects wore natural clothing (as opposed to motion capture suits which are often used for pure motion capture sessions) on which reflective markers were attached using transparent adhesive tape. Our motivation was to obtain natural looking image data that contains all the complexity posed by moving clothing. One negative outcome of this is that the markers tend to move more than they would with a tight-fitting motion capture suit. As result, our ground truth motion capture data may not always be as accurate as that obtained by more traditional methods; we felt that the trade-off of accuracy for realism here was acceptable. We have applied minimal post-processing to the motion capture data, steering away from the use of complex software packages (*e.g.* Motion Builder) that may introduce biases or alter the motion data in the process.

5.7.2 Evaluation Metric

Various evaluation metrics have been proposed for human motion tracking and pose estimation. For example, a number of papers [1, 2, 3, 4, 166, 189, 206] have suggested using joint-angle distance as the error measure. This measure, however, assumes a particular parameterization of the human body and cannot be used to compare methods where the body models have different degrees of freedom or have different parameterizations

¹⁰Dataset is available from <http://vision.cs.brown.edu/humaneva/>.

¹¹We collected motion capture data using a commercial motion capture (MoCap) system from ViconPeak (<http://www.vicon.com/>). The ViconPeak MoCap system is an industry standard for optical marker-based motion capture and has been successfully employed in a variety of entertainment applications for over 10 years. The system uses reflective markers and six 1M-pixel cameras to recover the 3D position of the markers on the body.

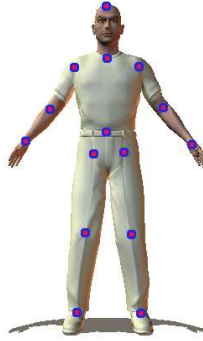


Figure 5.11: **Virtual marker-based evaluation metric.** We define an evaluation metric based on the average distance between a set of 15 virtual markers corresponding to the 3D joint positions and limb ends illustrated in the figure above.

of the joint angles.

We propose an error measure based on a sparse set of virtual markers that correspond to the locations of joints¹² and limb endpoints (see Figure 5.11). This metric is not sensitive to parameterization of the skeletal structure of the body and can easily be derived from most body representations, allowing easy comparison across many approaches. This error metric was first introduced for 3D pose estimation and tracking by us in [197] and later extended in [14]. It has since been also adopted by others for 3D tracking [131] and for 2D pose estimation evaluation in [122, 196].

Assuming that we can represent the pose of the body using $K = 15$ virtual markers, we can write the state of the body as $\mathbf{X}_{mrk} = \{p_1, p_2, \dots, p_K\}$, where $p_k \in \mathbb{R}^3$ is the position of the marker k in the world¹³. Notice, that converting from any standard representation of the body pose to \mathbf{X}_{mrk} is trivial. In particular, to convert from our redundant representation of the body $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ to \mathbf{X}_{mrk} all we need to do, is for every marker (except for the markers corresponding to the limb ends) compute an average of the proximal and distal ends¹⁴ of the two limbs connected at the corresponding joint. For example, computing the virtual marker position corresponding to the left knee joint, $p_{left_knee} = \|H(\mathbf{X}_{left_shin})[0, 0, l_{left_shin}]^T - \mathbf{x}_{left_calf}\|^{1/2}$, where $H(\mathbf{X}_{left_shin})$ is as before a 3D homogeneous object-to-world transformation matrix; l_{left_shin} is the length of the left shin. In other words, $H(\mathbf{X}_{left_shin})[0, 0, l_{left_shin}]^T$ is simply a distal endpoint of the left shin and \mathbf{x}_{left_calf} is the proximal endpoint of the left calf. The error in the overall estimated pose $\hat{\mathbf{X}}_{mrk}$ to the ground truth pose \mathbf{X}_{mrk} can then be expressed as the average absolute distance between individual markers,

$$Error(\mathbf{X}_{mrk}, \hat{\mathbf{X}}_{mrk}) = \sum_{k=0}^K \frac{\|p_k - \hat{p}_k\|}{K}. \quad (5.21)$$

Since the position of virtual markers is defined in the global coordinate frame the error will have a physical

¹²The ground truth location of joints was computed from the motion capture data using the Plug-in Gait software module from ViconPeak (<http://www.vicon.com/>).

¹³Notice that p_k can also be $\in \mathbb{R}^2$ if a 2D body model is used. This is the error measure that will be employed in the next chapter.

¹⁴This assumes that both proximal and distal markers correspond to the joint center. Alternatively, if this is not the case, there will be a constant offset between the proximal and/or distal ends of the limb and the required joint marker. This offset can typically be solved for in a least-squared sense using regression.

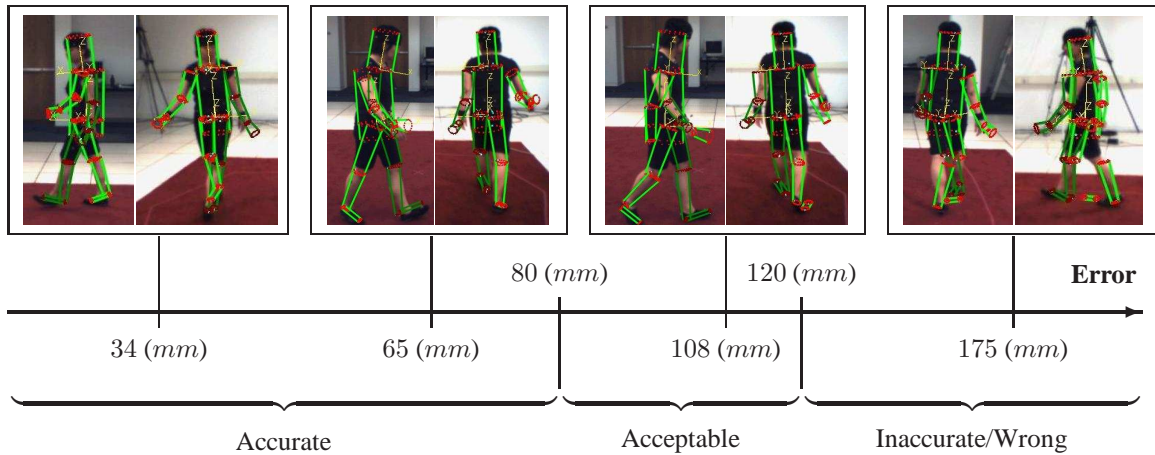


Figure 5.12: **Evaluation metric illustration.** This figure visually illustrates the range of errors corresponding to the virtual marker-based metric introduced. Typically an error of < 80 (mm) corresponds to accurate pose estimation and an error between 80 – 120 (mm) to a reasonable pose. As can be seen from the figure with an error of 108 (mm) the body is vertically shifted down and the arms are less than perfect, but the overall pose is still reasonable. Typically error > 120 (mm) corresponds to wrong or inaccurate poses as illustrated.

meaning and is expressed in (mm). Lower error will correspond to poses that more closely match the ground truth motion capture data. Since the proposed error function averages error over a set of limbs, the exact meaning of the error will depend on the distribution of errors across the body parts at a given frame. The same is true for most other metrics we have found in the literature. In other words, the same quantitative error may be due to either a single joint being off by a considerable amount, or by all the joints being off by small amount. Our approach, due to the nature of the model and inference, tries to distribute error across limbs. In our experience, errors of under 80 (mm) correspond to accurate poses, 80 – 120 (mm) typically correspond to poses of acceptable accuracy, and errors of > 120 (mm) typically correspond to wrong or inaccurate poses. By acceptable accuracy we mean that all parts are recovered, but there may be misalignments at the joints (see Figure 5.12) or slight global vertical shift of the body. To compute performance over a temporal sequence (for tracking), we average the error over all the frames in the sequence and report the mean and standard deviation.

5.7.3 Pose Estimation

Figures 5.13–5.17 show the automatic pose estimation of the 3D body model using bottom-up part detectors. The approach is tested on a total of 198 frames; 128 frames using a 10-part model and 70 frames using a 15-part loose-limbed body model. Note that we use only detectors for the head, and outermost extremities, which for the 10-part model means upper arms and calfs; for the 15-part model hands and feet. These detectors are very noisy and at best can only give a rough position of the part in space. They are unable to differentiate left and right sides of the body, estimate the twist, or even differentiate the direction in which the limb is pointing. For body parts with no associated bottom-up detectors, the initial distributions are assumed to be uniform over the entire state space. After several iterations of belief propagation, the algorithm “finds” the limbs and has a reasonable distribution over the limbs poses. Notice that while we run PAMPAS for 10 message passing iterations to ensure convergence, the solution often settles after 5–6 iterations.

Subject	S1	S2	S1	S2	S1
Action	Walking	Walking	Jog	Walking	Jog
Frames	57	40	31	39	31
Model	10-part	10-part	10-part	15-part	15-part
Mean Error (mm)	89.7	161.6	83.7	158.5	130.5
Standard deviation of Error (mm)	71.5	103.6	51.3	132.2	87.4
% of frames with Error < 80 (mm)	70.2	27.5	61.3	46.2	45.2
% of frames with Error < 120 (mm)	82.5	50.0	87.1	61.5	61.3
% of frames with Error \geq 120 (mm)	17.5	50.0	12.9	38.5	38.7

Table 5.2: **Summary of pose estimation performance using loose-limbed body model.** More detailed results can be found in Figures 5.13–5.17.

We have tested our approach on sequences from two subjects performing two different motions (walking and jogging). In all experiments we used same values for all parameters in our system and same likelihoods. In all experiments reported here, we used 7 camera views for inference (3 color and 4 greyscale). We have also experimented with pose estimation and tracking using 4 and 3 views with similar results. The challenge with the HumanEva-I dataset used here is that, due to specular highlights and poor greyscale imagery, simple background subtraction employed by our system often produces poor segmentation of the foreground (see the right column corresponding to camera BW4 in Figure 5.8) that has to be dealt with. In particular, standard voxel-based methods that require good background subtraction, would typically not be able to cope with such noisy data. Our approach deals gracefully with this, producing accurate results automatically from the single multiocular image. In most cases the recovered joint positions are < 80 (mm) away from the true joint positions. The summary of performance is presented in Table 5.2.

Perhaps most revealing, regarding the accuracy of the method, is the bar plot presented in bottom of each of the Figures 5.13–5.17. The plot shows the histogram of errors for all tested frames (selected uniformly and without bias for each sequence). In most frames tested, the error falls below the 120 (mm) level (see Figures 5.13, 5.14, 5.15, and 5.17) that we consider to be “acceptable”. The worst performance was observed for sequence in Figure 5.16, where we are able to “correctly” estimate the pose (below 120 (mm)) in about 50% of the frames. However, even at this error rate we believe that our pose estimation approach is less restrictive than related approaches that attempt to estimate the pose either in a particular canonical pose class (e.g. stylized 2D scissor-leg walking stance [169]) or by having a cooperative subject [36, 112, 113].

5.7.4 Tracking

In this section we test the performance of our approach in the context of tracking, where we assume that a sequence of multiocular frames is available for inference. The loose temporal consistency is used to propagate results from one frame to the next (see description in Section 5.6.1) to help focus the inference. In this paradigm we assume that limbs at the next frame are sufficiently close to the correctly estimated pose at the previous frame. Hence, having a proposal distribution that focuses a fraction of samples in locations where the limbs were previously found proves useful. Since typically the previous frame estimates are sufficiently close to the solution at the current frame, we only run PAMPAS for 2 message passing iterations (instead of 10) as a way of speeding up the inference. The results are shown in Figures 5.18–5.21. The approach is tested on a total of 1205 frames; 400 frames using a 10-part model and 805 frames using 15-part loose-limbed body

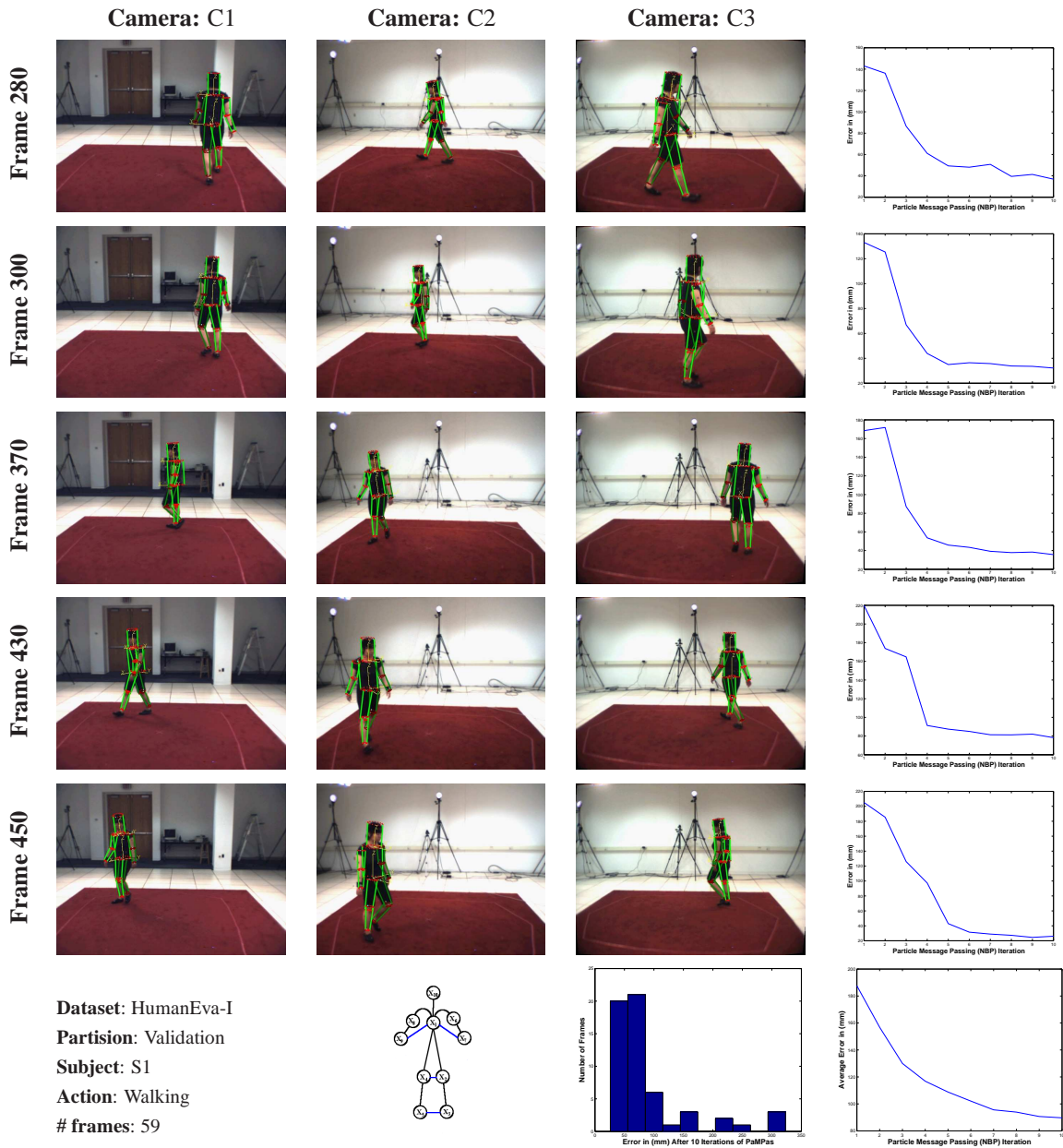


Figure 5.13: **Pose estimation using 10-part loose-limbed body model.** Results of pose estimation from a single multiocular frame are shown for a number of frames from HumanEva-I dataset. Top five rows show the final result in terms of most likely sample from the marginal for each part after 10 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The right column of the first five rows shows the error as a function of message passing iterations for respective frames. Notice that typically the error decreases sharply for the first 4–5 iterations and then stays relatively low with minor variations that are due to sampling. The last row illustrates performance over all (59) frames tested for the sequence (every 10-th frame was selected). As can be seen from bar plot, the pose was estimated in most frames with low error. The error as a function of message passing iterations averaged over all frames is shown in the bottom right corner of the figure.

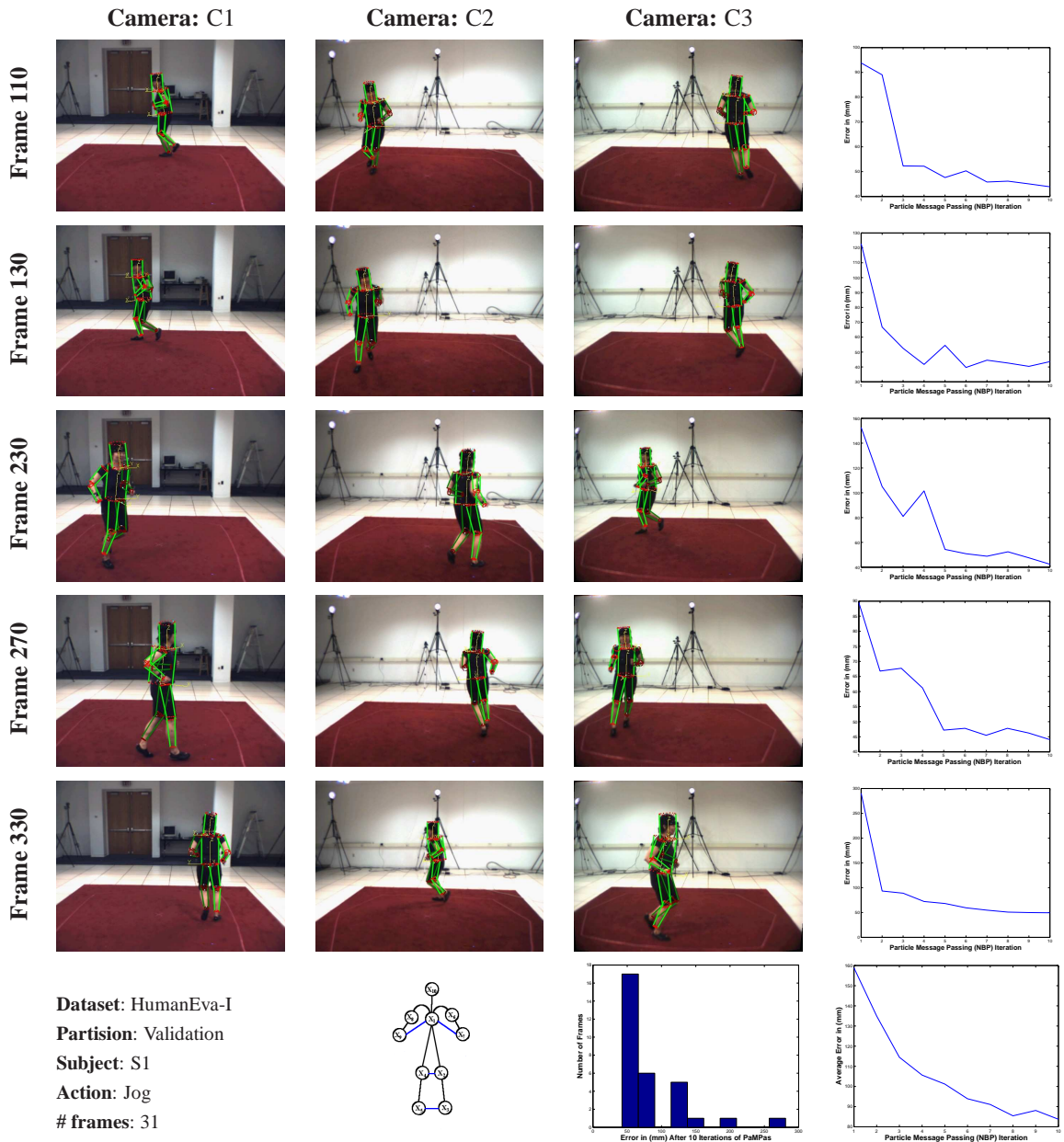


Figure 5.14: **Pose estimation using 10-part loose-limbed body model.** Results of pose estimation from a single multiocular frame are shown for a number of frames from HumanEva-I dataset. Top five rows show the final result in terms of most likely sample from the marginal for each part after 10 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The right column of the first five rows shows the error as a function of message passing iterations for respective frames. Notice that typically the error decreases sharply for the first 4–5 iterations and then stays relatively low with minor variations that are due to sampling. The last row illustrates performance over all (31) frames tested for the sequence (every 10-th frame was selected). As can be seen from bar plot, the pose was estimated in the pose with low error. The error as a function of message passing iterations averaged over all frames is shown in the bottom right corner of the figure.

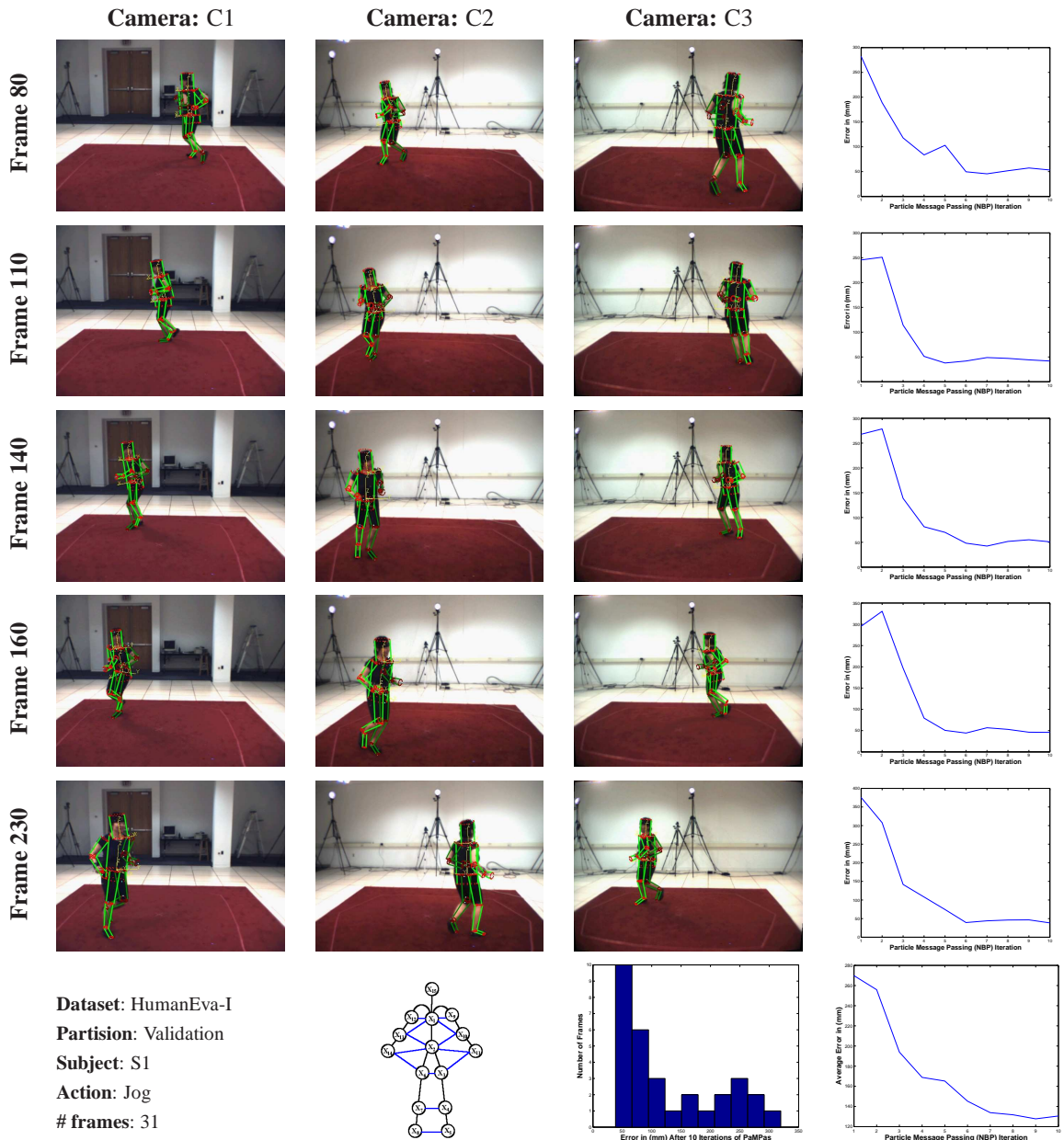


Figure 5.15: **Pose estimation using 15-part loose-limbed body model.** Results of pose estimation from a single multiocular frame are shown for a number of frames from HumanEva-I dataset. Top five rows show the final result in terms of most likely sample from the marginal for each part after 10 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The right column of the first five rows shows the error as a function of message passing iterations for respective frames. Notice that typically the error decreases sharply for the first 4–5 iterations and then stays relatively low with minor variations that are due to sampling. The last row illustrates performance over all (31) frames tested for the sequence (every 10-th frame was selected). As can be seen from bar plot, the pose was estimated in most frames with low error. The error as a function of message passing iterations averaged over all frames is shown in the bottom right corner of the figure.

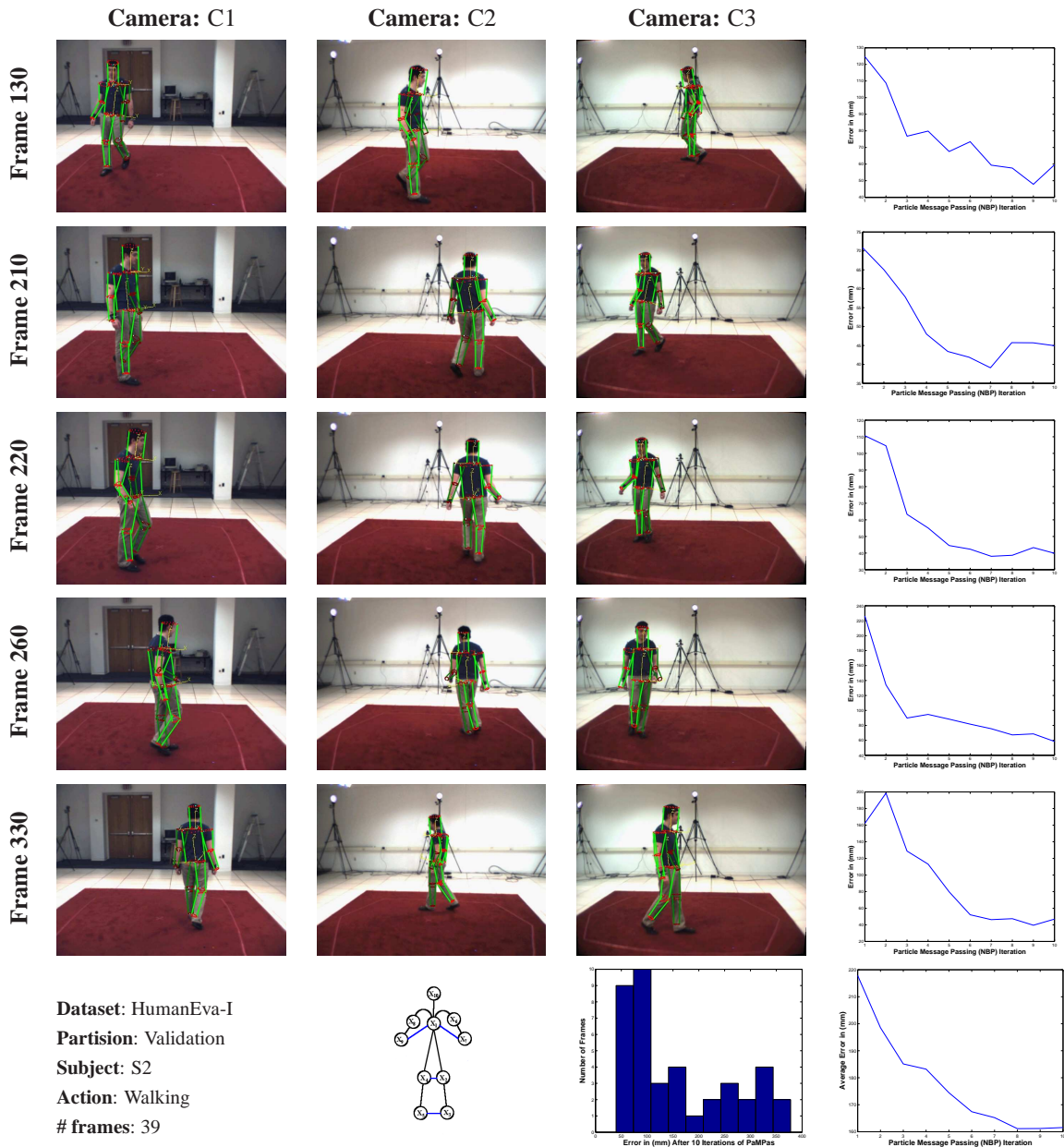


Figure 5.16: **Pose estimation using 10-part loose-limbed body model.** Results of pose estimation from a single multiocular frame are shown for a number of frames from HumanEva-I dataset. Top five rows show the final result in terms of most likely sample from the marginal for each part after 10 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The right column of the first five rows shows the error as a function of message passing iterations for respective frames. Notice that typically the error decreases sharply for the first 4–5 iterations and then stays relatively low with minor variations that are due to sampling. The last row illustrates performance over all (39) frames tested for the sequence (every 10-th frame was selected). As can be seen from bar plot, the pose was estimated in most frames with low error. The error as a function of message passing iterations averaged over all frames is shown in the bottom right corner of the figure.

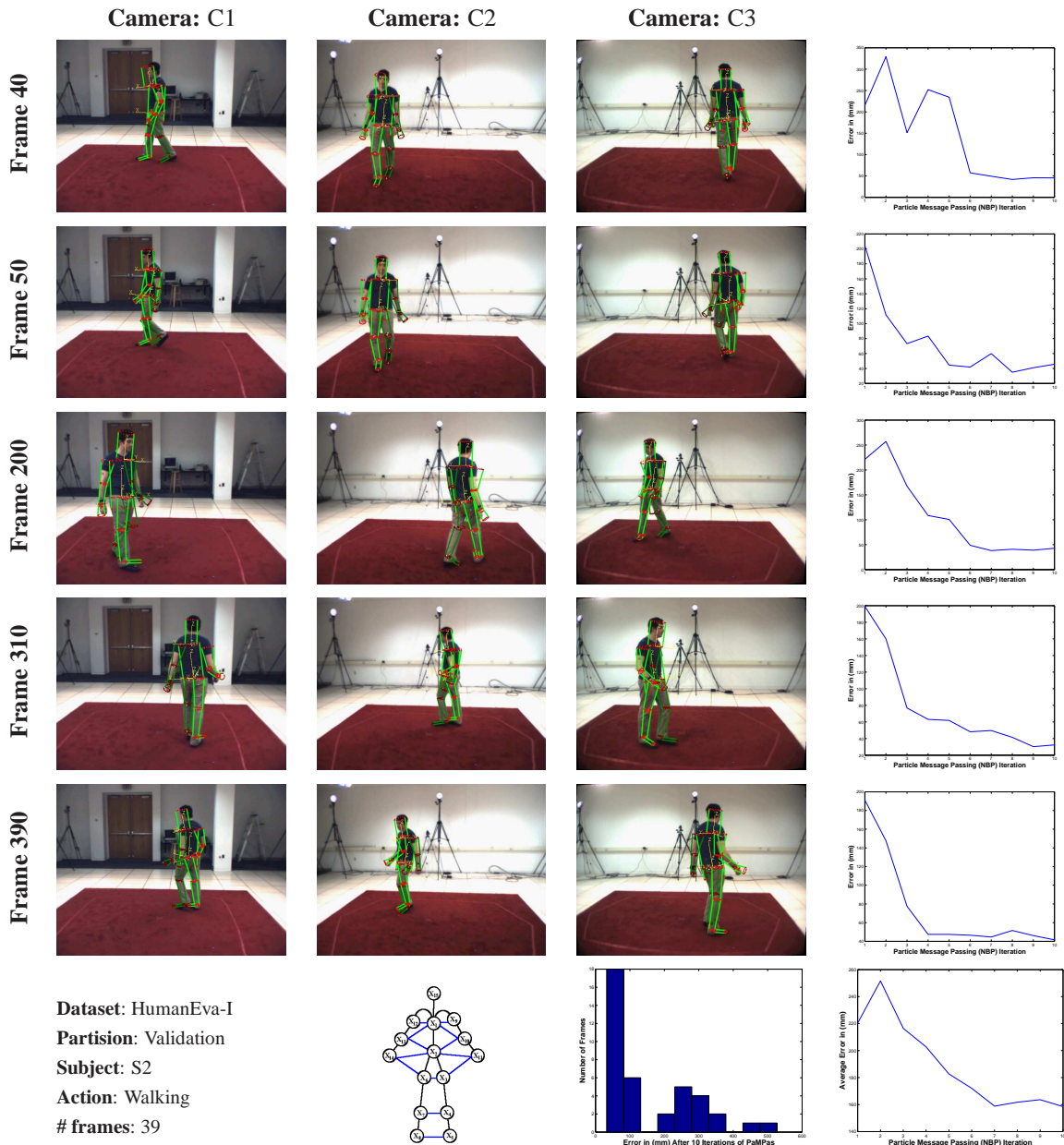


Figure 5.17: **Pose estimation using 15-part loose-limbed body model.** Results of pose estimation from a single multiocular frame are shown for a number of frames from HumanEva-I dataset. Top five rows show the final result in terms of most likely sample from the marginal for each part after 10 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The right column of the first five rows shows the error as a function of message passing iterations for respective frames. Notice that typically the error decreases sharply for the first 4–5 iterations and then stays relatively low with minor variations that are due to sampling. The last row illustrates performance over all (39) frames tested for the sequence (every 10-th frame was selected). As can be seen from bar plot, the pose was estimated in most frames with low error. The error as a function of message passing iterations averaged over all frames is shown in the bottom right corner of the figure.

Subject	S2	S1	S1	S2
Action	Walking	Walking	Jog	Walking
Frames	400	391	201	213
Model	10-part	15-part	15-part	15-part
Mean Error (<i>mm</i>)	74	59	77	69
Standard deviation of Error (<i>mm</i>)	9.95	25.2	20.2	18.8
Average for the model (<i>mm</i>)	74	66		
Standard deviation for the model (<i>mm</i>)	9.95	23.5		

Table 5.3: **Summary of tracking performance using loose-limbed body model.** More detailed results can be found in Figures 5.18–5.21.

model. The average performance over the sequence ranges between 59–77 (*mm*) in all cases (see summary of results in Table 5.3). Also, notice that the approach quickly recovers when infrequent miss-tracking occurs (see Frame 78 in Figure 5.18).

5.7.5 Comparison with Annealed Particle Filter

In previous section we explored performance of the loose-limbed body model in the context of tracking. In this section, we compare the results obtained by our approach to a relatively standard tracking algorithm, Annealed Particle Filter (APF) (see Section 2.8 for more detailed description). In particular, we make use of the APF algorithm implemented¹⁵ and tested by Balan *et al.* in [14]. In our comparison, Annealed Particle Filter performs inference over kinematic tree body model with 15 parts, comparable to our 15-part loose-limbed body model; the resulting state-space parameterization of the pose is $\in \mathbb{R}^{40}$, corresponding to global position and orientation of the torso in 3D and 36 joint angles. Consequently, the implementation of APF we employ is also using comparable likelihood function that incorporates silhouette and edge information (see [14] for details). Unlike the original APF algorithm proposed by Deutscher *et al.* [52], the variant of [14] is also able to incorporate the temporal and structural priors, that ensure that parts do not penetrate each other and that joints are within the allowable limits. In Figure 5.22 we compare our model with three variants of APF algorithm: generic APF with interpenetration constraints and very generic joint limits with **(i)** 250 and **(ii)** 500 particles, and **(iii)** an APF algorithm that in addition encodes action-specific joint limits and temporal prior. In all cases Annealed Particle Filter requires an initial pose at the first frame to bootstrap the inference; this was obtained from ground truth motion capture data.

Loose-limbed body model in both sequences outperforms the generic APF algorithms (consequently, the number of particles seems to play little significance in the overall performance of APF) and performs comparably to the action-specific APF variant (see Figure 5.22). In all cases, however, the variance for the estimates obtained using APF are lower than those obtained using our loose-limbed body model. This is not surprising, considering the nature of inference employed in the loose-limbed body model, where the pose at the previous time instant is simply a proposal for inference at the next time frame. While this type of inference is beneficial in that it allows easy recovery from intermittent failures, the pose estimation that is inherently incorporated at every frame also tends to produce noisier results when such failures are not present.

¹⁵Implementation of APF is courtesy of Alexandru Balan and is freely distributed from <http://www.cs.brown.edu/~alb/software.htm>.

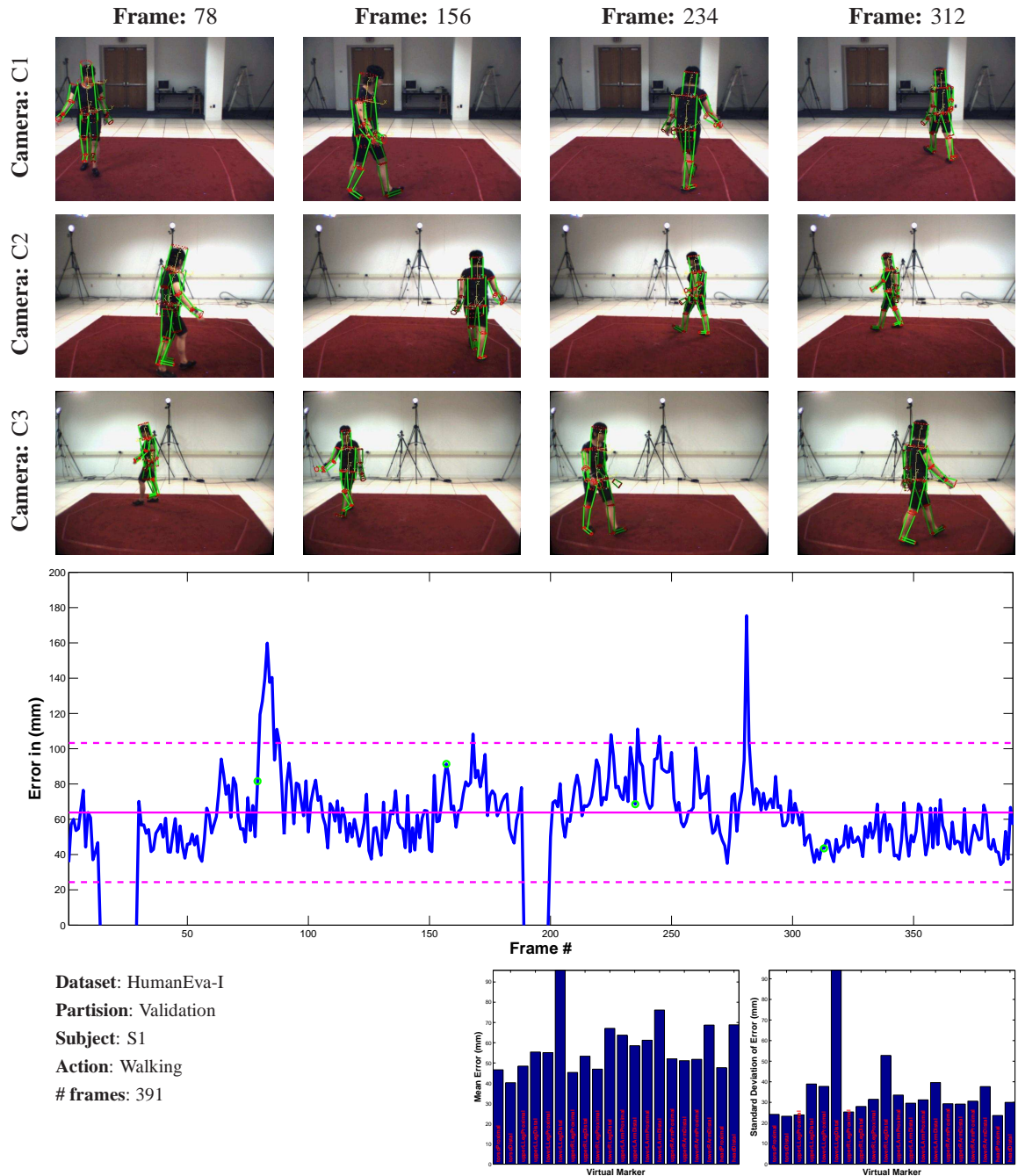


Figure 5.18: **Tracking using 15-part loose-limbed body model.** Results of tracking in a multiocular sequence from the HumanEva-I dataset are shown for a number of frames. Top three rows show the final result in terms of most likely sample from the marginal for each part after 2 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The figure in the second to last row shows per frame error (in blue) for all 390 frames used for testing. The mean error computed over the entire sequence and $\pm 2\sigma$ are shown in solid and dashed magenta respectively. Frames selected automatically and temporally equidistantly to visually illustrate performance (top three rows), are designated by green circles on the graph. The last row illustrates an alternative analysis of error by showing statistics for individual virtual markers, with mean on the left and standard deviation on the right, averaged over the entire sequence. Notice that the approach successfully recovers from the miss-tracking that is starting to present itself at frame 78. We believe that the source of the tracking failure here is caused by the ambiguity in the image evidence; self-occlusions of the body under the particular placement of the cameras with respect to the observed pose make it hard to disambiguate true and recovered pose based on image evidence alone.

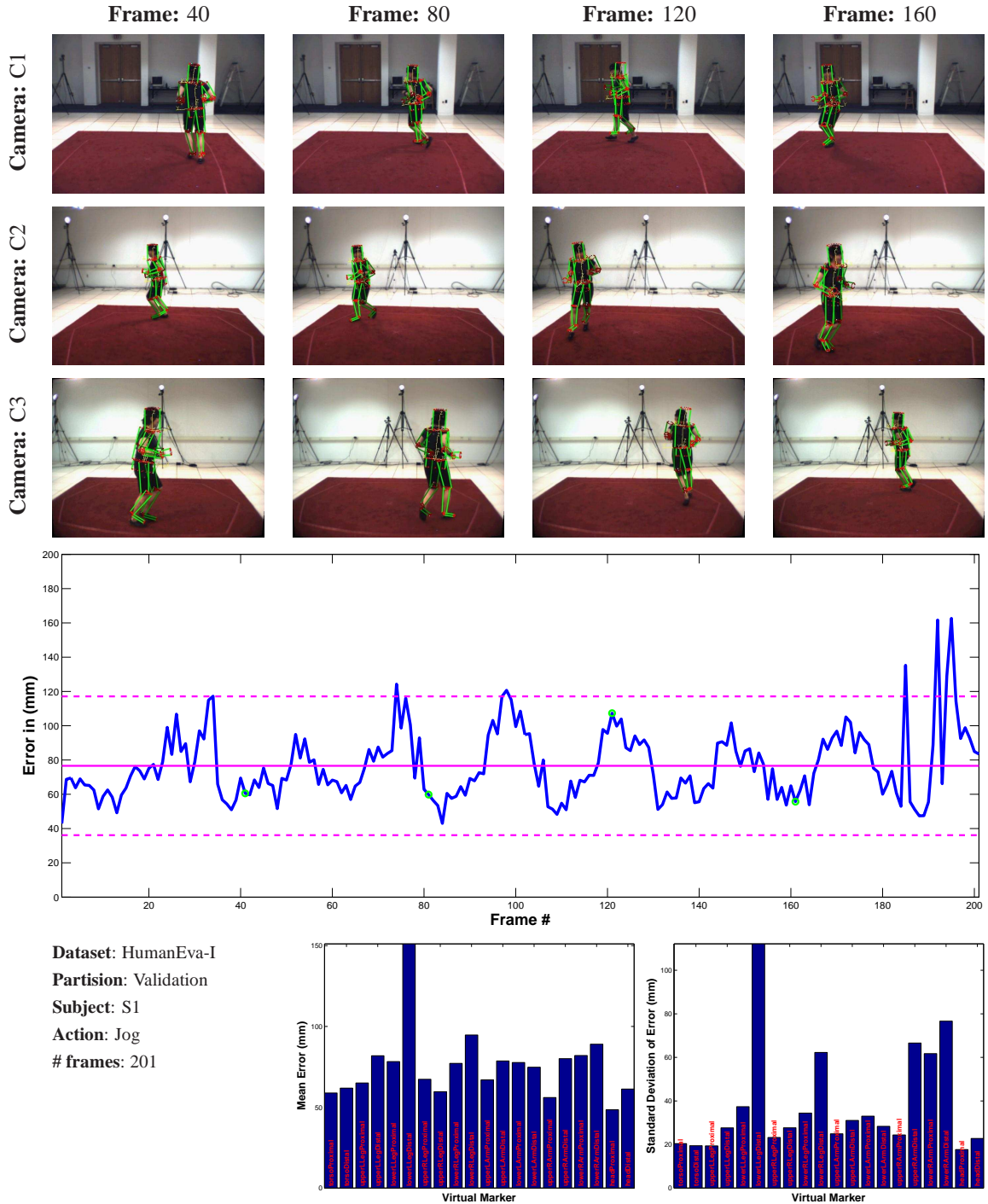


Figure 5.19: **Tracking using 15-part loose-limbed body model.** Results of tracking in a multiocular sequence from the HumanEva-I dataset are shown for a number of frames. Top three rows show the final result in terms of most likely sample from the marginal for each part after 2 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The figure in the second to last row shows per frame error (in blue) for all 201 frames used for testing. The mean error computed over the entire sequence and $\pm 2\sigma$ are shown in solid and dashed magenta respectively. Frames selected automatically and temporally equidistantly to visually illustrate performance (top three rows), are designated by green circles on the graph. The last row illustrates an alternative analysis of error by showing statistics for individual virtual markers, with mean on the left and standard deviation on the right, averaged over the entire sequence.

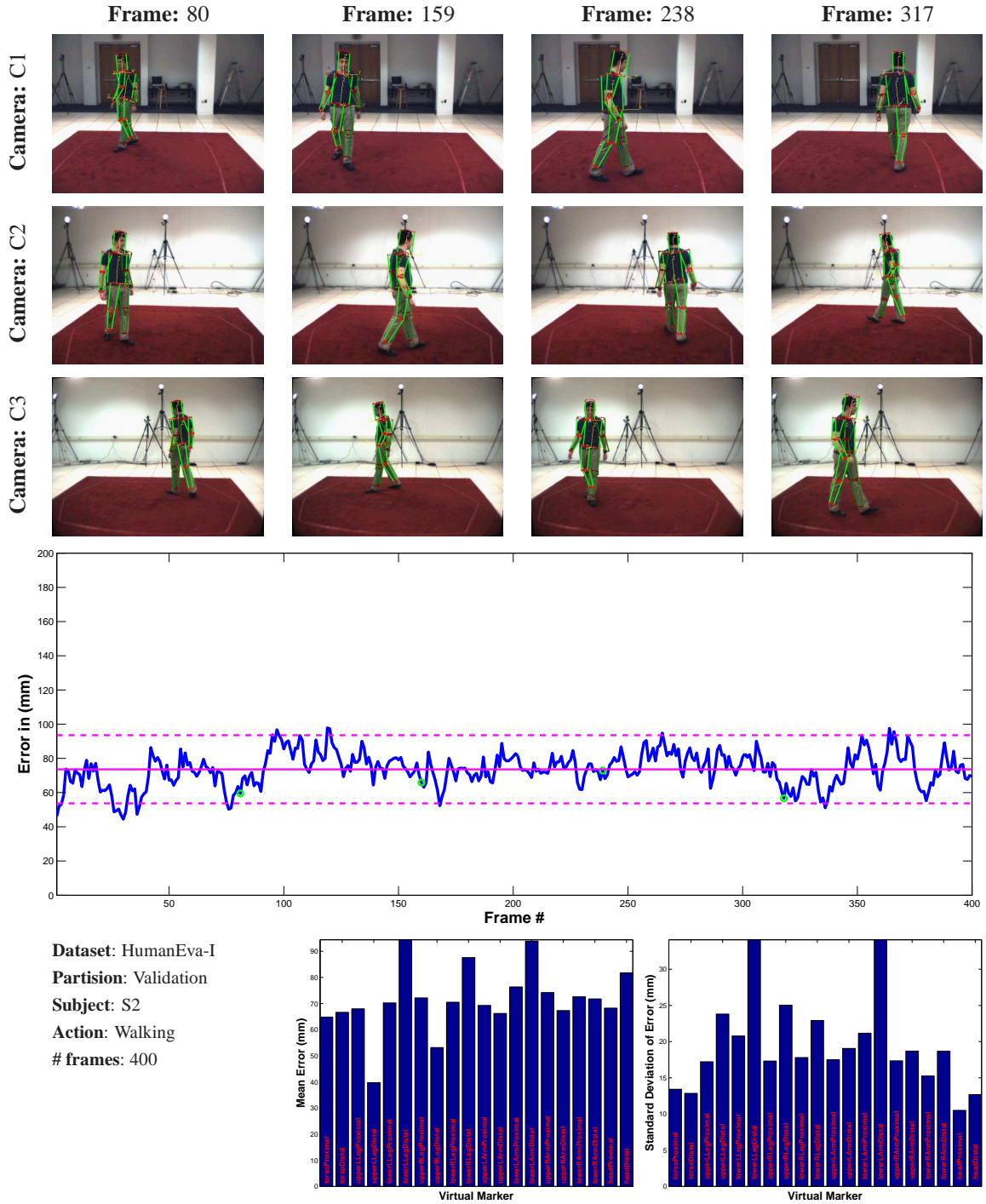


Figure 5.20: **Tracking using 10-part loose-limbed body model.** Results of tracking in a multiocular sequence from the HumanEva-I dataset are shown for a number of frames. Top three rows show the final result in terms of most likely sample from the marginal for each part after 2 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The figure in the second to last row shows per frame error (in blue) for all 400 frames used for testing. The mean error computed over the entire sequence and $\pm 2\sigma$ are shown in solid and dashed magenta respectively. Frames selected automatically and temporally equidistantly to visually illustrate performance (top three rows), are designated by green circles on the graph. The last row illustrates an alternative analysis of error by showing statistics for individual virtual markers, with mean on the left and standard deviation on the right, averaged over the entire sequence.

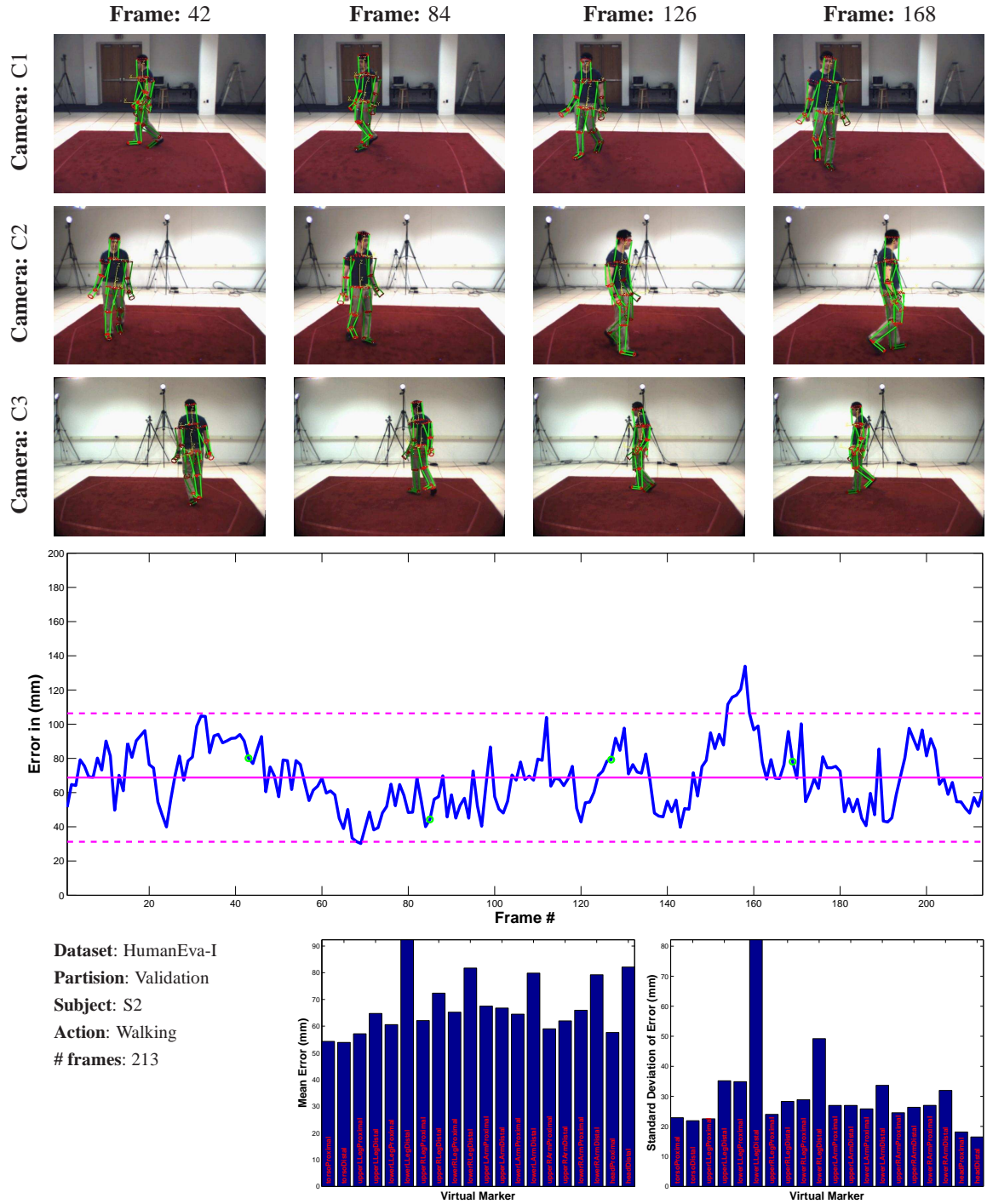


Figure 5.21: **Tracking using 15-part loose-limbed body model.** Results of tracking in a multiocular sequence from the HumanEva-I dataset are shown for a number of frames. Top three rows show the final result in terms of most likely sample from the marginal for each part after 2 iterations of PAMPAS. The results are projected into 3 synchronized views for clarity (7 views were used for inference). The figure in the second to last row shows per frame error (in blue) for all 213 frames used for testing. The mean error computed over the entire sequence and $\pm 2\sigma$ are shown in solid and dashed magenta respectively. Frames selected automatically and temporally equidistantly to visually illustrate performance (top three rows), are designated by green circles on the graph. The last row illustrates an alternative analysis of error by showing statistics for individual virtual markers, with mean on the left and standard deviation on the right, averaged over the entire sequence.

In the APF algorithm, on the contrary, the strong dependence on the estimates from previous frame smooths the posterior at the expense of persistent failures (*i.e.* when failure occurs it usually persists for many, if not all, frames). More importantly, our algorithm is fully automatic and is able to estimate the pose at the first frame as well as track it over time; the APF approach was specifically developed for tracking, consequently it requires manual initialization.

5.7.6 Analysis of Failures

In the context of pose estimation, while our approach performs reasonably well in most frames, it does occasionally suffer from failures. In this section we would like to analyze the common failure modes (see Figure 5.23).

Intuitively, our approach iteratively estimates the plausible domain for the position and orientation of limbs and the distribution over that domain. Part detectors are critical in providing the initial guess to the plausible portion of the state space (domain) that should be considered. However, part detectors, are not always precise and hence the algorithm can become trapped in local optima. In particular, since the left and right limbs are indistinguishable, the only detector that gives clues as to overall orientation (view) of the body is the head detector. In the absence of reliable head estimates (a common scenario in practice due to the poor image quality and sparse placement of cameras), the model suffers from a 180 degree ambiguity. This ambiguity, that is illustrated in Figure 5.23 (**top**), can be resolved to some extent by the articulation of the body itself. Joints that have asymmetric degrees of freedom (*i.e.* hard stops), modeled in our case by kinematic constraints, can help to resolve this ambiguity in some cases. In other cases, however, where articulation is minimal, they do not provide reliable distinguishing power (see Figure 5.23 (**top**)). Intuitively, the 15-part body model should help in these cases, because feet provide additional constraints on the overall orientation of the body. Unfortunately, floor shadows make it challenging to find feet reliably. Hence, we have observed limited performance benefit from this more refined model.

It is also worth mentioning that since we work with loopy graphical models, in general our method is not guaranteed to converge and in the case of convergence is only guaranteed to converge to a local optimum. If the model does not converge, which in our experience happens infrequently, it can oscillate between solutions as illustrated in Figure 5.23 (**bottom**).

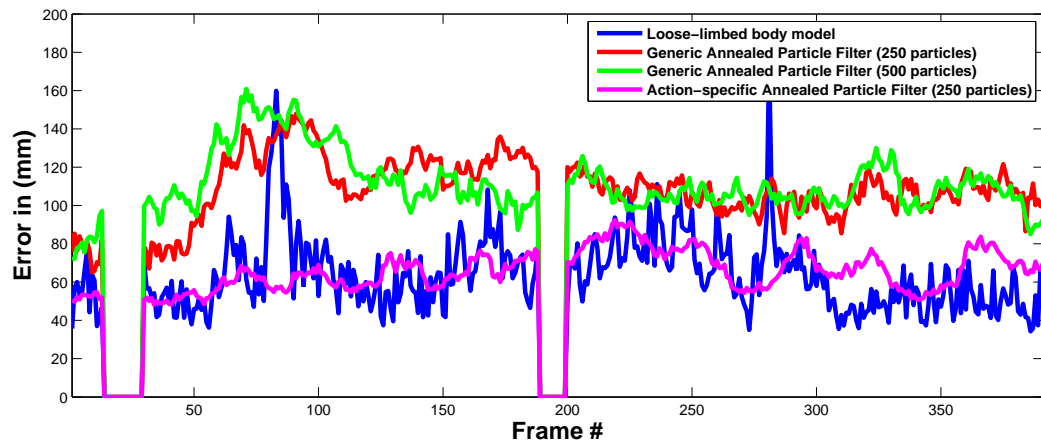
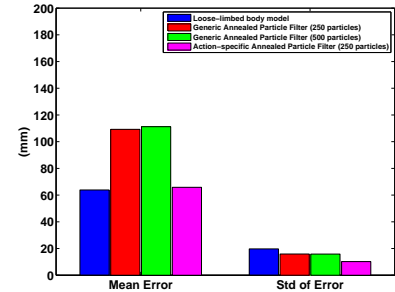
5.7.7 Discussion of Quantitative Performance

It may be surprising that for the frames where our algorithm produces visually pleasing results (see experiments in previous sections) the error is still in the range of 30–40 (*mm*). This is in part due to the stringent error measure criterion employed in this thesis and in part to some error being present in the ground truth data itself. In general, the visualization may be a bit misleading unless one zooms and closely looks at individual body parts and joint locations. In particular, so long as the model overlaps mostly with the body, things tend to look good (even though individual joints may be off). Consequently, this is why we believe that a well established quantitative metric, such as the one introduced here, is needed to drive the future research in pose estimation and tracking.

In many cases where the error is 40 (*mm*) or lower the pose obtained by our approach provides a very good interpretation of the image, however, there exists little misalignment at the joints (consequently, since in most camera views pixel corresponds to about 5 to 8 *mm*, the joints only need to be off by 5 to 8 *pixels* to

Dataset: HumanEva-I
Partition: Validation
Subject: S1
Action: Walking
frames: 391

	Error	
	Mean (mm)	Std (mm)
Loose-Limbed Model	63.8	19.7
Generic APF (250)	109.2	15.9
Generic APF (500)	111.2	15.8
Action-specific APF (250)	65.8	10.2



Dataset: HumanEva-I
Partition: Validation
Subject: S2
Action: Walking
frames: 213

	Error	
	Mean (mm)	Std (mm)
Loose-Limbed Model	68.8	18.8
Generic APF (250)	86.7	14.8
Generic APF (500)	81.2	12.2
Action-specific APF (250)	70.0	7.3

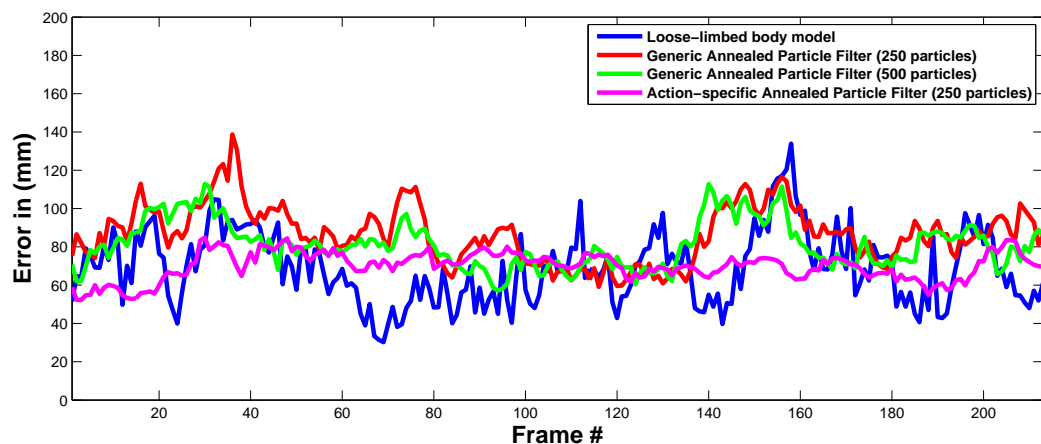
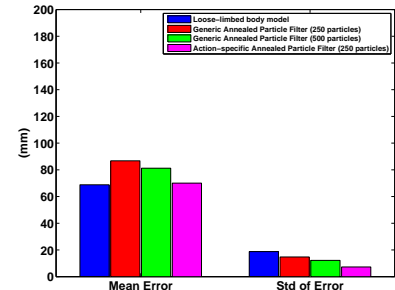


Figure 5.22: Comparison with annealed particle filter. Tracking results produced by our loose-limbed body model and by Annealed Particle Filter (APF) are illustrated and compared on two sequences, illustrated in Figures 5.18 and 5.21. Three variants of APF are implemented for comparison (see text for details). All methods use comparable 15-part body models and likelihood functions; for APF this results in kinematic tree model with 40 parameters. Top row, in each case, denotes the sequence used (left) and the statistics for performance of various methods, averaged over the length of the entire sequence, in both table (middle) and bar plot (right) form. Bottom plot, in each case, illustrates performance for the entire sequence.

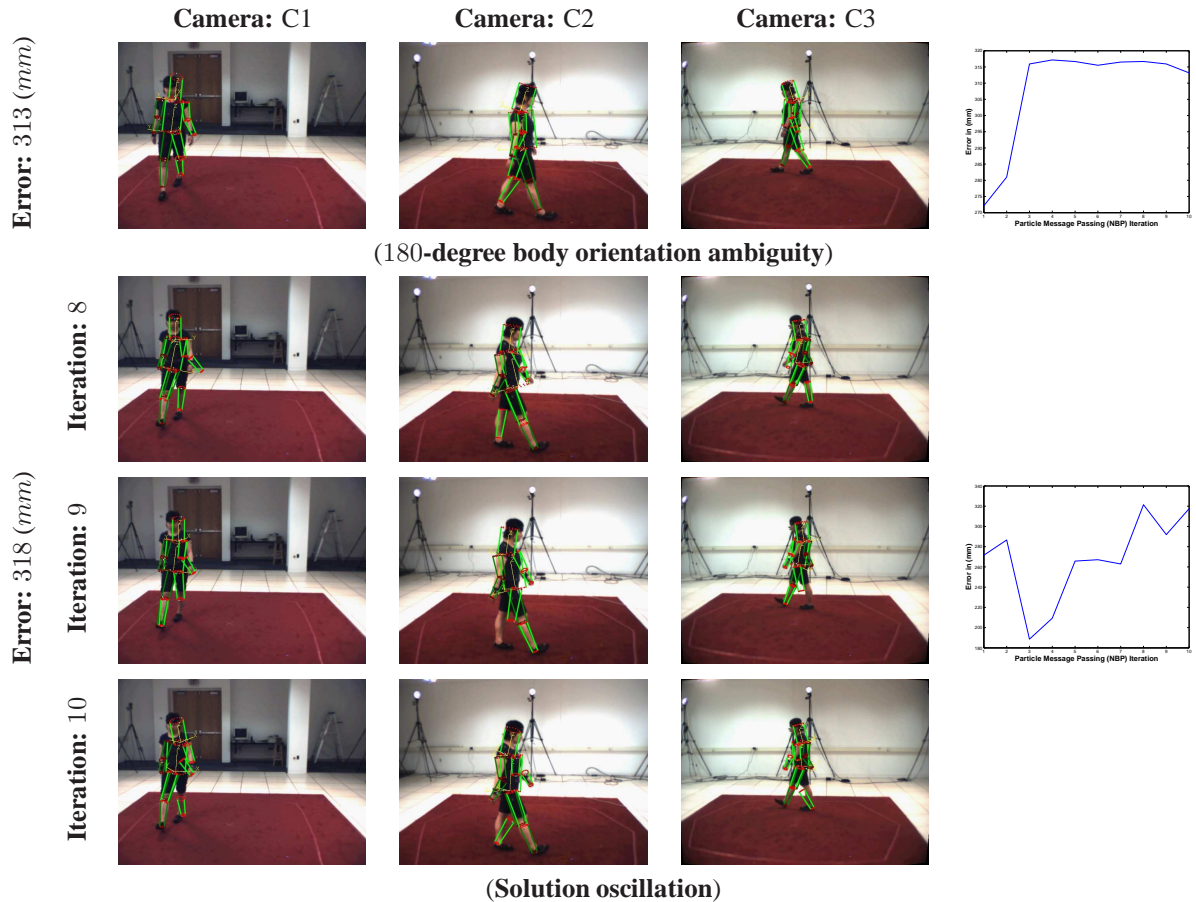


Figure 5.23: **Failure modes.** One of the most common failure modes of our approach is due to the rotational symmetry of the body. Since the only detector that is sensitive to the overall orientation of the body is the head, in the absence of reliable head detection (a common scenario in practice), the overall pose of the body can potentially be recovered pointing in the opposite direction (**top**). In the figure, dark limbs correspond to the left side of the model. This is particularly common in the scenarios where articulations, that also provide hints as to the overall orientation of the body, are minimal. Notice that the plot on the right, that illustrates the error as a function of message passing iterations, clearly shows that BP has converged, but in this case to a wrong solution (which consequently is the local maximum of the joint probability function). Sometimes, however, lack of correct orientation (or lack of a good match to the image data in general) may lead to oscillations between solutions in the inference (**bottom**). In particular, notice how the legs assume similar configuration at iteration 8 and 10 and a competing configuration at iteration 9. This is a problem known in the general loopy graphical model literature.

produce an error of this magnitude). The ground truth motion capture data is also not perfect, which results in additional error overhead. There are a number of confounding artifacts that may explain why the motion capture data may not result in perfect ground truth.

First, the recovered ground truth joints are not exact by definition. There seems to be large variety of opinions, from the biomechanics¹⁶ perspective, as to how accurately the Vicon system can recover joint positions. In particular, the Vicon software that we are using to extract joints, is developed for Gait analysis

¹⁶I would like to thank Lars Mundermann and Stefano Corazza from Stanford's BioMotion Laboratory for relevant and very insightful discussions.

	10-part model	15-part model
Part Detectors	47 sec	45 sec
Message Passing	20 sec / iteration	84 sec / iteration
Belief Estimation	10 sec	64 sec
Total		
Pose Estimation	259 sec	948 sec
Tracking	99 sec	274 sec

Table 5.4: **Runtime speed of inference.** All numbers are reported per frame unless otherwise stated. These results were measured on a single processor 2.0 GHz machine with 1 GB of RAM.

and hence is naturally less accurate for more complex motions. In addition, and for the same reason, the ground truth joint positions recovered for the upper body tend to be less accurate than lower body joints. Consequently, since we put markers on regular clothes, instead of body directly (or body suites) our motion capture data is inherently less accurate than standard methods. It’s hard to quantify to what extent these artifacts affect our error computations. We would argue, however, that these artifacts are minor with respect to the errors produced by the algorithm in most cases.

Secondly, since calibration and synchronization of video with motion capture data was done in software, there is some error present due to the calibration. We conducted a set of simple experiments, where we manually clicked on unique fixed point in the scene in all views, reconstructed the 3D point corresponding to the intersection of all rays, then projected it back and looked at the difference between the clicked points and re-projections. The difference, when converted (under appropriate scaling) into 3D, was in the range of 6–20 (*mm*) depending on where the point was in the scene. Of course, this experiment is biased by the manual clicking involved (it is hard to click on points in the scene precisely). Furthermore, it is unclear how the observed re-projection error relates to the joint error measure computed by our approach. Nevertheless, this suggests that there is some non zero contribution to the observed error due to the calibration.

Even with these problems, we believe that measuring the error in ways advocated by this thesis is meaningful. Particularly so, if one is interested in the relative and not absolute measure of performance. Lastly, it is worth mentioning that independent studies¹⁷ on HUMANEVA-I dataset have all reported error > 30 (*mm*) (typically in the 100 (*mm*) range).

5.7.8 Analysis of Runtime Speed

Currently we have implementations of the *loose-limbed body model* in both Matlab and C++. All experiments in this chapter were done using the C++ version. While significantly faster, then our Matlab implementation, the C++ version is still relatively slow and does not allow for real-time inference. The overall performance for a typical run of each one of the two models and modes of operation is illustrated in Table 5.4.

Part detectors present a fixed overhead for each frame, that roughly amounts to 45-50 seconds for 7 views. Notice that since part detectors operate on pairs of views, their runtime in general scales exponentially with the number of views available. The rest of the time spent in PAMPAS, consists of a number of message passing iterations and a single belief estimation stage at the end. The majority of time in both stages is spent drawing samples from the product of messages (represented by Gaussian mixtures).

¹⁷Results can be found at <http://vision.cs.brown.edu/humaneva/publications.html>.

The model presented here, however, has a great potential for parallelization. Of particular interest is the parallel implementation on the new multicore architectures, now becoming common.

5.8 Conclusion and Discussion

This chapter has presented a probabilistic method for multiocular, fully automatic, 3D human pose estimation and tracking. We show that a *loose-limbed body model* with continuous-valued parameters can effectively represent a person’s location and pose, and that inference over such a model can be tractably performed using non-parametric belief propagation. The belief propagation framework allows us to avoid distinguishing between pose estimation and tracking, but instead to use bottom-up part detectors to stabilize the motion estimation and provide “initialization” cues at every time-step.

The main advantages of our approach are: the complexity of the search task is linear rather than exponential in the number of body parts; bottom-up processes are integrated at every frame allowing automatic initialization and recovery from transient tracking failures; the conditional probabilities between limbs in space and time are learned from training data. Additionally, we exploit a novel data set with synchronized 3D “ground truth” and video data for quantitative evaluation of performance.

CHAPTER 6

Hierarchical Approach for Monocular 3D Pose-Estimation and Tracking

The estimation of 3D human motion (especially in the context of tracking where initial pose is known) is relatively well understood in controlled laboratory settings with multiple cameras where a number of Bayesian inference methods can recover 3D human motion (*e.g.* [14, 49, 210]). In the previous chapter we addressed this problem using a novel *loose-limbed body model* that facilitates automatic pose estimation in addition to tracking. In this chapter we will address the more general problem of articulated 3D pose estimation and tracking from monocular static images and video.

Most prior methods for 3D human motion estimation, rely on accurate background subtraction and edge information; this is a strong limitation that prevents their use in more realistic and complex environments. When the background is changing or the camera is moving, reliable background subtraction is difficult to achieve. The problems become particularly acute in the case of monocular tracking where the mapping from 2D image features to a 3D body model is highly ambiguous. Consequently, solutions to the monocular (static camera) case have so far relied on strong prior models [193], manual initialization [209] and/or accurate silhouettes [3, 4, 189, 209]. The fully automatic case involving a monocular camera is the focus of this chapter.

Recent work on 2D body pose estimation and tracking treats the body as a “cardboard person” [111] in which the limbs are represented by 2D planar (or affine) patches connected by joints (see Section 2.4.2 and 2.4.3). Such models are lower-dimensional than the full 3D model and recent work has shown that they can be estimated from monocular 2D images [59, 170, 173]. The results are typically noisy and imprecise but they provide exactly the kind of information necessary to generate *proposals* for the probabilistic inference of 3D human pose. Thus we simplify the 3D inference problem by introducing an intermediate 2D estimation stage.

While there has been recent work on directly inferring 3D pose from low-level 2D features, these methods typically rely on accurate background subtraction information [3, 4, 189] which may be difficult to obtain outside the controlled laboratory setting. Consider, for example, the image in Figure 6.2 (a). A key observation here is that 2D body models can substitute for silhouettes in these 2D to 3D discriminative inference methods (see Figure 6.1) and, moreover, provide a richer representation than silhouettes in that the 2D models represent joint angles and structures internal to standard silhouettes. This richer model reduces the ambiguity



Figure 6.1: **Typical discriminative inference process.** Discriminative approaches [2, 3, 4, 189, 206] attempt to estimate the 3D pose directly from the image features, as illustrated above.

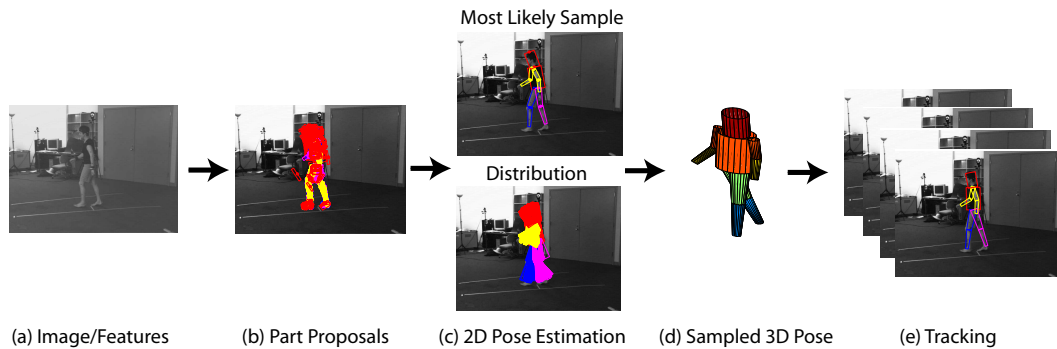


Figure 6.2: **Example of the hierarchical inference process.** (a) monocular input image with bottom up limb proposals overlaid (b); (c) distribution over 2D limb poses computed using non-parametric belief propagation; (d) sample of a 3D body pose generated from the 2D pose; (e) illustration of tracking.

in the 2D to 3D mapping.

Recent approaches to 2D articulated human body detection and pose estimation exploit part-based tree-structured models [59, 93, 122, 170, 174, 178] that capture kinematic relations between body parts. In such models a body part is represented as a node in a graph and edges between nodes represent the kinematic constraints between connected parts. These models are attractive because they allow local estimates of limb pose to be combined into globally consistent body poses. While these distributed models admit efficient inference methods, that scale in time linearly proportional to the number of body parts, the local nature of the inference itself is also the Achilles heal of these methods. The image evidence for each part is estimated independently of the other parts and, without a global measure of the image likelihood of a body pose, multiple body parts can, and often do, explain the same image data.

In particular, for 2D body pose estimation, the “wrong” solutions are often more likely than the “true” solution. Figure 6.4 illustrates the problem that results when local image likelihood measures for each body part do not take into account the poses of other parts and do not exploit any knowledge of what image evidence is left unexplained. This problem is not unique to human pose estimation and applies to other generic object-recognition domains.

Recent attempts to solve the problems illustrated in Figure 6.4 have focused on the use of strong prior models of body pose that rule out unlikely poses [122]. These approaches are not appropriate for dealing with unexpected or unusual motions such as those in Figure 6.3. In particular, they require that we already know the activity being observed and that the variation in the pose is within learned limits. Other computational strategies incrementally explore the space of body poses but give up the formal probabilistic interpretation of graphical model [170]. In this chapter we argue that such approaches are fighting the wrong image likelihood and that the solution lies in the proper formulation of this likelihood function. A fully global likelihood is

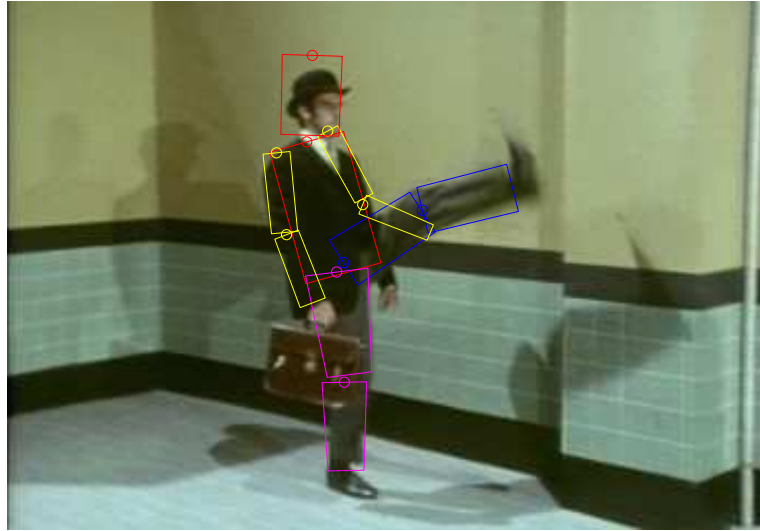


Figure 6.3: **Silly walks.** The detection of 2D body pose in real images is challenging due to complex background appearance, loose monochromatic clothing, and the sometimes unexpected nature of human motion. In this scene, strong, activity-dependent, prior models of human pose are too restrictive. The result here was found by our method which makes weak assumptions about body pose but uses a new occlusion-sensitive image likelihood.

computationally impractical and consequently we develop a principled approximation to the global likelihood that is sensitive to local occlusion relationships between parts.

The resulting 2D pose estimation is an adaptation of the *loose-limbed body model* introduced in the previous chapter for the purposes of monocular 2D pose estimation. As before, simple body part detectors provide noisy probabilistic proposals for the location and 2D pose (orientation and foreshortening) of visible limbs (Figure 6.2 (b)). The pose is estimated by inference in the view-based 2D graphical model representation of the body. As before we also use a variant of non-parametric belief propagation (PAMPAS) [99, 220] to infer probability distributions representing the belief in the 2D pose of each limb (Figure 6.2 (c)). The inference algorithm also introduces hidden binary occlusion variables and marginalizes over them to account for occlusion relationships between body parts. The bi-directional conditional distributions linking 2D body parts are learned from examples (similarly to Chapter 5).

This process of using limb proposals and non-parametric inference in a graphical model provides reasonable guesses for 2D body pose from which to estimate the 3D pose of the body. Sminchisescu *et al.* [206] and Agarwal and Triggs [2] learned a probabilistic mapping from 2D silhouettes to 3D pose using a Mixture of Experts (MoE) model. We extend their approach to learn a mapping from 2D poses (including joint angles and foreshortening information) to 3D poses. The approach uses a mixture of regularized linear regression models that are trained from a set of 2D-3D pose pairs obtained from motion capture data.

Sampling from this model provides predicted 3D poses (Figure 6.2 (d)), that are appropriate as proposals for a Bayesian temporal inference process (Figure 6.2 (e)). Our multi-stage approach overcomes many of the problems inherent in inferring 3D pose directly from image features. The proposed hierarchical Bayesian inference process copes with the complexity of the problem through the use of intermediate generative 2D model.

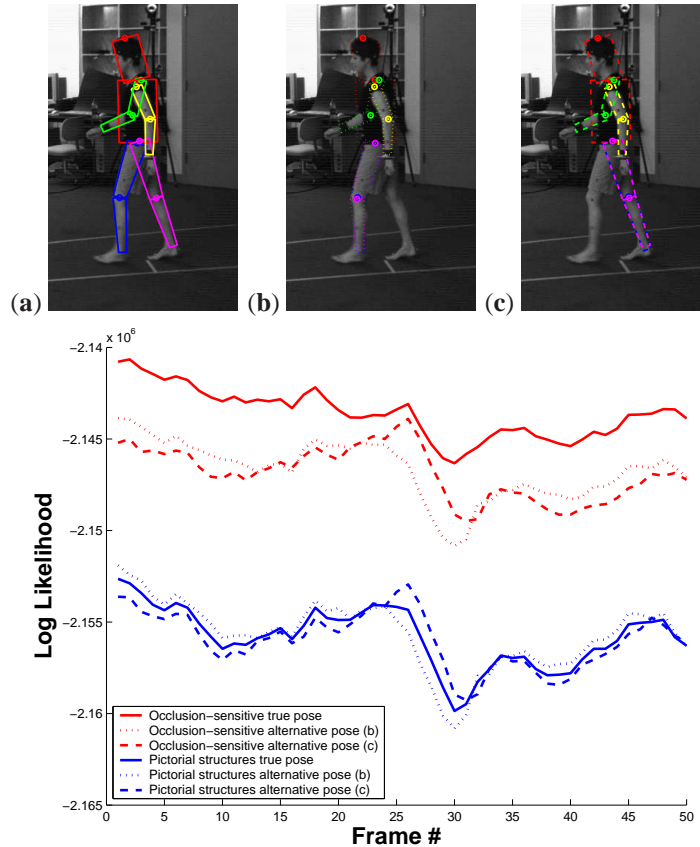


Figure 6.4: **Fighting the likelihood.** (a) shows the ground truth body pose while (b) and (c) show common failure modes of pictorial structure approaches in which both legs explain the same image data. With local image likelihoods, the poses in (b) and (c) are often better interpretations of the scene than the true pose. This can be seen in the plot where 50 frames of a test sequence are evaluated. The blue curves illustrate the local pictorial structures likelihood. The likelihood of the ground truth is solid blue while the likelihoods for the two alternative poses (both legs front or both legs back) are shown as dashed lines. The local likelihood marginally prefers the true pose in only 2 out of 50 frames tested. With our proposed occlusion-sensitive likelihood (shown in red) the true pose is always more likely than the alternative poses.

We qualitatively and quantitatively evaluate our 2D pose estimation procedure, comparing the performance to the state-of-the-art discrete tree-structured model of Felzenszwalb and Huttenlocher [59] and results published in [122]. We show that our continuous-state, occlusion-sensitive, model is better suited, in terms of quantitative performance, for 2D pose inference. We also quantitatively evaluate the 3D proposals using ground truth 2D poses. Finally, we test the full hierarchical inference strategy proposed in this chapter on the monocular sequence in Figure 6.2. We test both automated 3D pose inference from monocular static frames, as well as tracking.

6.1 Previous Work

Generative, model-based, approaches for recovering 2D articulated pose can be loosely classified into two categories. Top-to-bottom approaches treat the body as a “cardboard person” [111] in which the limbs are represented by 2D patches connected by joints. These patches are connected in a kinematic tree [30, 52, 90,

147, 173, 193, 209] and the pose of the person is represented by a high-dimensional state vector that includes the position and orientation of the root limb in the global image coordinate frame and the parameters of each limb relative to its parent in the tree. The high-dimensional state space makes exhaustive search for the body pose difficult. While impractical for pose estimation from a single frame, these methods have been shown to be appropriate and effective for tracking.

In contrast, bottom-up approaches address the dimensionality of the state space by representing each part independently in the 2D image coordinate frame. In such models a body part is represented as a node in a graph and edges in the graph represent kinematic constraints between connected parts. This formulation allows independent search for the parts which are then combined subject to the kinematic constraints. The results are typically imprecise, but enable automatic initialization (pose estimation). These “Pictorial Structures” approaches assume the graph of the body is a tree, which makes inference tractable [59, 170, 178]. While efficient Belief Propagation inference methods¹ in these graphical models exist [59], they require a discretization of the state space of 2D limb poses and simple forms for the conditional distributions relating connected limbs (see discussion in Section 3.5.2).

The pictorial structures approach also has problems as illustrated in Figure 6.4 where multiple body parts explain the same image regions. The problems arise from the assumption that the global image likelihood can be expressed as a product of individual local terms (one per part), without regard to occlusions. As a result, as shown in Figure 6.4, we find that the true pose is almost always (in 48 out of 50 frames tested) less likely than the alternative hypothesis that corresponds to the local maximum. To deal with this, previous algorithms have sampled multiple poses from the solution space and then used an external global likelihood to choose among the sampled hypothesis [59]. This approach however requires smoothing of likelihood functions, to ensure that the true pose is sampled. The direct *maximum a posteriori*² (MAP) estimate of the posterior almost always results in the undesired solution. Alternatively, Ramanan and Forsyth [170] first find a solution for one side of the body and then remove the image regions explained by that solution from future consideration. They then solve for the other side of the body independently. While this sidesteps the problem it does not explicitly model the possible occlusion relationships and the algorithmic solution loses the probabilistic elegance present in the graphical model formulation. A more recent approach of Kumar *et al.* [121] acknowledges that occlusions of parts must be accounted for and proposes a layered pictorial structure model that exhaustively searches over the depth-based layering of parts. The resulting approach is more robust, but requires video for on-line learning of the layering model.

Alternatively one can impose strong global constraints on the allowed poses that prohibit solutions like those in Figure 6.4 (b) and (c) [122]. In [122] a single latent variable that accounts for the unmodeled correlation between parts of the body is added. This may be appropriate when the activity is known and the range of poses is highly constrained; for example, walking poses can be represented using a small number of hidden variables [160]. We argue that these strong priors are invoked to deal with inadequate image likelihoods. In Figure 6.4 the local likelihoods prefer the *wrong* solutions and hence the prior is *fighting* with the likelihood to undo its mistakes. Furthermore strong priors are unable to cope with unusual activities such

¹Belief Propagation inference in these graphical models can be recast and solved using dynamic programming.

²Maximum a posteriori (MAP) (*a.k.a.* posterior mode) estimation is often used to obtain a point estimate of the posterior distribution. It is closely related to maximum likelihood (ML) estimation, but can incorporate a prior distribution over the variables, and hence can be seen as a regularization of ML estimation. Often the MAP estimate is computed in the cases where the expected value of the posterior density cannot be computed explicitly.

as the one in Figure 6.3.

The closest work to ours, addresses the problem with the image likelihoods in these decentralized models in the context of 3D articulated hand pose estimation [219]. They explicitly model occlusions in 3D and deal with distributed reasoning in graphical models using Non-parametric Belief Propagation [220]. The approach deals with the issue of overcounting image evidence but does not address the problem of having the model explain as much of the image evidence as possible locally. As a result the particular message passing order must be imposed to account for the layering of parts. They also only deal with tracking from a hand initialized pose; here we go further to deal with automatic pose estimation (initialization). Consequently, our formulation allows for more general likelihoods. Our framework also uses a slightly different inference approach, from the one introduced in [220], that extends the basic Particle Message Passing (PAMPAS) algorithm introduced in previous chapters.

In summary, to address articulated 2D pose estimation from monocular static imagery, we propose a method for approximating the global likelihood using consistent local likelihoods. This allows us to use a part-based graphical model of the body and perform inference with a generic approximate BP algorithm, PAMPAS. Unlike [59] we deal with the continuous estimation of part locations, orientation, foreshortening and scale. Like previous approaches, for now we assume a known view but multiple views can be searched simultaneously and it is relatively straightforward to compare the results to select the best view. Without strong priors, the method finds solutions that better explain the image evidence, and results in more accurate pose estimation. Also, unlike previous methods [59, 196] we infer 2D pose as an intermediate step to inferring the full 3D articulated body pose.

Lee and Cohen [127] also use a bottom-up proposal process and infer 3D pose parameters using a data-driven MCMC procedure. Our approach differs in that we break the problem into simpler pieces: generate 2D proposals, inference of 2D pose, and prediction from 2D to 3D. We are also able to incorporate temporal coherence (tracking) where appropriate.

The 2D to 3D inference stage has received a good deal of attention with a variety of geometric [147, 222] and machine learning methods [2, 3, 4, 181, 189, 206] being employed. Most previous approaches have focused on directly inferring 3D pose from 2D silhouettes which may be difficult to obtain in general. Additionally silhouettes contain less information than our 2D models which represent all the limbs, the joint angles, and foreshortening. This helps reduce the ambiguities found in matching silhouettes to 3D models [209] but does not remove ambiguities altogether. Consequently we learn a conditional distribution using a Mixture of Experts (MoE) model similar to that of Sminchisescu [206] and Agarwal and Triggs [2]. Our work is similar in spirit to [90] in which 3D poses are inferred from 2D tracking results, but our approach can infer 3D pose from a single monocular image and does not require manual initialization.

6.2 Modeling a Person

We model a 3D human body using a set of P (here $P = 10$) tapered cylinders corresponding to body parts and connected by revolute joints. Each part has an associated set of fixed parameters that are assumed to be known (*e.g.* length and cross-sectional radius at the two joints). We represent the overall pose of the body $\mathbf{Y}_t = [\Xi_t, \Gamma_t, \Theta_t]^T$ at time t using a set of joint angles Θ_t , a global position Ξ_t , and global orientation Γ_t in 3D. Joint angles are represented with respect to the kinematic chain along which they are defined using

unit quaternions (see Kinematic Tree model discussed in Section 2.4.1). For our body model, this results in $\mathbf{Y}_t \in \mathbb{R}^{47}$, or $\mathbf{Y}_t \in \mathbb{R}^{55}$ depending on whether one chooses to model the clavicle joints. We tested our approach with both parameterizations. For simpler motions (walking) we revert to the lower-dimensional state space which proved to be adequate, for more complex motions (dancing/ballet) that allow for a higher degree of flexibility in the body pose we resort to the more complex higher-dimensional parameterization.

In 2D the limbs in the image plane are modeled by trapezoids, obtained by projecting tapered cylinders from above into image plane. The overall body pose is defined using a redundant representation (introduced in previous chapter) $\mathbf{X}_t = \{\mathbf{X}_{1,t}, \mathbf{X}_{2,t}, \dots, \mathbf{X}_{P,t}\}$ in terms of 2D position, rotation, scale and foreshortening of parts, $\mathbf{X}_{i,t} \in \mathbb{R}^5$. This redundant representation (see Figure 6.5) stems from the inference algorithm that we will employ to infer the pose of the body in 2D. To simplify notation we will drop the temporal sub-script t where not necessary (*e.g.* letting $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_P\}$ instead of $\mathbf{X}_t = \{\mathbf{X}_{1,t}, \mathbf{X}_{2,t}, \dots, \mathbf{X}_{P,t}\}$, since 2D inference is done independently at every frame).

Notice that the decentralized, redundant, representation of the pose introduced in the previous section is only being employed for the 2D body pose, not for the 3D pose. The reason for this stems from the architecture we choose for our hierarchical framework. Since, inference from 2D pose to 3D pose takes form of the Mixture of Experts, or put simply multivalued regression, there is little benefit in using decentralized loose-limbed body model representation for the 3D pose; doing so would increase overall dimensionality of the 3D pose, without any computational benefit.

6.3 Finding an Articulated Pose of a Person in 2D

In 2D, the body is represented as a graphical model (Figure 6.5) in which nodes in the graph correspond to the rigid body parts and undirected edges to the probabilistic constraints between parts encoded using pairs of consistent conditional distributions. This redundant but decentralized representation allows for tractable inference, by partitioning the search for the pose in a high dimensional space into a number of lower dimensional distributed searches that collaborate to infer the overall state of the body, subject to the imposed constraints.

6.3.1 Likelihood

To estimate the pose of an object we must be able to evaluate how well different body configurations explain observed image data. We formalize this using a probabilistic likelihood function that takes a body pose and the image evidence and returns the likelihood of the pose. The desired properties of a good likelihood function lie in its robustness to partial occlusions, camera noise, changing lighting and the variability of appearance of the body. We will build on the likelihood formulation introduced in Section 5.4, extending it to account for self-occlusions of the body that may result from articulation of the body itself under particular viewing direction. While self-occlusions can also present themselves in multi-camera scenarios, in practice they are much more problematic in monocular imagery. In multi-camera observations occlusions typically occur in only one of the many views of the object, and hence can often be ignored (as we have done in Chapter 5).

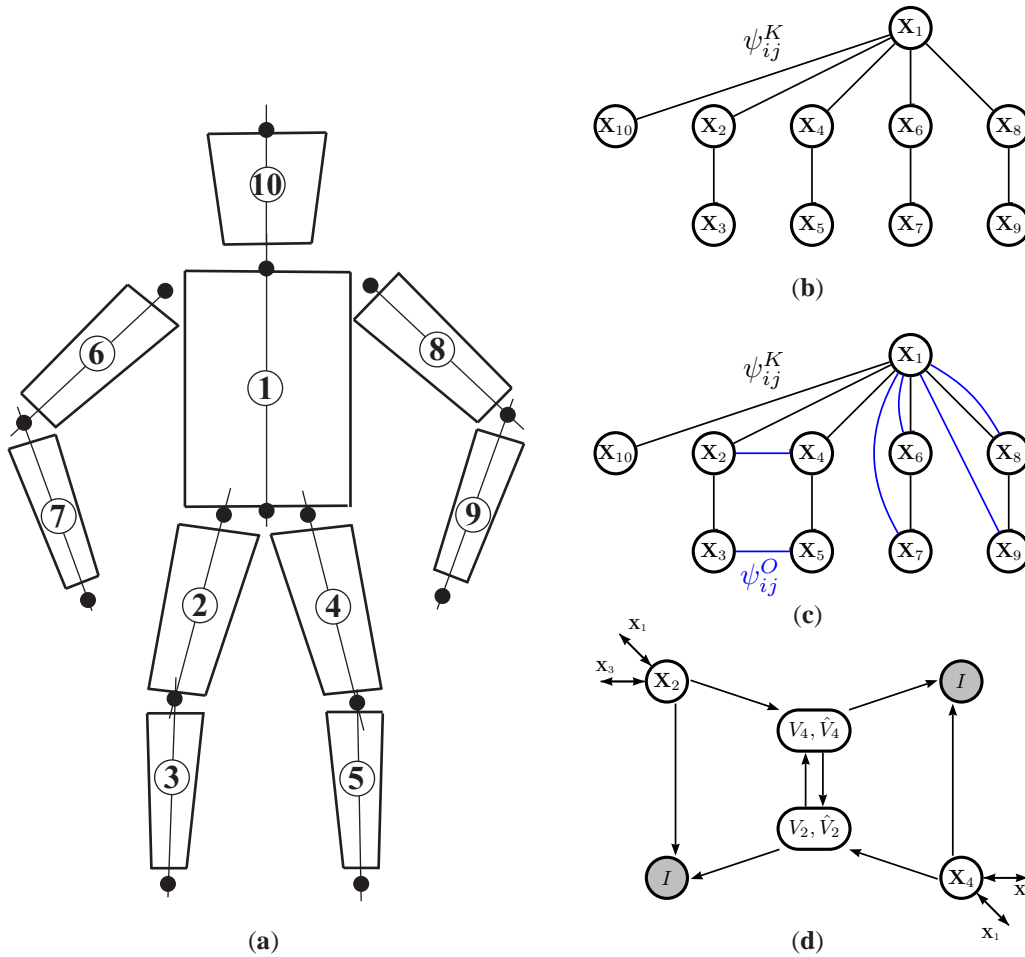


Figure 6.5: **Representing 2D body as a graph.** Figure (a) shows the representation of the 2D body as a graph with body parts labeled using the corresponding node numbers; (b) shows the corresponding tree-based representation of the body, and (c) our extended body model that contains additional occlusion constraints designated by edges in blue; (d) shows actual directed graphical model interactions encoded by a single blue edge in (c) between X_2 and X_4 ; I is the image evidence.

Global vs. Local Image Likelihoods

Given the state of the body \mathbf{X} , we define a global likelihood $\phi(I|\mathbf{X})$ in terms of some features I (with slight abuse of notation) observed in an image. For convenience, we assume that these features are defined per-pixel and on a pixel grid. To support distributed modeling of the body we write this global likelihood as the product of local likelihood terms $\phi(I|\mathbf{X}) \propto \prod_{i \in [1, \dots, P]} \phi_i(I|\mathbf{X}_i)$. Drawing inspiration from [59] and [260], we define local likelihoods, as in previous chapter, in terms of the product of individual pixel likelihoods in sub-regions of the image that are defined by the local state \mathbf{X}_i . For clarity we re-state the likelihood formulation introduced in Section 5.4, in a slightly more general form, here.

Formally, we assume that pixels in a feature image, I , can be partitioned into three disjoint sub-sets $\Omega_1(\mathbf{X}_i) \cup \Omega_2(\mathbf{X}_i) \cup \Omega_3(\mathbf{X}_i) = \Upsilon$, where Υ is the set of all pixel grid positions $u \equiv (x, y)$ in an image; $\Omega_1(\mathbf{X}_i)$ is the set of pixels enclosed by part i as defined by the state \mathbf{X}_i ; $\Omega_2(\mathbf{X}_i)$ contains the pixels outside part i that are statistically correlated with the part i (for example pixels in the border slightly outside the

limb); and $\Omega_3(\mathbf{X}_i) \equiv \Upsilon - (\Omega_1(\mathbf{X}_i) \cup \Omega_2(\mathbf{X}_i))$ which corresponds to the set of pixels where we assume no correlation of the image statistics based on the pose of part i . Assuming pixel independence in the feature image, we write the local likelihood $\phi(I|\mathbf{X}_i)$ for the part i as a product of individual pixel probabilities as

$$\phi_i(I|\mathbf{X}_i) = \prod_{u \in \Omega_1(\mathbf{X}_i)} p_1(I_u) \prod_{s \in \Omega_2(\mathbf{X}_i)} p_2(I_s) \prod_{r \in \Omega_3(\mathbf{X}_i)} p_3(I_r) \quad (6.1)$$

for features I_u , $u = (x, y) \in \Upsilon$.

The standard pictorial structures silhouette likelihood, (denoted by FG in Chapter 5) [59] can easily be written in this form,

$$\phi_{i,fg}(I|\mathbf{X}_i) = \prod_{u \in \Omega_1(\mathbf{X}_i)} p_{1,FG}(I_u) \prod_{s \in \Omega_2(\mathbf{X}_i)} p_{2,FG}(I_s) \prod_{r \in \Omega_3(\mathbf{X}_i)} p_{3,FG}(I_r), \quad (6.2)$$

by letting I_u be a silhouette image obtained by background subtraction (*i.e.* $I_u \equiv FG_c(x, y)$ using notation of Chapter 5) and by setting

$$\begin{aligned} p_{1,FG}(I_u) &= \begin{cases} q_1 & \text{if } I_u = 1 \\ 1 - q_1 & \text{otherwise} \end{cases} \\ p_{2,FG}(I_u) &= \begin{cases} q_2 & \text{if } I_u = 1 \\ 1 - q_2 & \text{otherwise} \end{cases} \\ p_{3,FG}(I_u) &= 0.5 \end{aligned} \quad (6.3)$$

for some constants $0 \leq q_i \leq 1$. For other non binary features such as limb/skin color (denoted here by C) we can express $p_{1,C}(I_u)$ and $p_{2,C}(I_u)$ as a per pixel ratio of learned foreground and background distributions; for example

$$\begin{aligned} p_{1,C}(I_u) &= \frac{p_{skin}(I_u)}{p_{skin}(I_u) + p_{bgd}(I_u)} \\ p_{2,C}(I_u) &= \frac{p_{bgd}(I_u)}{p_{skin}(I_u) + p_{bgd}(I_u)} \\ p_{3,C}(I_u) &= 0.5 \end{aligned} \quad (6.4)$$

where I_u , in this case, is simply a pixel value of the original image.

Occlusion-sensitive Local Likelihoods

The above formulation is only valid if the local terms $\phi_i(I|\mathbf{X}_i)$ for $i \in [1, \dots, P]$ are independent. In absence of occlusions, this assumption holds and likelihoods factor. When limbs occlude each other, however, the assumption does not hold and the product of local likelihoods gives a poor approximation to the global likelihood (see Figure 6.4).

To allow a similar decomposition (and hence distributed inference) when occlusions exist, we augment the state, \mathbf{X}_i , of limb i with two sets of binary hidden variables $V_i = \{v_{i,u}\}$ and $\hat{V}_i = \{\hat{v}_{i,u}\}$, where u is a pixel $u \in \Upsilon$. Let $v_{i,u} = 0$ if pixel u for the part i is occluded by any other body part, and 1 otherwise.

Intuitively this corresponds to the “visibility” of the part i at a given pixel u . Notice that if $v_{i,u} = 1$ for some pixel $u \in \Omega_1(\mathbf{X}_i)$, then we know that part i at a given pose \mathbf{X}_i generated pixel u in the image. Similarly we let $\hat{v}_{i,u} = 0$ if at pixel u for part i at \mathbf{X}_i is occluding any other part, and 1 otherwise. Intuitively \hat{V}_i encodes which pixels in the image could possibly be explained by other body parts that are further away from the camera. In particular, if $v_{i,s} = 1$ and $\hat{v}_{i,s} = 1$ for a pixel slightly outside part i , $s \in \Omega_2(\mathbf{X}_i)$, then that pixel, s , must have been generated by a background model (since by definition there cannot be any other part in front or behind i at s). Intuitively V_i and \hat{V}_i in conjunction allow the likelihood to not only be sensitive to occlusions [219] but also to reason locally about globally plausible explanations of the image. In other words, each limb maintains a summary of what is in front and behind it via binary masks V_i and \hat{V}_i .

An illustration of these visibility variables is shown in Figure 6.6. For example, Figure 6.6 (c) indicates that the torso is occluded by the lower arm ($v_{i,u} = 0$) and Figure 6.6 (g) indicates that the arm is occluding part of the torso ($\hat{v}_{i,u} = 0$).

Modifying our likelihood, to take into account the hidden per-pixel binary occlusion variables we have

$$\phi_i(I|\mathbf{X}_i, V_i, \hat{V}_i) = \prod_{u \in \Omega_1(\mathbf{X}_i)} [p_1(I_u)]^{v_{i,u}} \prod_{s \in \Omega_2(\mathbf{X}_i)} [p_2(I_s)]^{v_{i,s} \hat{v}_{i,s}} \prod_{r \in \Omega_3(\mathbf{X}_i)} [p_3(I_r)]^{v_{i,r} \hat{v}_{i,r}}. \quad (6.5)$$

Notice that $v_{i,u}$ and $\hat{v}_{i,u}$ are simply used as selectors. If pixel $u \in \Omega_1(\mathbf{X}_i)$ is unoccluded then contribution of pixel u , $p_1(I_u)$, to the likelihood will be considered. Similarly, if pixel $s \in \Omega_2(\mathbf{X}_1)$ is both unoccluded and unexplained then its contribution will be considered as well. Pixels for which $v_{i,u} = 0$ and/or $\hat{v}_{i,u} = 0$ will have constant likelihood 1.

The per-pixel occlusion-sensitive likelihoods are shown in Figure 6.6 for the torso (e) and lower arm (h). The local estimate of the global likelihood is simply the product of the pixel likelihoods, where brighter indicates more likely.

It is important to note that conditioned on the sets of hidden variables V_i and \hat{V}_i the local likelihoods $\phi_i(I|\mathbf{X}_i, V_i, \hat{V}_i)$ are truly independent if V_i and \hat{V}_i are consistent across all $i \in [1, \dots, P]$. By consistency here we mean that parts do not assume mutually occluding states for example (meaning that there may exist only one part i for which $v_{i,u} = 1$, for all others $v_{j,u} = 0$, where $j \in [1, \dots, P]/i$). This ensures that $\phi(I|\mathbf{Y}) \propto \prod_{i \in [1, \dots, P]} \phi_i(I|\mathbf{X}_i, V_i, \hat{V}_i)$ always holds.

6.3.2 Modeling Constraints

The 2D body in our decentralized model is represented by constraints between the parts that express traditional kinematic relationships (similar to those described in Section 5.3.1 but in 2D) as well as occlusion relationships between possibly occluding parts.

Occlusion Constraints

Enforcing the consistency of the hidden occlusion variables V_i and \hat{V}_i requires reasoning that involves all potentially occluding and occluded parts for any given node i . We can express these occlusion constraints using pairwise potential functions $\psi_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j, \mathbf{X}_i, V_i, \hat{V}_i)$ between every pair of potentially occluding parts i and j . We formally encode the consistency of all occlusion relationships between part i and j using the unnormalized distribution:

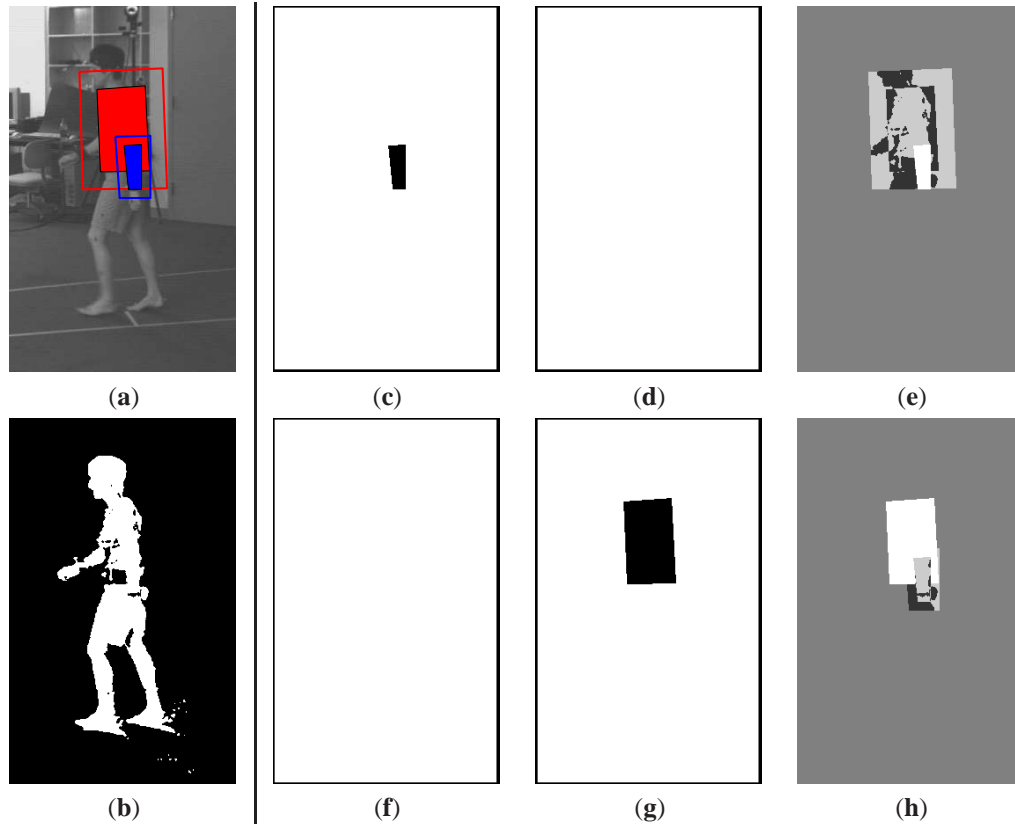


Figure 6.6: **Occlusion-sensitive likelihood.** Two overlapping parts (torso and lower arm) are shown in (a). The solid regions correspond to Ω_1 while the regions outside but enclosed by the line correspond to Ω_2 . (b) shows the observed silhouette; (c) and (f) show the state of the hidden variables V_i for the torso and left lower arm respectively; (d) and (g) show the corresponding states of the \hat{V}_i 's; (e) and (h) shows the per pixel local occlusion-sensitive likelihoods with pixel brightness corresponding to high probability. Notice that in the cases where a part is both occluded and occluding other parts, both V_i and \hat{V}_i will contain non-uniform structure.

$$\psi_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j, \mathbf{X}_i, V_i, \hat{V}_i) \propto \prod_{u \in \Upsilon} \begin{cases} 0 & \text{if } \mathbf{X}_j \text{ occludes } \mathbf{X}_i, u \in \Omega_1(\mathbf{X}_j), v_{i,u} = 1 \\ 0 & \text{if } \mathbf{X}_i \text{ occludes } \mathbf{X}_j, u \in \Omega_1(\mathbf{X}_i), v_{j,u} = 1 \\ 0 & \text{if } \mathbf{X}_j \text{ occludes } \mathbf{X}_i, u \in \Omega_1(\mathbf{X}_i), \hat{v}_{j,u} = 1 \\ 0 & \text{if } \mathbf{X}_i \text{ occludes } \mathbf{X}_j, u \in \Omega_1(\mathbf{X}_j), \hat{v}_{i,u} = 1 \\ 1 & \text{otherwise} \end{cases} \quad (6.6)$$

Intuitively this simply enumerates all inconsistent cases and assigns them 0 probability. The first case for example can be interpreted as the following: if \mathbf{X}_j occludes \mathbf{X}_i and any pixel u is inside the image region of occluding part j , then $v_{i,u}$ corresponding to the visibility of the occluded part i at the pixel u must be set to 0.

Kinematic Constraints

Every pair of connected parts (i, j) in the body also has an associated kinematic potential function that enforces kinematic constraints and positions of joints. As before, see Section 5.3.1, potentials are modeled

using convenient robust Gaussian mixture representation but in 2D. Also, as before, we learn the conditional distributions, corresponding to these potential functions, directly instead of indirectly deriving them from the joint distributions (see discussion in Section 5.3.1). The conditional distributions were learned separately for 8 view-based models using 3D motion capture data. The training motion capture data was partitioned according to the relative heading of the body with respect to the camera viewing direction at 45 degree increments. The 3D body pose was projected into a desired camera view and the conditionals were learned from the 2D projections of individual limbs. As before, we used a standard iterative Expectation-Maximization (EM) algorithm with K-means initialization for learning the Gaussian mixture model (GMM) (see Algorithm 1 in Section 3.4.2 for details). For all experiments in this chapter we used $M_{ij} = 8$ mixture components.

6.3.3 Inference

Inference in the standard pictorial structures model involves estimating the location and pose of every body part. With our occlusion-sensitive model we have the additional problem of dealing with the hidden occlusion variables. Given the formulation above, the joint probability for the graphical model with P body parts, can be written as

$$p(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_P | I) \propto \sum_{V_i} \sum_{\hat{V}_i} [\prod_{ij} \psi_{ij}^K(\mathbf{X}_j, \mathbf{X}_i) \prod_{ij} \psi_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j, \mathbf{X}_i, V_i, \hat{V}_i) \prod_j \phi_i(I | \mathbf{X}_j, V_j, \hat{V}_j)] \quad (6.7)$$

where \mathbf{X}_i represents the state of the limb i ; $\psi_{ij}^K(\mathbf{X}_j, \mathbf{X}_i)$ is the kinematic compatibility term between the connected nodes i and j ; $\psi_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j, \mathbf{X}_i, V_i, \hat{V}_i)$ is the occlusion compatibility between potentially occluding nodes i and j and $\phi_i(I | \mathbf{X}_i, V_i, \hat{V}_i)$ is the local image likelihood. The two summations marginalize over the hidden occlusion variables in V_i and \hat{V}_i . Notice, unlike other graphical models introduced in this thesis where the states are either continuous or discrete, the model introduced here has both continuous variables (corresponding to the 2D location of parts) and discrete variables (corresponding to per-pixel occlusions).

We solve for the part poses using belief propagation where the message update equations are:

$$m_{ij}^K(\mathbf{X}_j) = \int \sum_{\mathbf{X}_i} \sum_{V_i} \sum_{\hat{V}_i} [\psi_{ij}^K(\mathbf{X}_j, \mathbf{X}_i) \phi_i(I | \mathbf{X}_i, V_i, \hat{V}_i) \prod_{k \in A/j} m_{ki}^K(\mathbf{X}_i) m_{ki}^O(\mathbf{X}_i, V_i, \hat{V}_i)], \quad (6.8)$$

$$m_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j) = \int \sum_{\mathbf{X}_i} \sum_{V_i} \sum_{\hat{V}_i} [\psi_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j, \mathbf{X}_i, V_i, \hat{V}_i) \phi_i(I | \mathbf{X}_i, V_i, \hat{V}_i) \prod_{k \in A/j} m_{ki}^K(\mathbf{X}_i) m_{ki}^O(\mathbf{X}_i, V_i, \hat{V}_i)]. \quad (6.9)$$

Inferring the state of the 2D body in our graphical model representation corresponds to estimating the belief (marginal) at each node in a graph,

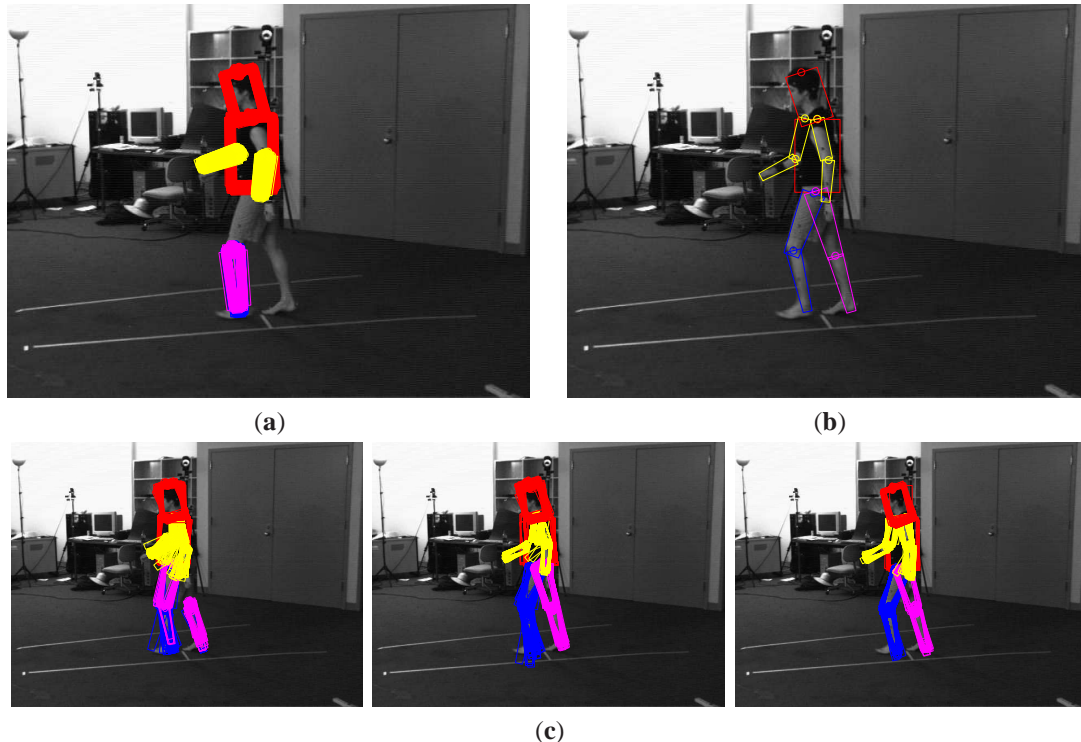


Figure 6.7: **Occlusion-sensitive inference.** Figure (a) shows the proposal distributions for the six body parts drawn from the ground truth pose and corrupted by Gaussian noise. Both left and right calves are initialized intentionally incorrectly on the left calf in the image; (b) shows the mean of the marginal distribution for each part after 3 iterations of belief propagation (BP). Figure (c) shows 100 samples from the marginal distributions after one, two and three iterations of BP. Notice that we initialize from a local maximum of the traditional likelihood function, precisely the place where most algorithms get “stuck”, yet our algorithm is still able to recover the correct pose.

$$b_i(\mathbf{X}_i) = \sum_{V_i} \sum_{\hat{V}_i} \phi_i(I|\mathbf{X}_i, V_i, \hat{V}_i) \prod_{k \in A} m_{ki}^K(\mathbf{X}_i) m_{ki}^O(\mathbf{X}_i, V_i, \hat{V}_i). \quad (6.10)$$

We use PAMPAS [99], as in previous chapters, to deal with this task. The messages are approximated using a kernel density formed by propagating particles through a conditional density (see Section 3.7 for details). In all the experiments we used 100 particles which, when propagated through the conditionals represented by mixtures of 8 Gaussians, resulted in density representation for the messages with $N = 800$ Gaussian kernels. We modify the method to include an annealing step [52] with each iteration of PAMPAS, as discussed in Section 3.7.7, that gradually introduces the effects of peaks in our local likelihoods. For the details on how the message updates are carried out using stratified sampling from the products of messages and a static proposal distribution see Algorithm 6 in Section 3.7. The illustration of the inference using PAMPAS with occlusion-sensitive likelihoods can be seen in Figure 6.7. Consequently, in Figure 6.7 we start intentionally from undesired initial conditions, where both legs are found in the same location; yet our occlusion-sensitive model is able to successfully recover from this local maximum.

Message Updating for Occlusion Messages

It is intractable to sample occlusion variables V_i and \hat{V}_i due to the exponentially large number of possible occlusion mask configurations. Consequently we approximate the computation of the marginals using an analytic procedure introduced in [219]. Assuming we know depth ordering for the parts in a given view we compute the approximate message $m_{ij}^O(\mathbf{X}_j, V_j, \hat{V}_j)$ for V_j and \hat{V}_j explicitly. To do so, we must consider two cases: (1) where \mathbf{X}_j is occluded by \mathbf{X}_i and (2) where \mathbf{X}_i is occluding \mathbf{X}_j . We assume that potentially occluding parts have a known and unchanging depth order to simplify the formulation. In general, we could introduce an additional discrete hidden variable designating the depth order between parts and marginalize over it as well, which would lead to a more complex inference scheme.

If \mathbf{X}_j is occluded by \mathbf{X}_i the message from \mathbf{X}_i to \mathbf{X}_j about the state of \hat{V}_j is uninformative and can be written in terms of individual per-pixel hidden binary variables as $m_{ij}^O(\hat{v}_{j,u} = 1) = 1$, for all $u \in \Upsilon$. The message for V_j is informative, however, and can be approximately computed as $m_{ij}^O(v_{j,u} = 1) \propto 1 - p(u \in \Omega_1(\mathbf{X}_i))$, where $p(u \in \Omega_1(\mathbf{X}_i))$ is simply the probability of pixel $u \in \Upsilon$ being inside the projection of \mathbf{X}_i . Similar expressions can be derived for the case where \mathbf{X}_j is occluding \mathbf{X}_i .

We can now approximate the marginal probability of a pixel u being “visible” for part j , $p(v_{j,u} = 1)$, by taking a product over all potential occluders,

$$p(v_{j,u} = 1) \propto \prod_i m_{ij}^O(v_{j,u} = 1). \quad (6.11)$$

Since $v_{j,u}$ is binary, the occlusion probability is simply $p(v_{j,u} = 0) = 1 - p(v_{j,u} = 1)$. Similarly for $p(\hat{v}_{j,u} = 1) \propto \prod_i m_{ij}^O(\hat{v}_{j,u} = 1)$, where $p(\hat{v}_{j,u} = 1)$ is the marginal probability of the pixel u not being explained by any other part i that is behind part j (further away from the camera). Computation of these marginals amount to “projecting” the distribution (represented in terms of weighted particles) for every possible occluder \mathbf{X}_i into the image and summing over the resulting weighted binary masks (with normalization).

We can now re-write the likelihood functions in terms of the marginal probabilities $z_{j,u} \equiv p(v_{j,u} = 1)$ and $\hat{z}_{j,u} \equiv p(\hat{v}_{j,u} = 1)$,

$$\begin{aligned} \phi_i(I|\mathbf{X}_j, V_j, \hat{V}_j) = & \quad (6.12) \\ & \prod_{u \in \Omega_1(\mathbf{X}_j)} [(1 - z_{j,u}) + z_{j,u} p_1(I_u)] \\ & \prod_{s \in \Omega_2(\mathbf{X}_j)} [(1 - z_{j,s} \hat{z}_{j,s}) + z_{j,s} \hat{z}_{j,s} p_2(I_s)] \\ & \prod_{r \in \Omega_3(\mathbf{X}_j)} [(1 - z_{j,r} \hat{z}_{j,r}) + z_{j,r} \hat{z}_{j,r} p_3(I_r)]. \end{aligned}$$

This equation downweights the image evidence for the part j at a pixel $u \in \Omega_1(\mathbf{X}_j)$ as the probability of that pixel’s visibility decreases (occlusion probability increases). Similarly, it also downweights the image evidence at the pixel $s \in \Omega_2(\mathbf{X}_j)$ as the probability of that pixel being explained by another body part further away from the camera increases. Notice that this likelihood can be implemented efficiently by only considering regions of the image $\Omega_1(\mathbf{X}_j)$ and $\Omega_2(\mathbf{X}_j)$ for a given \mathbf{X}_j , and precomputing $\prod_{r \in \Upsilon} [(1 - z_{j,r} \hat{z}_{j,r}) + z_{j,r} \hat{z}_{j,r} p_3(I_r)]$.

Limb Proposals

Plausible poses/states for some or all the body parts are needed as proposals to initiate inference (see Section 3.7.3). There exist a number of efficient methods for detecting 2D body parts in an image [127, 147, 176]. Among them approaches for face detection [236], skin color-based limb segmentation [127, 128], and color-based segmentation exploiting the homogeneity and the relative spatial extent of body parts [127, 128, 147, 176]. Here we took a simple approach and constructed a set of proposals by coarsely discretizing the state space and evaluating local part-based likelihood functions at these discrete locations. For all of the experiments here we discretized the state space into 5 scales, 5 foreshortenings, 20 vertical and 20 horizontal positions, and 8 rotations. Out of $5 \times 5 \times 20 \times 20 \times 8 = 80,000$ evaluated discrete states, we chose the 100 most likely states for each part and used these as a particle based proposal distribution for belief propagation. It is important to note that not all parts need to be detected and, in fact, detecting all the parts is largely impossible due to the self occlusions. To initialize the search we used, as in Chapter 5, proposals for 6 parts: torso, head and four outermost extremities. All other parts were initialized with a uniform distribution over the entire state space.

6.4 Proposing 3D Body Model from 2D

In order to produce estimates for the body in 3D from the 2D body poses, we need to model the conditional distribution $p(\mathbf{Y}|\mathbf{X})$ of the 3D body state \mathbf{Y} given 2D body state \mathbf{X} . Intuitively this conditional mapping should be related to the inverse of the camera projection matrix and, as with many inverse problems, is highly ambiguous.

To model this non-linear relationship we use a Mixtures of Experts (MoE) model to represent the conditionals [3, 4, 206]. The more complete definition of MoE model and the learning procedure can be found in Section 3.8.2, here we briefly restate³ the process for convenience. The parameters of the MoE model are learned by maximizing the log-likelihood of the training data set $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ consisting of N input-output pairs (x_i, y_i) . We use an iterative Bayesian EM algorithm, based on maximum likelihood, to learn parameters of the MoE. Our model for the conditional can be written as:

$$p(\mathbf{Y}|\mathbf{X}) = \sum_{m=1}^M p_e(\mathbf{Y}|\mathbf{X}, z_m = 1, \theta_{e,m}) p_g(z_m = 1|\mathbf{X}, \theta_{g,m}) \quad (6.13)$$

where $p_e(\mathbf{Y}|\mathbf{X}, z_m = 1, \theta_{e,m})$ is the probability of choosing pose \mathbf{Y} given the input \mathbf{X} according to the m -th expert, and $p_g(z_m = 1|\mathbf{X}, \theta_{g,m})$ is the probability of that input being assigned to the m -th expert using an input sensitive gating network; in both cases θ represents the parameters of the mixture and gate distributions.

For simplicity and to reduce complexity of the experts we choose linear regression with constant offset $\mathbf{Y} = \beta\mathbf{X} + \alpha$ as our expert model (this is a simple generalization of the linear regression model described in Section 3.8.1), which allows us to solve for the parameters $\theta_{e,m} = \{\beta_m, \alpha_m, \Sigma_m\}$ analytically using the weighted linear regression. The expert model can be written as follows:

³Notice that here \mathbf{X} is the variable we are conditioning on and \mathbf{Y} is the variable we are trying to infer; opposite is true for the notation in Section 3.8.2.

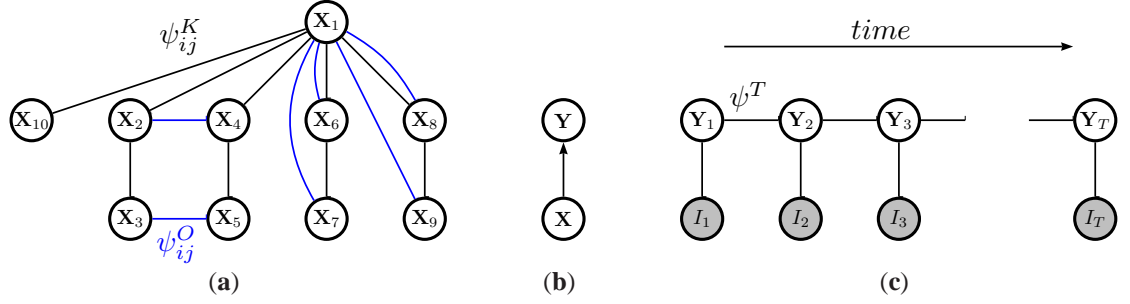


Figure 6.8: **Hierarchical inference.** Graphical model representation of the hierarchical inference process; (a) illustrates the 2D body model used for inference of the 2D pose at every frame, with kinematic constraints marked in black, and occlusion constraints in blue, and (c) the Hidden Markov Model (HMM) used for inferring and tracking the state of the 3D body, \mathbf{Y}_t , over time $t \in [1, \dots, T]$, using the hierarchical inference proposed, in which proposals for each node, \mathbf{Y} , are constructed from 2D body pose \mathbf{X} using the model in (b).

$$p_e(\mathbf{Y}|\mathbf{X}, z_m = 1, \theta_{e,m}) = \frac{1}{\sqrt{(2\pi)^{d_Y} |\Sigma_m|}} \exp^{-\frac{1}{2} \Delta_m^T \Sigma_m^{-1} \Delta_m}, \quad (6.14)$$

where d_Y is the dimensionality of the 3D pose \mathbf{Y} , β_m and α_m regression parameters, Σ_m is the covariance of the kernel regressor, and

$$\Delta_m = \mathbf{Y} - \beta_m \mathbf{X} - \alpha_m. \quad (6.15)$$

Pose estimation is a high dimensional and ill-conditioned problem, so simple least squares estimation of the linear regression matrix parameters typically produces severe over-fitting and poor generalization. To reduce this, we add smoothness constraints on the learned mapping. We use a damped regularization term $R(\beta) = \lambda \|\beta\|^2$ that penalizes large values in the coefficient matrix β , where λ is a regularization parameter (*a.k.a.* ridge regression). Larger values of λ will result in overdamping, where the solution will be underestimated, small values of λ will result in overfitting and possibly ill-conditioning. Since the solution of the ridge regressors is not symmetric under the scaling of the inputs, we normalize the inputs $\{x_1, x_2, \dots, x_N\}$ by the standard deviation in each dimension respectively before solving⁴.

The weighted ridge regression solution for the parameters β_k and α_k can be written in matrix notation as follows,

$$\begin{bmatrix} \beta_m \\ \alpha_m \end{bmatrix}^T = \begin{bmatrix} \mathcal{D}_{\mathbf{X}}^T \text{diag}(Z_m) \mathcal{D}_{\mathbf{X}} + \text{diag}(\lambda) & Z_m \\ Z_m^T & Z_m^T Z_m \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{D}_{\mathbf{X}}^T \\ Z_m^T \end{bmatrix} \text{diag}(Z_m) \mathcal{D}_{\mathbf{Y}}, \quad (6.16)$$

where $Z_m = [z_m^{(1)}, z_m^{(2)}, \dots, z_m^{(N)}]^T$ is the vector of ownership weights described later in the section and $\text{diag}(Z_m)$ is diagonal matrix with Z_m on the diagonal; $\mathcal{D}_{\mathbf{X}} = [x_1, x_2, \dots, x_N]$ and $\mathcal{D}_{\mathbf{Y}} = [y_1, y_2, \dots, y_N]$ are

⁴To avoid problems with 2D and 3D angles that wrap around at 2π , we actually regress the $(\cos(\theta), \sin(\theta))$ representation for 2D angles and unit quaternion $\mathbf{q} = [q_x, q_y, q_z, q_w]^T$ representation for 3D angles. After the 3D pose is reconstructed we normalize the not-necessarily normalized quaternions to valid 3D rotations. Since quaternions also suffer from the *double cover* problem, where two unit quaternions correspond to every rotation, care must be taken to ensure that consistent parameterization is used.

vectors of inputs and outputs formed from the training data \mathcal{D} .

Maximization for the gate parameters can be done analytically as well. Given the gate model,

$$p_g(z_m = 1 | \mathbf{X}, \theta_{g,m}) = \frac{1}{\sqrt{(2\pi)^{d_{\mathbf{x}}} |\Lambda_m|}} \exp^{-\frac{1}{2}(\mathbf{X} - \mu_m)^T \Lambda_m^{-1} (\mathbf{X} - \mu_m)} \quad (6.17)$$

maximization of the gate parameters $\theta_{g,m} = (\Lambda_m, \mu_m)$ becomes similar to the mixture of Gaussians estimation, where

$$\mu_m = \frac{\sum_{n=1}^N z_m^{(n)} x_n}{\sum_{n=1}^N z_m^{(n)}} \quad (6.18)$$

$$\Lambda_m = \frac{1}{\sum_{n=1}^N z_m^{(n)}} \sum_{n=1}^N z_m^{(n)} [x_n - \mu_m][x_n - \mu_m]^T \quad (6.19)$$

and $z_m^{(n)}$ is the estimated ownership weight of the example n by the expert m estimated by expectation

$$z_m^{(n)} = \frac{p_e(y_n | x_n, z_m^{(n)} = 1, \theta_{e,m}) p_g(z_m^{(n)} = 1 | x_n, \theta_{g,m})}{\sum_{j=1}^M p_e(y_n | x_n, z_j^{(n)} = 1, \theta_{e,j}) p_g(z_j^{(n)} = 1 | x_n, \theta_{g,j})}. \quad (6.20)$$

The above outlines the full EM procedure for the MoE model. We learn MoE models for two classes of actions: walking and dancing. Examples of the ground truth 2D query poses with corresponding expected 3D body poses can be seen in Figure 6.9 (a) and (b) respectively. Similar to [3, 4] we initialize the EM learning by clustering the output 3D poses using the K-means procedure. We learn a single conditional MoE model that we use for all view-based 2D pose estimates. We experimented with learning of view-based conditional MoE models, but they tended to have artifacts at view boundaries and suffered from overfitting.

Implementation Details

Instead of learning the full conditional model $p(\mathbf{Y} | \mathbf{X})$, we learn two independent models $p(\Gamma | \mathbf{X})$ and $p(\Theta | \mathbf{X})$ one for the pose of the 3D body $p(\Theta | \mathbf{X})$ given the 2D body pose \mathbf{X} , and one for the global orientation of the body $p(\Gamma | \mathbf{X})$. The reasoning for this is twofold. First, this partitions the learned mapping into a fully camera-independent model for the pose $p(\Theta | \mathbf{X})$, and the more specific camera-dependent model for the orientation of the body in the world $p(\Gamma | \mathbf{X})$. Second, we found that the optimal damping coefficient is significantly different for the two models, therefore imposing a single joint conditional model (and hence a single coefficient) would result in somewhat larger reconstruction error. Estimation of the depth $p(\Xi | \mathbf{X})$ is done analytically (using simple regression) by considering the estimated position and overall scale of the 2D body.

6.5 Tracking in 3D

Once the distribution for the 3D body pose at every frame is inferred using the conditional MoE model described, we can incorporate temporal constraints to regularize the individual 3D pose estimates by tracking. We exploit the relatively standard undirected variant of the Hidden Markov Model (HMM) shown in Figure 6.8 (c). To infer the state of \mathbf{Y}_t at every frame t given the temporal potential $\psi^T(\mathbf{Y}_t, \mathbf{Y}_{t+1}) = \mathcal{N}(\mathbf{Y}_t - \mathbf{Y}_{t+1}; 0, \Sigma_T)$, with learned covariance matrix Σ_T , we use the same inference framework of PAM-PAS. Unlike many competing approaches, we allow the model to optimize the pose estimates not only forward

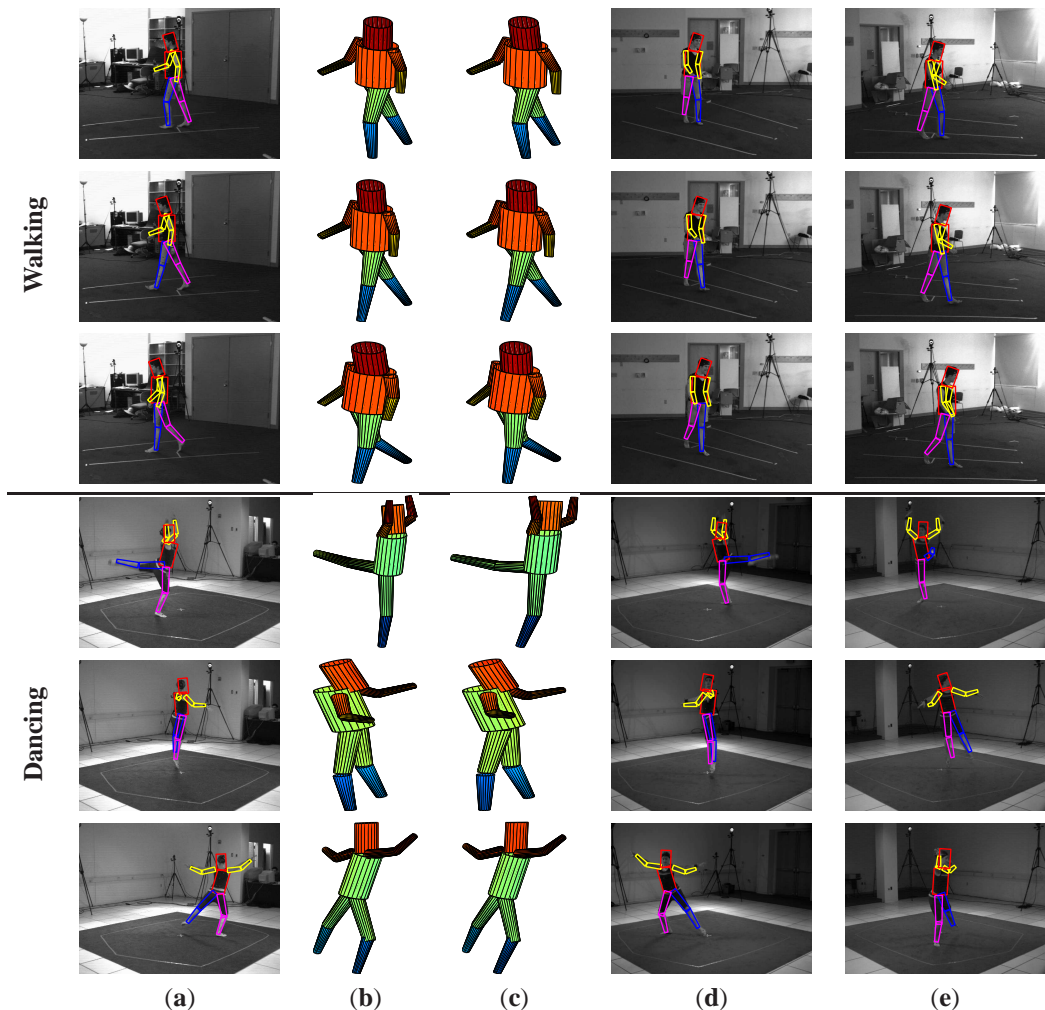


Figure 6.9: **Proposed 3D pose.** (a) Synthetic query 2D body pose, obtained from projected motion capture data; (b) expected 3D pose produced by the learned action-specific Mixture of Experts (MoE) model. (c) Ground-truth 3D body pose; (d) and (e) illustrate the projection of the expected 3D pose shown in (b) onto the original and one alternative image view.

but also backward in time in a batch (this amounts to temporal smoothing).

The likelihood, $\phi(I_t|Y_t)$, of observing the 3D pose Y_t at time t given image evidence I_t is defined in terms of Chamfer distance of the projected pose Y_t to the silhouettes and edges obtained from I_t using standard techniques.

6.6 Experiments

In this section we present a set of quantitative and qualitative experimental results for testing various stages of our hierarchical inference framework, as well as the framework as a whole. We first test how well we can recover the 2D pose independently in each frame using our *occlusion-sensitive loose-limbed body model* in Section 6.6.1; we then analyze how well our discriminative MoE model can predict the 3D pose from the 2D poses recovered at every frame, in Section 6.6.2; finally, we briefly explore the benefits of adding temporal

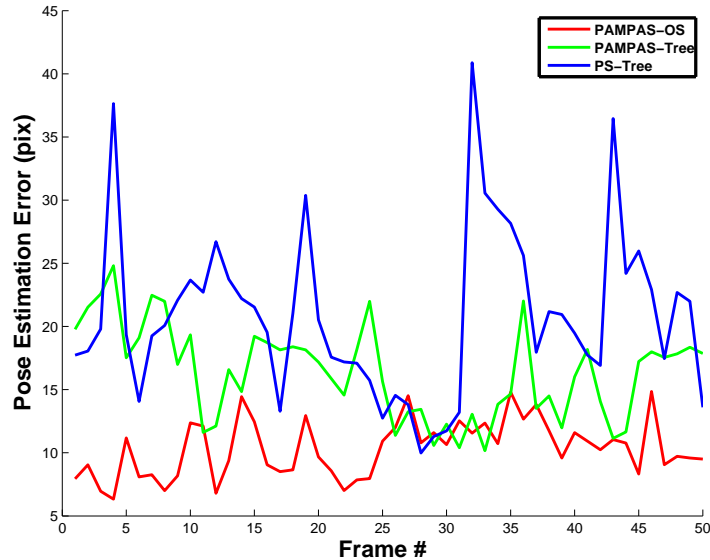


Figure 6.10: **Quantitative performance evaluation of 2D pose estimation.** Mean error of the joint locations for each frame of 50 frame image sequence with ground truth [197]. For the description of the metric see text.

consistency (tracking) into our hierarchical framework in Section 6.6.3.

6.6.1 Monocular 2D Pose Estimation

We learned occlusion-sensitive models for 8 discrete views of a person including frontal, side and 3/4 views. For each view we assume the depth ordering of the body parts is known. The kinematic constraints between parts were learned from projected motion capture data. In all experiments the likelihood uses a combination of silhouette and color/intensity information (assuming independence). Color was primarily used to achieve robustness in the cases where silhouettes were ambiguous or unreliable in localizing a given part. For the silhouette likelihood we used the pictorial structures type model and learned $p_{1,FG}(I_u = 1) = q_1$ and $p_{2,FG}(I_s = 1) = q_2$ using the procedure described in [59]. Similar to [59] we assumed that $p_{3,FG}(I_r = 1) = 0.5$. For the color/intensity likelihood we learned a kernel density model for each part and the background.

For frontal views, the lack of self occlusion means that tree based approaches will usually perform well. Consequently we focus on the more challenging side-views containing occlusion. We quantitatively compare our approach (PAMPAS-OS) to leading tree-based methods using 50 frames from the Brown ground truth sequence, obtained similarly to the HUMANEVA-I dataset described in Section 5.7.1. Unlike HUMANEVA-I, the dataset used in this chapter contains images from 4 synchronized greyscale cameras (instead of 7 in HUMANEVA-I); however, we only employ images from one camera (BW1) for inference. Additional description of the data used in this chapter will be given in the next section. To evaluate performance of our 2D method, we extend the error metric presented in Chapter 5. As before, the proposed metric computes the average distance error between a set of 15 virtual marker locations corresponding to the joints. However, since our pose in this case is in 2D, the distance is computed in the image plane, instead of the world; the resulting error is in (*pixels*).

For comparison we implemented two tree-based methods: pictorial structures (PS-Tree) [59] and a variant

	Strong Prior	Discrete State	Mean Error (<i>pixels</i>)	Std. of Error (<i>pixels</i>)
PAMPAS-OS	No	No	10.33	2.25
PAMPAS-Tree	No	No	16.40	3.67
PS-Tree	No	Yes	20.84	6.64
PS-Tree [122]	No	Yes	13.79	3.99
LBP [122]	Yes	Yes	12.00	3.99
Factor [122]	Yes	Yes	6.42	1.55

Figure 6.11: **Overall performance comparison of 2D pose estimation.** Performance of the occlusion-sensitive inference compared with two tree-based algorithms implemented by us. We also compare to the results reported by [122] on the same image sequence.

of our approach that does not model occlusions (PAMPAS-Tree) by simply removing the occlusion constraints from our model. Figure 6.10 shows the mean error for 15 markers at every frame for the three methods; the statistics, for comparison, are shown in Figure 6.11. Following [122] we deal with the left/right ambiguity by switching the left/right limbs and reporting the interpretation with a smallest error. Notice, that for the results reported in Figures 6.10 and 6.11 we are estimating the pose independently at every frame (*i.e.* only doing pose estimation, not tracking).

Our occlusion-sensitive inference approach outperforms pictorial structures by 50% (25% for the implementation in [122]¹). We found that occlusion-reasoning accounts for a 37% performance gain over the simple PAMPAS-Tree method. According to the published literature [122] our approach also outperforms max-product loopy-BP, but does not do as well as the common-factor model (Factor) presented in [122]. This is not surprising, since the common-factor model uses a walking prior learned for this data. Our approach does not assume a strong prior on the motion or style, instead it only encodes weak prior⁵ on the relative position of neighboring limbs.

Figure 6.12 illustrates the behavior of PS-Tree, PAMPAS-Tree and PAMPAS-OS on a few frames of the sequence. As expected we observed many failures in the pictorial structures model due to the overlapping parts (*e.g.* see Figure 6.12 (a)). PAMPAS-Tree, not surprisingly had similar modes of failure while the occlusion-sensitive PAMPAS-OS does a better job of explaining the image evidence (as can be seen from the results in Figure 6.11 and quantitative comparison in Figure 6.10).

In addition to the quantitative experiments we also ran our model on less structured scenarios from TV and movies for which strong prior models will typically not work. Figure 6.13 illustrates two representative results. In both cases, camera motion makes background subtraction difficult. Crude background subtraction was obtained using homographies estimated between 2 frames sufficiently far apart in time (using the code from <http://www.robots.ox.ac.uk/~vgg/>). Estimated homographies allow us to compensate for the camera motion and use frame differencing to obtain very rough estimates for the foreground silhouette (see Figure 6.13 (c)). Color likelihoods were defined as in [170].

Our current un-optimized implementation of PAMPAS-OS in Matlab takes roughly 5 minutes for message passing, and 1.5 minutes for belief estimation per frame. The occlusion constraints account for a 43%

¹Our independent implementation of PS-Tree [59] resulted in somewhat larger error than reported in [122].

⁵Consequently, while in this thesis we advocate the use of weak prior models due to their generality, the models introduced here do not prevent the use of strong priors. To the contrary, the loose-limbed models we introduced can easily be extended to include priors of the form explored in [122] to boost performance in specific domains.

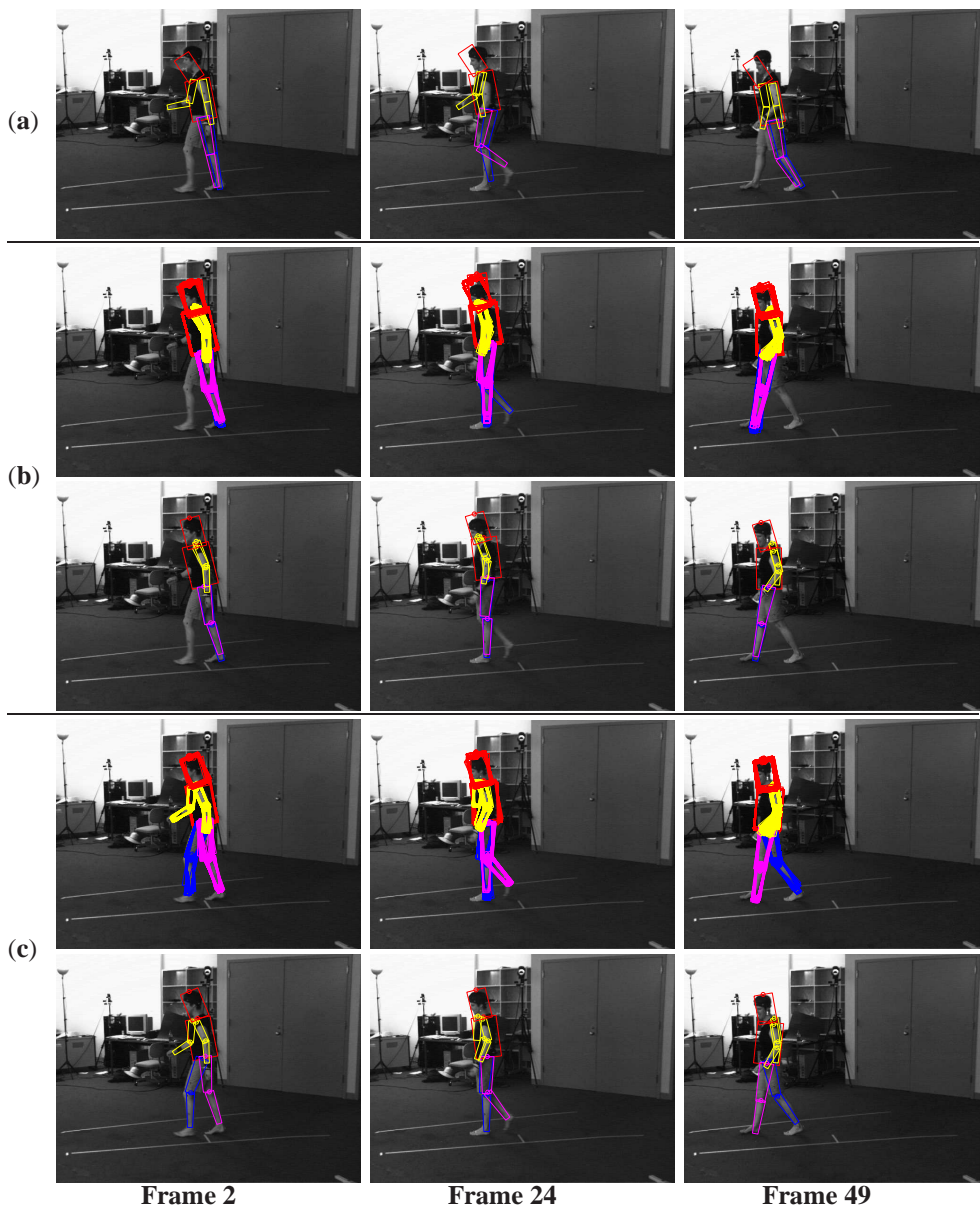


Figure 6.12: **Visual performance evaluation of 2D pose estimation.** (a) MAP estimates for the tree-based implementation of pictorial structures on three frames from our test sequence. Performance of occlusion-insensitive and occlusion-sensitive PAMPAS is shown in (b) and (c) respectively. The top rows show 100 samples from the marginal distribution at every node (belief) after 5 iterations of BP, and bottom rows the weighted mean computed over those samples. BP was run using 100 particles which resulted in the $N = 800$ Gaussian kernel mixtures for the messages.

overhead over PAMPAS-Tree.

6.6.2 Monocular 3D Pose Estimation

In previous section we tested the performance of one of the key components of our hierarchical framework, that allows us to reliably recover the 2D pose of the person from monocular images (independently at every frame). We showed that our occlusion-sensitive model performs better than other methods tested. In this

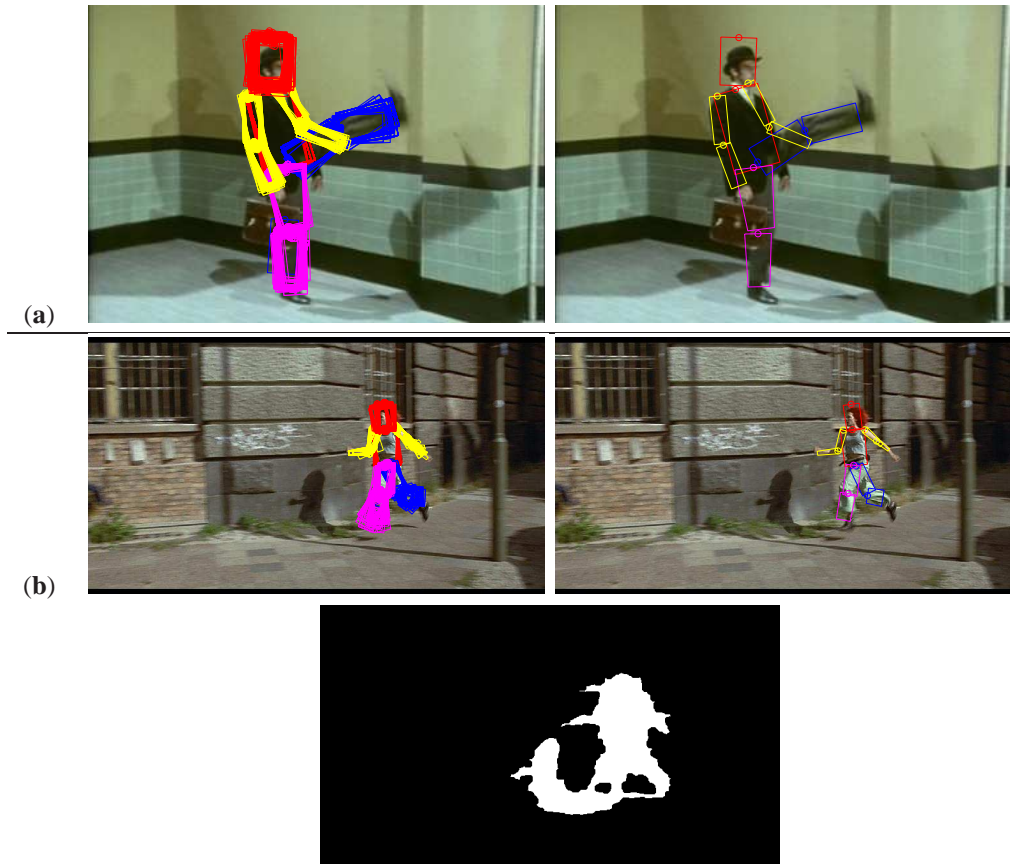


Figure 6.13: **Occlusion-sensitive reasoning in movies**. Results on frames from TV/films. Left column in (a) and (b) shows 100 samples from the marginal distribution (belief) after 3 iterations of BP, and right column shows the weighed mean pose. In both cases very rough and noisy background subtraction was obtained by estimating homographies between 2 frames from the sequence sufficiently far apart in time. Example of the rough background subtraction obtained in this way for the image in (b) is illustrated in the last row.

section, we investigate the second major component of our framework, that allows us to infer the 3D pose of the person from 2D pose estimates; for now, still from single monocular images. We first conduct a set of synthetic experiments that allows us to quantitative test how well our Mixture of Experts (MoE) model can recover 3D pose in general, assuming the 2D pose is known⁶. We then show how this learned MoE model can be used to recover the 3D pose from the real 2D poses obtained in the previous section.

Datasets. For all experiments presented in this section we used two datasets that exhibit two different types of actions: **walking** and **dancing**. Both datasets contain a number of motion capture examples used for training, and a single synchronized motion capture example with multi-view video used for testing (the same as in previous section). Video was captured using 4 stationary grayscale cameras at 60 Hz, and 3D pose was captured using a Vicon system at 120 Hz. The motion capture (mocap) was aligned to video and sub-sampled to 60 Hz, to produce synchronous video/mocap streams. All cameras were calibrated using standard calibration procedures. **Walking** dataset [197] contains 4587 training and 1398 testing poses/frames; **dancing**: 4151 training and 2074 testing poses/frames.

⁶We use projected motion capture data in lieu of known 2D poses.

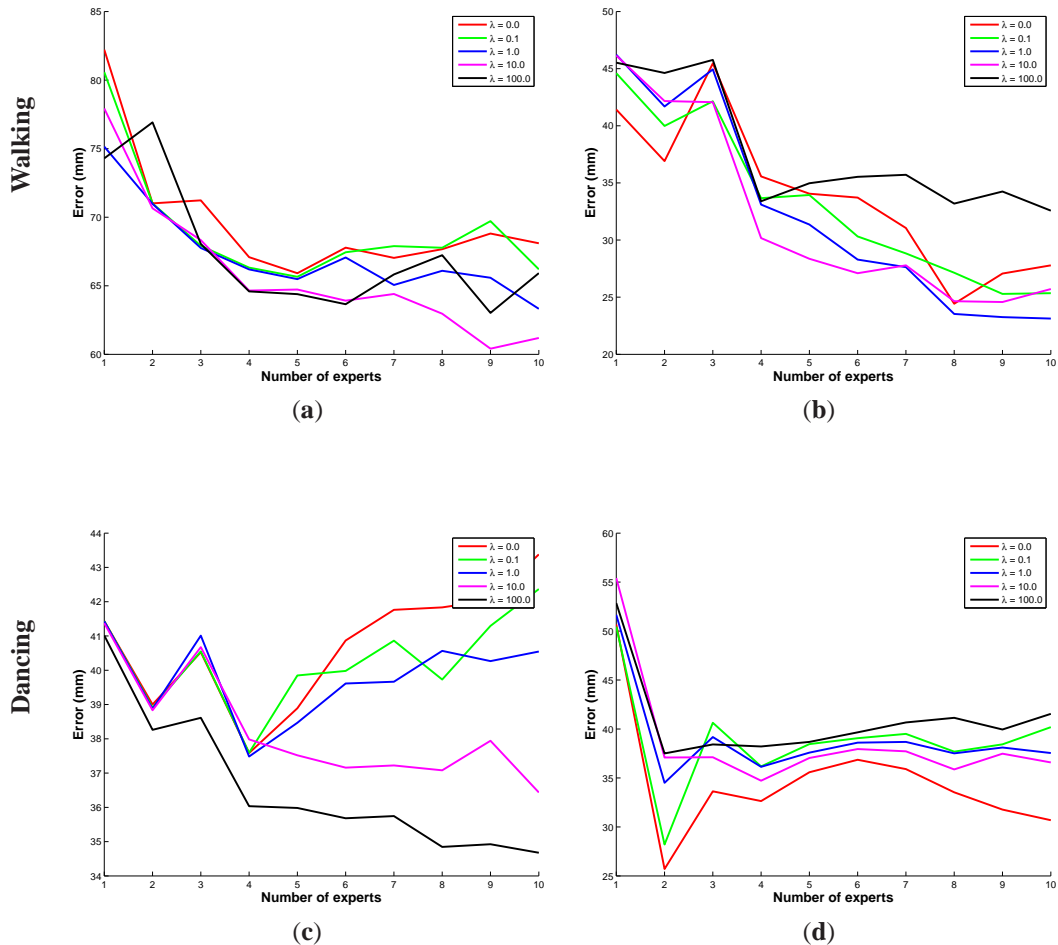


Figure 6.14: **Learning parameters for conditional MoE models.** Quantitative evaluation of various parameter choices for action-specific **dancing** (a),(b) and **walking** (c),(d) conditional MoE models, where $p(\mathbf{Y}|\mathbf{X}) = p(\Xi|\mathbf{X})p(\Gamma|\mathbf{X})p(\Theta|\mathbf{X})$. Plots separately illustrate the error computed by comparing the expected 3D pose in (a) and (c), $\mathbb{E}[p(\Theta|\mathbf{X})]$, and global orientation (view) in (b) and (d), $\mathbb{E}[p(\Gamma|\mathbf{X})]$, to the ground truth 3D poses. Average error across 1398 frames for **walking** and 2074 frames for **dancing** is computed and illustrated as a function of parameters explored. Due to stochastic nature of MoE learning, the error is also averaged over 4 trained instances of the MoE model learned with the specified set of parameters. Different number of mixture components $M \in [1, \dots, 10]$ are tested, and range of damping coefficients $\lambda \in \{0, 0.1, 1, 10, 100\}$ is explored. In both **walking** and **dancing**, it is clear that there is benefit in using large number of mixture components (> 5), and a moderate value for λ .

Synthetic Experiments. We learn two action-specific Mixture of Expert (MoE) conditional models $p(\mathbf{Y}|\mathbf{X})$. For each of the action types we first look at how sensitive our learned mapping is to the parameters of the model (*i.e.* the number of mixture components, and the regularization term λ). The results can be seen in Figure 6.14. To quantitatively evaluate the performance we use the error measure introduced in Chapter 5. To reiterate, the error is computed by choosing 15 virtual markers corresponding to joints and “ends” of limbs, and computing an expected absolute distance in (*mm*) over all these virtual markers. Once the optimal set of parameters was chosen, the resulting MoE models were ran on the synthetic test data (see sample results in

Figure 6.9) and the error for the reconstructed 3D poses⁷ was analyzed (see Figure 6.15).

We consider 3 variants of the base error metric: **(Pose)** 3D body pose error computed by first aligning the global coordinate frames of the reconstructed and ground truth body, **(View)** global body orientation error computed by first aligning the reconstructed pose/articulation with the ground truth pose/articulation, and **(View+Pose)** an overall error for the reconstructed state \mathbf{Y} . The alignment for **(View)** and **(Pose)** is done by simply using the ground truth portion of the 3D pose for the corresponding components of the state vector. The key observation is that **walking**, being considerably simpler of the two action types, can be recovered significantly better (with 50% less error), than the more complex **dancing**. The peaks in the error in both cases often correspond to singular or close to singular instances where foreshortening in the pose of 2D limbs for example is severe; similar singularities arise where the pose of the entire body is close to lateral or frontal (with unbent arms and legs), where the view itself is singular (*i.e.* the model can't easily distinguish between left and right, or front and back, orientation of the body when singularities occur).

Real Experiments. We can also apply the learned **walking** Mixture of Experts (MoE) model to the real 2D poses obtained in Section 6.6.1. This allows us to recover the full 3D pose automatically, using the proposed hierarchical method, from single monocular image; results are illustrated in Figure 6.16. The 3D poses look very reasonable, however, the right arm that is occluded in 2D in most frames is often miss estimated. Notice, that the prior over walking postures, implicitly embedded into the action-specific **walking** MoE model, allows the conditional model to correct some of the errors made by the 2D pose estimation component.

6.6.3 Monocular 3D Tracking

In Sections 6.6.1 and 6.6.2 we experimented with two major components of our hierarchical inference method that, in conjunction, allow the inference of 3D pose from monocular single images. In this section, we briefly illustrate how these individual 3D pose estimation results can be regularized by Bayesian temporal inference (Figure 6.17).

Our experiments with hierarchical single-frame method, illustrated in Section 6.6.2, show that while the method can reasonably recover the 3D pose, it cannot reliably resolve the left/right ambiguity of the body. In other words, the inference method often switches the identity of the two legs from frame to frame (see frames 17–19 in Figure 6.18); in general, it is ambiguous to reason about left/right leg identity from the lateral view images illustrated in Figure 6.16. In addition, the recovered poses are also often noisy (jittery). We found temporal smoothing, results of which are illustrated in Figure 6.17, to be useful in regularizing this intra-frame variability in pose. Consequently, temporal smoothing produces more coherent tracks over time as illustrated in Figure 6.18.

6.7 Conclusion and Discussion

The automatic estimation of human pose and motion from monocular image data remains a challenging problem. Here we have proposed a framework to address this problem that uses hierarchal Bayesian inference to go from crude body part detections to a distribution over 3D body pose. We make modest assumptions about the availability of noisy body part detectors and a reasonable image likelihood model.

⁷Supplementary videos are available from <http://www.cs.brown.edu/people/lis/>.

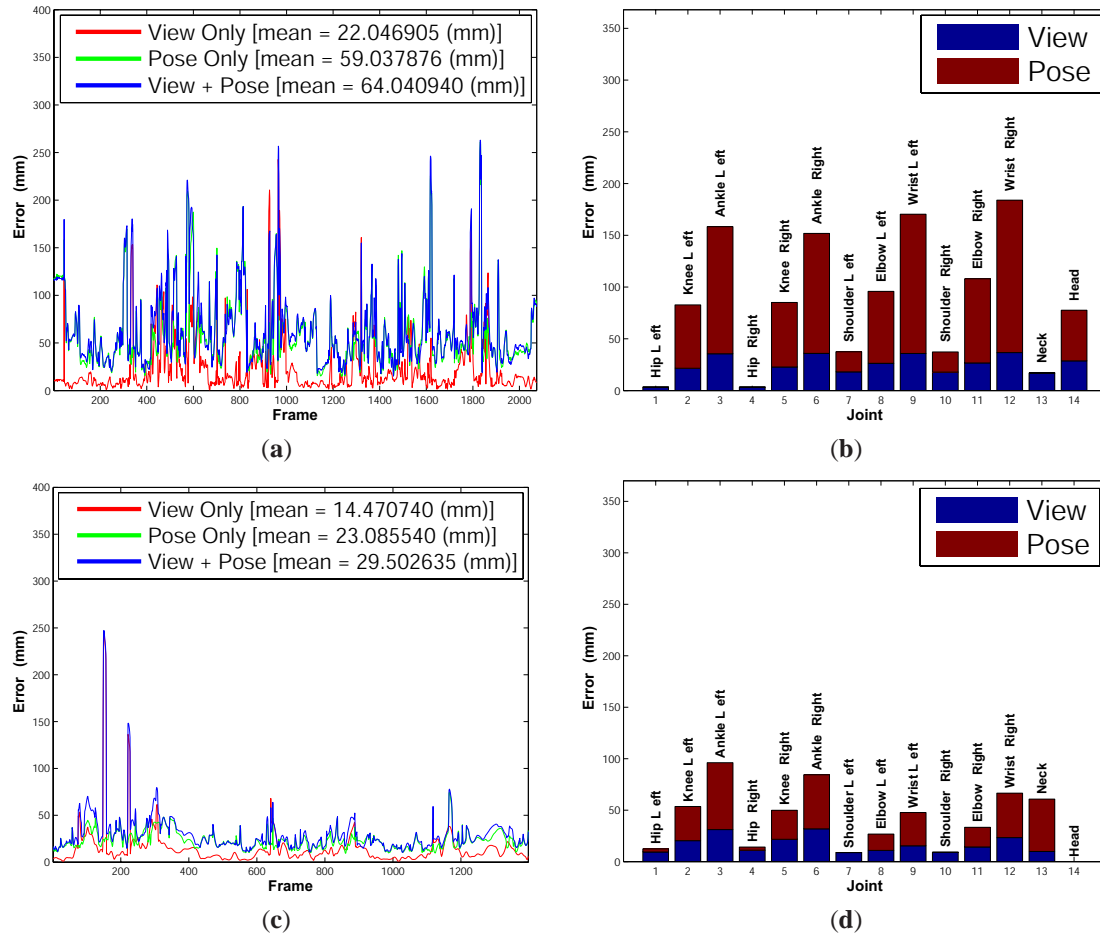


Figure 6.15: **Quantitative evaluation of action-specific conditional model.** The model, $p(\mathbf{Y}|\mathbf{X}) = p(\Xi|\mathbf{X})p(\Gamma|\mathbf{X})p(\Theta|\mathbf{X})$, is tested by comparing the expectation to ground truth data for two classes of motion. Per-frame error for the reconstructed 3D pose Θ , global orientation Γ , and the full 3D state of the body \mathbf{Y} are shown for **dancing** in (a) and **walking** in (c); the average per-joint error as compared to the ground truth is shown in (b) and (d) respectively.

We use belief propagation to infer 2D limb poses that are consistent with the human body model. Note that we make no strong assumptions about the prior poses of the body. Our approach extends recent work on inferring 3D body models from 2D silhouettes by using the inferred 2D articulated model instead. This provides a richer representation which reduces ambiguities in the 2D to 3D mapping. We also show that the 3D pose proposals can be used in a tracking framework, that can further regularize the 3D pose estimates.

As part of this hierarchical inference strategy, we also introduce a novel approach for articulated 2D body pose estimation that uses occlusion-sensitive local image likelihoods that approximate the global likelihood by accounting for occlusions and competing explanations of image evidence by multiple parts. We model occlusion relationships between parts explicitly by introducing two sets of per-pixel hidden binary variables for each part. Intuitively these variables model which pixels are explained by which parts when multiple parts project to the same image regions. The resulting occlusion reasoning involves interactions between non-adjacent parts which introduces loops in the graphical model representation of the body. To achieve tractable real-valued inference in such a graph, we also introduced an extension to the approximate belief

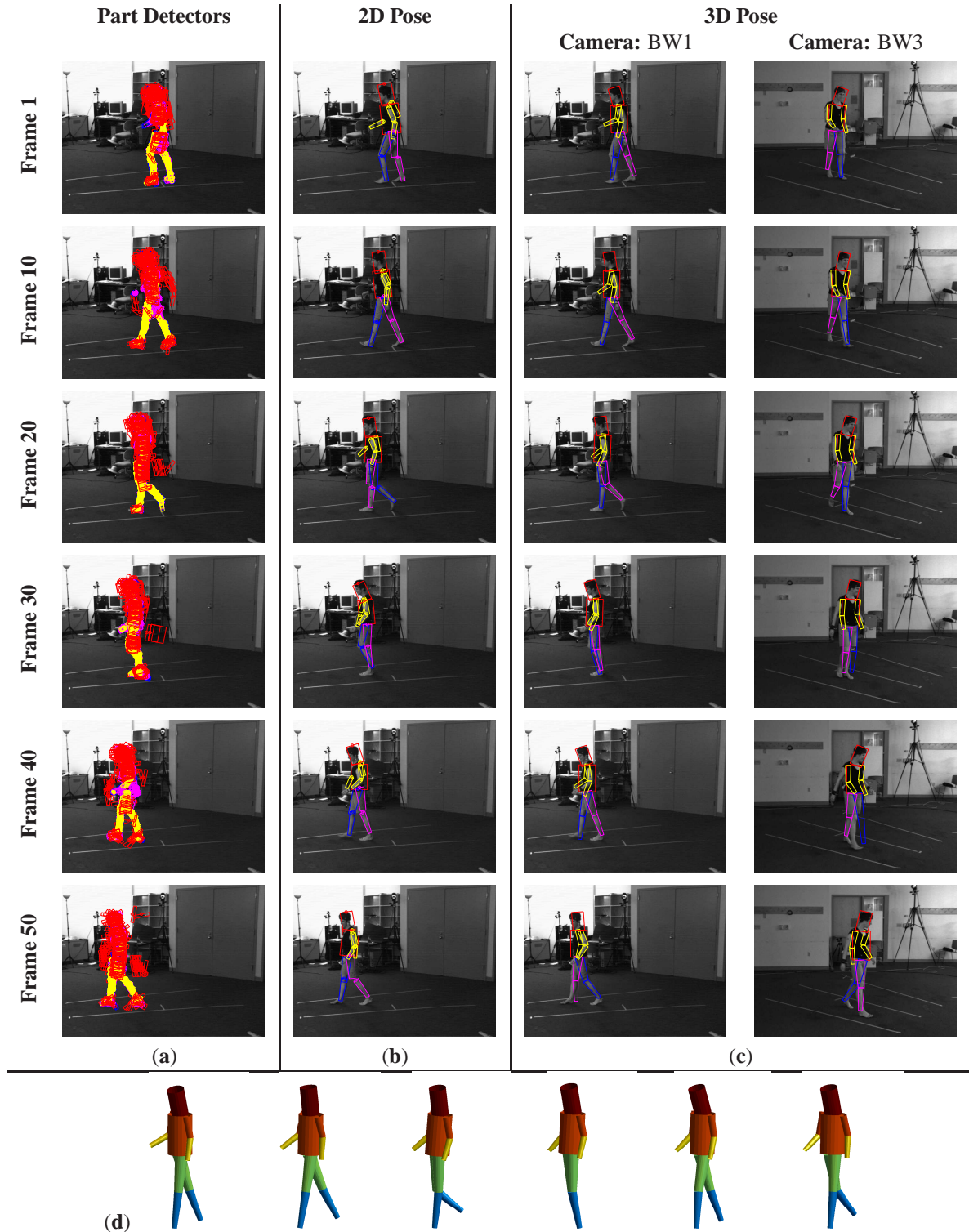


Figure 6.16: **Hierarchical 3D pose estimation.** (a) bottom-up proposals for the limbs, (b) most likely sample from the marginals for each limb after 2D pose is estimated by PAMPAS-OS (see Section 6.6.1 for more details), and (c) most likely 3D pose obtained by propagating 2D poses through a conditional $p(\mathbf{Y}|\mathbf{X})$ model (also rendered as a synthetic 3D character in (d)). In (c) the recovered 3D model is projected in to the same view as in (b) as well as an additional view (not used in the inference) to illustrate the errors in 3D pose estimation (that may not be observed otherwise).

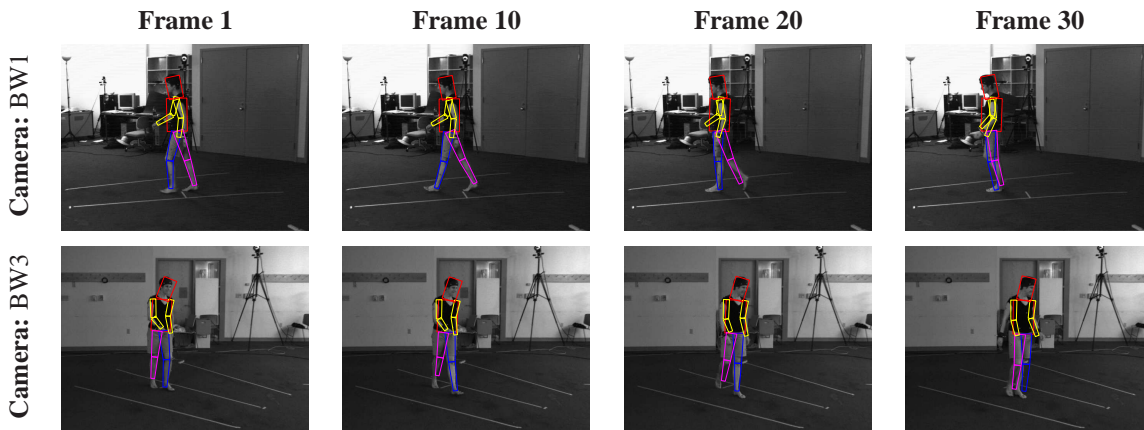


Figure 6.17: **Monocular tracking in 3D**. Tracking based on the 3D proposals (Fig. 6.16) illustrated at 10 frame increments. The 3D poses are projected into images for clarity; top row shows the projections into the view used for inference, the bottom row projections into a different view not available to the hierarchical inference framework.

propagation inference algorithm (PAMPAS) that takes into account, and analytically marginalizes over, the hidden occlusion variables of our model.

We quantitatively compare our 2D pose estimation approach to two state-of-the-art algorithms using tree-structured kinematic models, as well as to published results in the literature. The proposed approach performs favorably and solves the problem of competing models that tend to match multiple body parts to the same image evidence without the addition of strong priors. Explicit reasoning about occlusions helps prevent this from happening in our case. Experimental results illustrate that our model has pose error at least 25%

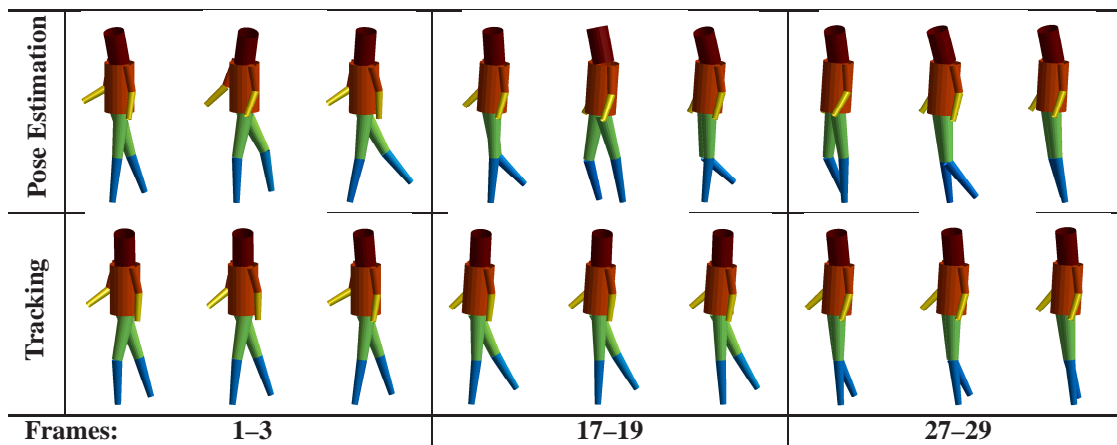


Figure 6.18: **Comparison of monocular 3D pose estimation with tracking**. Illustrated is the comparison between 3D pose estimation (**top**), obtained independently at every frame using the proposed hierarchical framework, and temporal tracking (**bottom**), obtained by smoothing the distribution over the 3D poses from (**top**). The results shown correspond to results illustrated otherwise in Figures 6.16 and 6.17 respectively. The 3D model in the inferred most likely pose is shown, for convenient, in a canonical view not corresponding to any of the real cameras. Notice, that while pose estimation is relatively reliable, it exhibits two unfavorable behaviors: (i) jitter from frame to frame and (ii) inconsistencies in identity of left and right leg (see frames 17-19); tracking smooths out these artifacts by incorporating information over time, resulting in smoother motion overall.

lower than tree-structured models. We also show that our approach performs favorably in complex scenarios, where strong assumptions about the kinematic motion of the body are not appropriate. We also quantitatively compare the overall performance of our hierarchical framework, used to infer the 3D pose and track human motion from monocular imagery.

CHAPTER 7

Summary and Discussion

In this thesis we introduced a novel class of models and corresponding inference algorithms that are able to address a variety of common problems in object localization, pose estimation and tracking. For the large portion of this thesis we concentrated on the challenging class of articulated objects (*i.e.* people). Dealing with people is challenging, particularly because of variation in appearance and articulations; furthermore, the pose of the person often requires representations that are high-dimensional and that must deal with ambiguous image observations. Reasoning about people and their pose in images, is popular however, due to the vast number applications in animation, surveillance, biomechanics and human computer interaction.

Instead of attempting to battle the dimensionality of the state-space and complexity of motion directly, we formulate the problem of pose estimation and tracking as one of inference in a graphical model. The nodes in this graph correspond to parts of the body and edges to kinematic, inter-penetration and occlusion constraints imposed by the structure of the body and the imaging process. This model, which we call a *loose-limbed body model*, allows us to infer the 3D pose of the body effectively and tractably from multiple synchronized views; or a 2D pose of the body from a single monocular image, in time linear in the number of articulated parts. Unlike previous decentralized models, we work directly with continuous variables, and use variants of Particle Message Passing (PAMPAS) for inference.

In addition, we also introduced hierarchical models for both articulated and generic object reasoning. In the case of generic objects, hierarchy facilitates tractable inference by ensuring that the temporal constraints are only propagated on the object level and not at the level of individual parts. In the case of articulated objects, hierarchy also mediates the complexity of the spatial inference, by allowing the model to first infer the 2D pose of the body in the image plane, then infer the 3D pose from the 2D body pose estimates and lastly apply the temporal continuity (tracking) at the 3D pose level. This leads to two important benefits: (1) the hierarchical model helps to reduce the depth and projection ambiguities by looking at a full 2D body pose rather than the pose of individual limbs, and (2) it gives a modular, tractable, and fully probabilistic solution that allows inference of 3D pose from a single monocular image in an unsupervised fashion.

In all cases we have shown both qualitatively and quantitatively that the models introduced perform as well, or better, than other state-of-the-art methods.

7.1 Future Work

While the models we introduced are effective and address a number of common problems in both articulated and rigid object motion estimation, there still a number of issues that must be addressed in the future to make these models widely applicable for large categories of objects.

7.1.1 Faster Inference Algorithms

Particle Message Passing (PAMPAS) and various extensions thereof, that have been introduced in this thesis, while tractable and have linear complexity, still are too slow to allow real-time (30 frames per second) processing on current hardware. The main computational bottleneck, is that sampling from products of messages, represented by kernel densities with many mixture components, is computationally expensive. Reducing the number of mixture components in the representation of messages would lead to significant computational speedup of PAMPAS. On an intuitive level, while the kernel densities that we are using to approximate messages are complex, the underlying distributions that they are approximating are often, in comparison, relatively simple (particularly after BP has converged or is close to convergence).

To speed up inference there have been recent attempts to develop faster Non-parametric Belief Propagation (NBP) inference algorithms by automatically reducing representation of the message to a number of prominent modes estimated by Mean-shift [78]. The results have been shown to be orders of magnitude faster than simple NBP or PAMPAS, for tracking. Our preliminary experiments (not described in this thesis), have shown that this approach indeed achieves significant speedups for simple examples where messages are close to convergence (*i.e.* have few modes). For pose estimation, where the messages are often initialized relatively far from the true solution, the process of reducing the number of mixture components in the representation, takes longer than the inference itself. A simple explanation for this is that the number of modes in a message in this case is typically significantly larger (tens instead of one or two that are often observed in tracking). We believe that hybrid algorithms that reduce message representation complexity only when possible, is the next logical step in producing tractable inference algorithms for this class of models.

Other approaches that we believe may be useful in reducing the complexity of inference are hybrid Markov Chain Monte Carlo methods, that can be used to replace the pure Monte Carlo sampling engine of PAMPAS. Hybrid methods have been shown to achieve faster (orders of magnitude faster) inference in other domains [38], and we believe can be relatively easily adopted for the use in the PAMPAS framework.

7.1.2 Deeper Hierarchical Models

We found hierarchical models to be very effective in managing both computational and modeling complexity of problems addressed by this thesis. Currently, however, we restricted ourselves to models with relatively few (2 to 3) levels. In such models each level in the hierarchy has a pre-defined semantic structure. Deep hierarchical networks (*a.k.a.* deep belief networks) [82, 83] have been successfully developed and applied in other applications. In these deep networks, however, layers typically lack semantic interpretation as the number of layers grows and the layers themselves are learned automatically using unsupervised methods. We believe that, in the context of object modeling, particularly of articulated object modeling, slightly deeper hierarchies (than the ones presented in this thesis) can be developed that can both be useful and still maintain the semantic interpretation. For example, currently the interactions of different views and features are all

rolled into the likelihood function in our framework. Using additional layers in the hierarchical model, these interactions can be made explicit and perhaps better modeled. Our current likelihood model, for example, assumes independence across features and across views. A more explicit model can potentially model correlations between these variables. In particular, as the number of views increase, the observations become less and less independent. This is not currently handled by the models introduced in this thesis (nor much of related literature).

7.1.3 Learning of Model Structure

In this thesis we showed that continuous-state graphical models are effective means of modeling objects and drawing inferences about these objects, particularly pertaining to position and configuration of these objects in space. The models that we presented were built using the expert domain knowledge of the object class, that involved knowing and leveraging kinematic structure of the object with complexity of inference. The parameters of those models were learned in semi-supervised fashion, from motion capture data in the case of humans or hand annotated images in the case of vehicles. These models provide a very productive paradigm for object reasoning, due to the linear complexity that stems from their decentralized nature.

The problem of building these models automatically from unlabeled (or weakly labeled) data, however, is still largely unaddressed. In the context of Machine Learning, this problem is often referred to as graphical model *structure learning*. While it has been addressed in the context of some specific classes of graphical models, for example, in parametric Bayesian networks that have no interactions between hidden variables [200, 201], the case of general undirected graphical models with non-parametric continuous random variables is still largely unexplored. Continuous non-parametric models are considerably more expressive which makes model structure learning hard. To our knowledge, the only approach that addresses structure learning in general graphs that have both continuous and discrete variables was introduced by Bach and Jordan [12]. The ability to build these rich models automatically, however, is the key to making them widely applicable in the domain of generic object recognition.

In the context of articulated human motion, the ability to build models automatically would allow building of action-specific models that could potentially model higher order action-specific correlations between limbs. For example, in walking, there are well known correlations between upper and lower extremities and left and right sides of the body. Other motions may exhibit similar correlation patterns, induced by subtle hidden causes like gravity, balance, and/or intent. Building models that can automatically find, and account for, such correlations would undoubtedly lead to better models and performance.

7.1.4 Scene Parsing

One of the key advantages of using graphical models for modeling objects, beyond tractable inference, is the ability to combine different models in the context of probabilistic inference. We believe that one of the prominent directions of future research is to combine models of various objects (or multiple instances of the same model) for scene parsing and interpretation. Much like in the speech recognition community, context provided by other objects can be useful in constraining the object(s) of interest (*e.g.* [85]).

7.2 Conclusions

In this thesis we presented a novel class of methods that model people using rich decentralized probabilistic models. These models have a number of appealing advantages over the centralized models typically employed. The inference methods, that make use of the decentralized model structure for tractable inference, have also been introduced. In addition, we introduced a number of extensions to our basic *loose-limbed body* model, that allowed monocular inference and illustrated inference over simple generic objects (*e.g.* vehicles). The next challenge is take the methods introduced in this thesis and extend them for use with generic and possibly interacting objects. Among the challenges one would have to address, the most predominant are the unsupervised or semi-supervised learning of the model structure and faster (close to real-time) inference methods.

BIBLIOGRAPHY

- [1] A. Agarwal and B. Triggs. Recovering 3D Human Pose from Monocular Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 28, No. 1, pp. 44–58, January 2006.
- [2] A. Agarwal and B. Triggs. Monocular Human Motion Capture with a Mixture of Regressors, *IEEE Workshop on Vision for Human-Computer Interaction*, pp. 72–79, 2005.
- [3] A. Agarwal and B. Triggs. Learning to Track 3D Human Motion from Silhouettes. *International Conference on Machine Learning (ICML)*, pp. 9–16, 2004.
- [4] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 882–888, 2004.
- [5] S. Agarwal, A. Awan and D. Roth. Learning to detect objects in images via a sparse, part-based representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 26, No. 11, pp. 1475–1490, November 2004.
- [6] B. Allen, B. Curless, Z. Popovic and A. Hertzmann. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis, *Symposium on Computer Animation (SCA)*, pp. 147–156, 2006.
- [7] J. Amores, N. Sebe and P. Radeva. Fast Spatial Pattern Discovery Integrating Boosting with Constellations of Contextual Descriptors, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 769–774, 2005.
- [8] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers and J. Davis. SCAPE: Shape Completion and Animation of People, *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 24(3):408–416, 2005.
- [9] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, *IEEE Transactions on Signal Processing*, Vol. 50, Issue 2, pp. 174–188, February 2002.
- [10] V. Athitsos, J. Wang, S. Sclaroff and M. Betke. Detecting Instances of Shape Classes That Exhibit Variable Structure, *European Conference on Computer Vision (ECCV)*, Vol. 1, pp. 121–134, May 2006.
- [11] V. Athitsos, J. Alon, S. Sclaroff and G. Kollis. BoostMap: A Method for Efficient Approximate Similarity Rankings, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 268–275, June 2004.
- [12] F. R. Bach and M. I. Jordan. Learning graphical models with Mercer kernels, *Advances in Neural Information Processing Systems (NIPS)*, Vol. 15, 2003.

- [13] A. Balan, L. Sigal, M. J. Black, J. Davis and H. Haussecker. Detailed Human Shape and Pose from Images, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [14] A. Balan, L. Sigal and M. J. Black. A Quantitative Evaluation of Video-based 3D Person Tracking, *IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 349–356, October 2005.
- [15] A. Banerjee, I. S. Dhillon, J. Ghosh and S. Sra. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions, *Journal of Machine Learning Research*, Vol. 6, pp. 1345–1382, 2005.
- [16] H. Bay, T. Tuytelaars and L. van Gool. SURF: Speeded Up Robust Features, *European Conference on Computer Vision (ECCV)*, Vol. 1, pp. 404–417, 2006.
- [17] S. Belongie, J. Malik and J. Puzicha. Matching shapes, *International Conference on Computer Vision (ICCV)*, pp. 454–461, 2001.
- [18] J. Berclaz, F. Fleuret and P. Fua. Robust People Tracking with Global Trajectory Optimization, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 744–750, 2006.
- [19] J. Bernardo and A. Smith. Bayesian Theory, *John Wiley*, New York, 2000.
- [20] S. Bhatia, L. Sigal, M. Isard and M. J. Black. 3D Human Limb Detection using Space Carving and Multi-view Eigen Models, *IEEE Workshop on Articulated and Nonrigid Motion*, 2004.
- [21] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, *ICSI-TR-97-021*, 1997.
- [22] C. M. Bishop. Pattern Recognition and Machine Learning, *Springer*, 2006.
- [23] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. *International Conference on Computer Vision (ICCV)*, pp. 231–236, 1993.
- [24] A. Blake and M. Isard. Active Contours. *Springer-Verlag*, 1998.
- [25] A. Blake and M. Isard. The Condensation Algorithm – conditional density propagation and applications to visual tracking *Advances in Neural Information Processing Systems*, pp. 361–368, 1997.
- [26] X. Boyen and D. Koller. Tractable Inference for Complex Stochastic Processes, *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 33–42, 1998.
- [27] E. Boyer. On Using Silhouettes for Camera Calibration, *Proceedings of the seventh Asian Conference on Computer Vision (ACCV)*, pp. 1–10, 2006.
- [28] E. Boyer and J.-S. Franco. A Hybrid Approach for Computing Visual Hulls of Complex Objects, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 695–701, 2003.
- [29] M. Brand. Shadow Puppetry, *International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1237–1244, 1999.
- [30] C. Bregler and J. Malik. Tracking people with twists and exponential maps, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8–15, 1998.
- [31] T. Brox, B. Rosenhahn, U. Kersting and D. Cremers. Nonparametric Density Estimation for Human Pose Tracking, *Symposium of the German Association for Pattern Recognition (DAGM)*, Springer-Verlag, LNCS 4174, pp. 546–555, 2006.

- [32] W. L. Buntine. Operations for learning with graphical models, *Journal of Artificial Intelligence Research*, Vol. 2, pp. 159–225, 1994.
- [33] M. Burl, M. Weber and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry, *European Conference on Computer Vision (ECCV)*, Vol. 2, pp. 628–641, 1998.
- [34] T.-J. Cham and J. Rehg. A multiple hypothesis approach to figure tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 239–245, 1999.
- [35] E. Charniak. Statistical Language Learning, *MIT Press*, Cambridge, Massachusetts, 1993.
- [36] G. K. M. Cheung, S. Baker and T. Kanade. Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 77–84, 2003.
- [37] G. K. M. Cheung, T. Kanade, J. Bouguet and M. Holler. A real time system for robust 3d voxel reconstruction of human motions, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 714–720, 2000.
- [38] K. Choo and D. Fleet. People tracking using hybrid Monte Carlo filtering, *International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 321–328, 2001.
- [39] C.-W. Chu, O. C. Jenkins and M. J. Mataric. Markerless Kinematic Model and Motion Capture from Volume Sequences, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 475–482, 2003.
- [40] P. Clifford. Markov random fields in statistics, *Disorder in Physical Systems*, pp. 19–32, Oxford University Press, 1990.
- [41] D. Comaniciu. An Algorithm for Data-Driven Bandwidth Selection, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 25, No. 2, 281–288, February 2003.
- [42] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach toward Feature Space Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 24, No. 5, pp. 603–619, May 2002.
- [43] G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks, *Artificial Intelligence*, Vol. 42, pp. 393–405, 1990.
- [44] J. Coughlan and S. Ferreira. Finding deformable shapes using loopy belief propagation, *European Conference on Computer Vision (ECCV)*, Vol. 3, pp. 453–468, 2002.
- [45] G. Csurka, C. Dance, C. Bray, L. Fan and J. Willamowski. Visual categorization with bags of keypoints, *In ECCV workshop on statistical learning in computer vision*, 2004.
- [46] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893, 2005.
- [47] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference, *Conference on Uncertainty in Artificial Intelligence*, pp. 211–219, 1996.
- [48] A. Dempster, N. Laird and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

- [49] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision (IJCV)*, Vol. 61, No. 2, pp. 185–205, 2004.
- [50] J. Deutscher, M. Isard and J. MacCormick. Automatic camera calibration from a single manhattan image, *European Conference on Computer Vision (ECCV)*, Vol. 4, pp. 175–188, 2002.
- [51] J. Deutscher, A. Davison and I. Reid. Automatic partitioning of high dimensional search spaces associated with articulated body motion capture, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 669–676, 2001.
- [52] J. Deutscher, A. Blake and I. Reid. Articulated body motion capture by annealed particle filtering, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 126–133, 2000.
- [53] J. Deutscher, B. North, B. Bascle and A. Blake. Tracking through singularities and discontinuities by random sampling, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1144–1149, 1999.
- [54] A. Doucet, N. de Freitas and N. Gordon. Sequential Monte Carlo methods in practice, *Statistics for Engineering and Information Sciences*, Springer Verlag, 2001.
- [55] A. Elgammal and C.-S. Lee. Inferring 3D Body Pose from Silhouettes Using Activity Manifold Learning, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 681–688, 2004.
- [56] G. Elidan, I. McGraw and D. Koller. Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing, *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, July 2006.
- [57] L. Fei-Fei, R. Fergus and P. Perona. One-Shot learning of object categories, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 28, No. 4, pp. 594–611, April 2006.
- [58] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 524–531, 2005.
- [59] P. Felzenszwalb and D. Huttenlocher. Pictorial Structures for Object Recognition, *International Journal of Computer Vision (IJCV)*, Vol. 61, No. 1, pp. 55–79, January 2005.
- [60] R. Fergus, P. Perona and A. Zisserman. A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 380–387, June 2005.
- [61] R. Fergus, P. Perona and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 264–271, 2003.
- [62] M. Fischler and R. Elschlager. The representation and matching of pictorial structures, *IEEE Transactions on Computer*, 22(1):67–92, January 1973.
- [63] J. Foley, A. van Dam, S. Feiner and J. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, ISBN: 0-201-12110-7, 1990.
- [64] D. A. Forsyth, O. Arikan, L. Ikemoto, J. O’Brien and D. Ramanan. Computational Studies of Human Motion: Part 1, Tracking and Motion Synthesis, *ISBN: 1-933019-30-1*, pp. 1–178, July 2006.
- [65] J.-S. Franco and E. Boyer. Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1747–1753, 2005.

- [66] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, Vol. 55(1), pp. 119–139, 1997.
- [67] Y. Freund and R. E. Schapire. A decision-theoretic generalization of online learning and an application to boosting, *Unpublished manuscript*. An extended abstract appeared in *Computational Learning Theory: Second European Conference (EuroCOLT)*, Springer-Verlag, pp. 23–37, 1995.
- [68] B. J. Frey and N. Jojic. A comparison of algorithms for inference and learning in probabilistic graphical models, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 27, No. 9, pp. 1392–1416, September 2005.
- [69] J. Gall, B. Rosenhahn, T. Brox, U. Kersting and H.-P. Seidel. Learning for multi-view 3D tracking in the context of particle filters, *International Symposium on Visual Computing (ISVC)*, Springer-Verlag, LNCS 4292, pp. 59–69, 2006.
- [70] J. Gall, J. Potthoff, C. Schnoerr, B. Rosenhahn and H.-P. Seidel. Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications, *Technical Report MPI-I-2006-4-009*, Saarbruecken, Germany, September 2006.
- [71] F. Grassia. Practical parameterization of rotations using the exponential map, *Journal of Graphics Tools*, 3(3):29–48, March 1998.
- [72] D. Gavrila and V. Philomin. Real-time Object Detection for Smart Vehicles, *IEEE International Conference on Computer Vision (ICCV)*, pp. 87–93, 1999.
- [73] D. Gavrila. The visual analysis of human movement: A survey, *Computer Vision and Image Understanding (CVIU)*, 73(1):82–98, 1999.
- [74] D. Gavrila and L. Davis. 3-D model-based tracking of humans in action: A multi-view approach, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 73–80, 1996.
- [75] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 6, No. 6, pp. 721–741, 1984.
- [76] N. Gordon, D. Salmond and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proceedings F: Radar and Signal Processing*, Vol. 140, Issue 2, pp. 107–113, April, 1993.
- [77] K. Grauman, G. Shakhnarovich, T. Darrell. Inferring 3D structure with a statistical image-based shape model, *IEEE International Conference on Computer Vision (ICCV)*, pp. 641–648, 2003.
- [78] T. Han, H. Ning and T. Huang. Efficient Nonparametric Belief Propagation with Application to Articulated Body Tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 214–221, 2006.
- [79] C. Harris and M. Stephens. A Combined Corner and Edge Detector, *Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.
- [80] W. K. Hastings. Monte Carlo sampling methods using Markov Chains and their applications, *Biometrika*, Vol. 57, pp. 97–109, 1970.
- [81] D. Haussler, A. Krogh, I. S. Mian and K. Sjolander. Protein modeling using hidden Markov models: Analysis of globins. *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, Vol. 1, pp. 792–802, 1993.

- [82] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks, *Science*, Vol. 313, No. 5786, pp. 504–507, July 2006.
- [83] G. Hinton, S. Osindero and Y. Teh. A fast learning algorithm for deep belief nets, *Neural Computation*, Vol. 18, pp. 1527–1554, 2006.
- [84] T. Hofmann. Probabilistic Latent Semantic Analysis, *UAI*, 1999.
- [85] D. Hoiem, A. Efros and M. Hebert. Putting Objects in Perspective, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 2137–2144, 2006.
- [86] T. Horprasert, D. Harwood and L. Davis. A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection, *ICCV Frame-Rate Workshop*, pp. 1–19, September 1999.
- [87] N. Howe. Silhouette Lookup for Monocular 3D Pose Tracking, *Image and Vision Computing*, Vol. 25, Issue 3, pp. 331–341, March 2007.
- [88] N. Howe. Boundary Fragment Matching and Articulated Pose Under Occlusion, *IV Conference on Articulated Motion and Deformable Objects*, 2006.
- [89] N. Howe and A. Deschamps. Better Foreground Segmentation Through Graph Cuts, *Technical Report*, Smith College, 2004.
- [90] N. Howe, M. Leventon, W. Freeman. Bayesian Reconstruction of 3D Human Motion from Single-Camera Video, *Neural Information Processing Systems (NIPS)*, 2000.
- [91] C. Hu, Q. Yu, Y. Li and S. Ma. Extraction of parametric human model for posture recognition using genetic algorithm, *IEEE International Conference on Automatic Face and Gesture Recognition (FG 2000)*, pp. 518–523, 2000.
- [92] G. Hua and Y. Wu. Measurement Integration Under Inconsistency for Robust Tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 650–657, 2006.
- [93] G. Hua, M.-H. Yang and Y. Wu. Learning to Estimate Human Pose with Data Driven Belief Propagation, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 747–754, 2005.
- [94] J. Hwang, S. Lay and A. Lippman. Nonparametric multivariate density estimation: a comparative study, *IEEE Transactions on Signal Processing*, Vol. 42(10), pp. 2795–2810, 1994.
- [95] A. T. Ihler, E. B. Sudderth, W. T. Freeman and A. S. Willsky. Efficient Multiscale Sampling from Products of Gaussian Mixtures, *Neural Information Processing Systems (NIPS)*, 2003.
- [96] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 690–695, 2001.
- [97] S. Ioffe and D. Forsyth. Probabilistic methods for finding people, *International Journal of Computer Vision (IJCV)*, Vol. 43, No. 1, pp. 45–68, 2001.
- [98] S. Ioffe and D. Forsyth. Finding people by sampling, *IEEE International Conference on Computer Vision (ICCV)*, pp. 1092–1097, 1999.
- [99] M. Isard. PAMPAS: Real-valued graphical models for computer vision, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 613–620, 2003.

- [100] M. Isard and J. MacCormick. BraMBLE: a Bayesian multiple-blob tracker, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 34–41, 2001.
- [101] T. Izo and E. Grimson. Simultaneous pose recovery and camera registration from multiple views of a walking person, *Image and Vision Computing*, 25(3): pp. 342–351, 2007.
- [102] T. Izo and E. Grimson. Simultaneous Pose Estimation and Camera Calibration from Multiple Views, *Proceedings of IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, Vol. 1, pp. 14–21, June 2004.
- [103] R. A. Jacobs, M. I. Jordan, S. Nowlan and G. Hinton. Adaptive mixtures of local experts, *Neural Computation*, Vol. 3, pp. 79–87, 1991.
- [104] T. Jaeggli, E. Koller-Meier and L. Van Gool. Monocular tracking with a mixture of view-dependent learned models, *IV Conference on Articulated Motion and Deformable Objects (AMDO)*, 2006.
- [105] F. V. Jensen. Bayesian Networks and Decision Graphs, *Springer-Verlag*, New York, 2001.
- [106] M. Jones and J. Rehg. Statistical Color Models with Application to Skin Detection, *International Journal of Computer Vision (IJCV)*, Vol. 46, No. 1, pp. 81–96, Jan 2002.
- [107] M. I. Jordan. Graphical models, *Statistical Science (Special Issue on Bayesian Statistics)*, Vol. 19, pp. 140–155, 2004.
- [108] M. I. Jordan, T. J. Sejnowski and T. Poggio. Graphical models: Foundations of neural computation, *MIT Press*, 2001.
- [109] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola and L. K. Saul. An introduction to variational methods for graphical models, *Machine Learning*, Vol. 37, Issue 2, pp. 183–233, 1999.
- [110] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, Vol. 6, pp. 181–214, 1994.
- [111] S. Ju, M. J. Black and Y. Yacoob. Cardboard people: A parameterized model of articulated motion, *IEEE International Conference on Automatic Face and Gesture Recognition (FG 1996)*, pp. 38–44, 1996.
- [112] I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 81–87, 1996.
- [113] R. Kehl, M. Bray, L. V. Gool. Full Body Tracking from Multiple Views Using Stochastic Sampling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 129–136, 2005.
- [114] K. Kim and L. Davis. Multi-Camera Tracking and Segmentation of Occluded People on Ground Plane using Search-Guided Particle Filtering, *European Conference on Computer Vision (ECCV)*, Vol. 3, pp. 98–109, 2006.
- [115] K. Kinoshita, Y. Ma, S. Lao and M. Kawade. A fast and robust 3D head pose and gaze estimation system, *International Conference on Multimodal Interfaces (ICMI)*, pp. 137–138, 2006.
- [116] S. Kirkpatrick, C. Gellatt and M. Vecchi. Optimisation by simulated annealing. *Technical report*, IBM Thomas J. Watson Research Centre, Yorktown Heights, NY, USA, 1982.
- [117] D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid bayes nets, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 324–333, 1999.

- [118] A. Kong, J. S. Liu and W. H. Wong. Sequential imputations and Bayesian missing data problems, *Journal of the American Statistical Association*, Vol. 89, pp. 278–288, 1984.
- [119] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm, *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 498–519, Feb 2001.
- [120] M. P. Kumar, P. H. S. Torr and A. Zisserman. Learning Layered Motion Segmentation of Video, *IEEE International Conference on Computer Vision (ICCV)*, pp. 33–40, 2005.
- [121] M. P. Kumar, P. H. S. Torr and A. Zisserman. Learning Layered Pictorial Structures from Video, *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, pp. 148–153, 2004.
- [122] X. Lan and D. Huttenlocher. Beyond trees: Common factor models for 2D human pose recovery, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 470–477, 2005.
- [123] X. Lan and D. Huttenlocher. A unified spatio-temporal articulated model for tracking. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 722–729, 2004.
- [124] Y. LeCun, F. Huang and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 97–104, 2004.
- [125] C.-S. Lee and A. Elgammal. Simultaneous Inferring View and Body Pose Using Torus Manifolds, *International Conference on Pattern Recognition (ICPR)*, Vol. 3, pp. 489–494, 2006.
- [126] M. Lee and R. Nevatia. Human pose Tracking using multi-level structured models, *European Conference on Computer Vision (ECCV)*, Vol. 3, pp. 368–381, 2006.
- [127] M. Lee and I. Cohen. Proposal Maps driven MCMC for Estimating Human Body Pose in Static Images, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 334–341, 2004.
- [128] M. Lee and I. Cohen. Human Upper Body Pose Estimation in Static Images, *European Conference on Computer Vision (ECCV)*, pp. 126–138, 2004.
- [129] S.-I. Lee, V. Ganapathi and D. Koller. Efficient Structure Learning of Markov Networks using L1-Regularization, *Advances in Neural Information Processing Systems (NIPS)*, 2006.
- [130] F. Lerasle, G. Rives and M. Dhome. Tracking of Human Limbs by Multiocular Vision, *Computer Vision and Image Understanding*, Vol. 75, Issue 3, pp. 229–246, 1999.
- [131] R. Li, M.-H. Yang, S. Sclaroff and T.-P. Tian. Monocular Tracking of 3D Human Motion with a Coordinated Mixture of Factor Analyzers, *European Conference on Computer Vision (ECCV)*, Vol. 2, pp. 137–150, 2006.
- [132] Y. Li, S. Ma and H. Lu. Human Posture Recognition Using Multi-Scale Morphological Method and Kalman Motion Estimation, *International Conference on Pattern Recognition (ICPR)*, Vol. 1, pp. 175–177, 1998.
- [133] Y. Li, A. Hilton and J. Illingworth. A relaxation algorithm for real-time multiview 3d-tracking. *Image and Vision Computing*, 20(12):841–59, 2002.
- [134] T. Lindeberg. Feature detection with automatic scale selection, *International Journal of Computer Vision (IJCV)*, Vol. 30, No. 2, pp. 77–116, 1998.
- [135] D. G. Lowe. Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision (IJCV)*, Vol. 60, No. 2, pp. 91–110, 2004.

- [136] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking, *European Conference on Computer Vision (ECCV)* Vol. 2, pp. 3–19, 2000.
- [137] D. J. C. MacKay. Information Theory, Inference, and Learning Algorithms, *Cambridge University Press*, Cambridge, U.K., 2003.
- [138] D. J. C. MacKay. Introduction to Monte Carlo methods, *Learning in Graphical Models*, pp. 175–204, 1999.
- [139] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three dimensional structure, *Proceedings of the Royal Society of London B*, Vol. 200, pp. 269–294, 1978.
- [140] S. McKenna, S. Jabri, Z. Duric and H. Wechsler. Tracking interacting people, *IEEE International Conference on Automatic Face and Gesture Recognition (FG 2000)*, pp. 348–358, 2000.
- [141] N. Metropolis and S. Ulam. The Monte Carlo method, *Journal of the American Statistical Association*, Vol. 44(247), pp. 335–341, September 1949.
- [142] I. Miki, M. Trivedi, E. Hunter and P. Cosman. Human Body Model Acquisition and Motion Capture Using Voxel Data, *Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects (AMDO)*, pp. 104–118, 2002.
- [143] B. Moghaddam and A. Pentland. Probabilistic Visual Learning for Object Representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 19, No. 7, pp. 696–710, July 1997.
- [144] A. Mohan, C. Papageorgiou and T. Poggio. Example-based object detection in images by components, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 23, No. 4, pp. 349–361, April 2001.
- [145] P. Moreels, M. Maire and P. Perona. Recognition by probabilistic hypothesis construction, *European Conference on Computer Vision (ECCV)*, Vol. 1, pp. 55–68, 2004.
- [146] G. Mori. Guiding Model Search Using Segmentation, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1417–1423, 2005.
- [147] G. Mori, X. Ren, A. Efros and J. Malik. Recovering Human Body Configurations: Combining Segmentation and Recognition, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 326–333, 2004.
- [148] G. Mori and J. Malik. Estimating Human Body Configurations using Shape Context Matching, *European Conference on Computer Vision (ECCV)*, Vol. 3, pp. 666–680, 2002.
- [149] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 289–296, 1998.
- [150] L. Muendermann, S. Corazza and T. Andriacchi. Accurately measuring human movement using articulated ICP with soft-joint constraints and a repository of articulated models, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [151] L. Mundermann, S. Corazza, A. M. Chaudhari, T. P. Andriacchi, A. Sundaresan and R. Chellappa. Measuring human movement for biomechanical applications using markerless motion capture, *IS&T/SPIE 18th Annual Symposium: Electronic Imaging*, San Jose, California, 2006.
- [152] K. Murphy, A. Torralba and W. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes, *Advances in Neural Information Processing Systems 16*, 2003.

- [153] K. Murphy and S. Russel. Rao-blackwellized particle filtering for dynamic bayesian networks, *Sequential Monte Carlo Methods in Practice*, pp. 499–515, Springer, 2001.
- [154] C. Musso, N. Oudjane and F. LeGland. Improving regularized particle filters, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.
- [155] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, pp. 355–368. MIT Press, 1999.
- [156] M. Niskanen, E. Boyer and R. Horaud. Articulated Motion Capture from 3-D Points and Normals, *British Machine Vision Conference (BMVC)*, Vol. 1, pp. 439–448, 2005.
- [157] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little and D. Lowe. A Boosted Particle Filter: Multitarget Detection and Tracking, *European Conference on Computer Vision (ECCV)*, Vol. 1, 28–39, 2004.
- [158] A. Opelt, A. Pinz and A. Zisserman. A Boundary-Fragment-Model for Object Detection, *European Conference on Computer Vision (ECCV)*, Vol. 2, pp. 575–588, 2006.
- [159] OpenCV Reference Manual, Intel Open Source Computer Vision Library, available at <http://www.intel.com/research/mrl/research/opencv/>.
- [160] D. Ormoneit, M. J. Black, T. Hastie, and H. Kjellström. Representing cyclic human motion using functional analysis. *Image and Vision Computing*, 23(14):1264–1276, Dec. 2005.
- [161] C. Papageorgiou and T. Poggio. A Trainable System for Object Detection, *International Journal of Computer Vision (IJCV)*, Vol. 38, No. 1, pp. 15–33, June 2000.
- [162] S. Park and J. K. Aggarwal. A Hierarchical Bayesian Network for Event Recognition of Human Actions and Interactions, *ACM Journal of Multimedia Systems*, 10(2), pp. 164–179, 2004.
- [163] S. Park and J. K. Aggarwal. Simultaneous Tracking of Multiple Body Parts of Interacting Persons, *Computer Vision and Image Understanding*, Vol. 102, Issue 1, pp. 1–21, April 2006.
- [164] V. Pavolvić, J. Rehg, T-J. Cham and K. Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models, *IEEE International Conference on Computer Vision (ICCV)*, pp. 94–101, 1999.
- [165] R. Plankers and P. Fua. Articulated Soft Objects for Video-based Body Modeling, *IEEE International Conference on Computer Vision (ICCV)*, pp. 394–401, 2001.
- [166] R. W. Poppe and M. Poel. Comparison of Silhouette Shape Descriptors for Example-based Human Pose Recovery, *IEEE International Conference on Automatic Face and Gesture Recognition (FG 2006)*, pp. 541–546, 2006.
- [167] R. Quandt and J. Ramsey. A new approach to estimating switching regressions, *Journal of the American Statistical Society*, Vol. 67, pp. 306–310, 1972.
- [168] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models, *IEEE ASSP Magazine*, pp. 4–15, January 1986.
- [169] D. Ramanan, D. Forsyth and A. Zisserman. Strike a Pose: Tracking People by Finding Stylized Poses, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 271–278, 2005.
- [170] D. Ramanan, D. Forsyth and K. Barnard. Detecting, Localizing, and Recovering Kinematics of Textured Animals, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 635–642, 2005.

- [171] D. Ramanan and D. Forsyth. Automatic Annotation of Everyday Movements, *Neural Information Processing Systems (NIPS)*, 2003.
- [172] D. Ramanan and D. Forsyth. Using temporal coherence to build models of animals, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 338–346, 2003.
- [173] D. Ramanan and D. Forsyth. Finding and tracking people from the bottom up, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 467–474, 2003.
- [174] X. Ren, A. Berg and J. Malik. Recovering Human Body Configurations using Pairwise Constraints between Parts, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 824–831, 2005.
- [175] C. P. Robert and G. Casella. Monte Carlo Statistical Methods (2nd edition), *Springer-Verlag*, 2004.
- [176] T. Roberts, S. McKenna and I. Ricketts. Human pose estimation using learnt probabilistic region similarities and partial configurations, *European Conference on Computer Vision (ECCV)*, Vol. 4, pp. 291–303, 2004.
- [177] J. Rodgers, D. Anguelov, H.-C. Pang and D. Koller. Object Pose Detection in Range Scan Data, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 2445–2452, 2006.
- [178] R. Ronfard, C. Schmid and B. Triggs. Learning to parse pictures of people, *European Conference on Computer Vision (ECCV)*, Vol. 4, pp. 700–714, 2002.
- [179] R. Rosales and S. Sclaroff. Learning Body Pose Via Specialized Maps, *Neural Information Processing Systems (NIPS)*, 2002.
- [180] R. Rosales and S. Sclaroff. Learning and Synthesizing Human Body Pose and Motion, *IEEE International Conference on Automatic Face and Gesture Recognition (FG 2000)*, 2000.
- [181] R. Rosales and S. Sclaroff. Inferring Body Pose without Tracking Body Parts, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 721–727, 2000.
- [182] B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette and H.-P. Seidel. A system for articulated tracking incorporating a clothing model, *Machine Vision and Applications (MVA)*, Vol. 18, No. 1, pp. 25–40, 2007.
- [183] B. Rosenhahn, T. Brox, U. Kersting, D. Smith, J. Gurney and R. Klette. A system for marker-less human motion estimation, *Kuenstliche Intelligenz (KI)*, No. 1, pp. 45–51, 2006.
- [184] B. Rosenhahn, U. Kersting, D. Smith, J. Gurney, T. Brox and R. Klette. A system for marker-less human motion estimation, *Symposium of the German Association for Pattern Recognition (DAGM)*, Springer-Verlag, LNCS 3663, pp. 230–237, 2005.
- [185] S. Roth, L. Sigal and M. J. Black. Gibbs likelihoods for Bayesian tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 886–893, 2004.
- [186] S. R. Sain. Multivariate locally adaptive density estimation, *Source Computational Statistics & Data Analysis archive*, Vol. 39(2), pp. 165–186, April, 2002.
- [187] M. Siddiqui and G. Medioni. Robust Real-Time Upper Body Limb Detection and Tracking, *ACM International Workshop on Video Surveillance & Sensor Networks (VSSN)*, 2006.
- [188] R. E. Schapire. Theoretical Views of Boosting and Applications, *Algorithmic Learning Theory*, 1999.

- [189] G. Shakhnarovich, P. Viola and T. Darrell. Fast Pose Estimation with Parameter Sensitive Hashing, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 750–757, 2003.
- [190] J. Shi and C. Tomasi. Good features to track, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, 1994.
- [191] K. Shoemake. Animating Rotations with Quaternion Curves, *Computer Graphics (SIGGRAPH)*, Vol. 19, pp. 245–254, 1985.
- [192] H. Sidenbladh and M. J. Black. Learning the statistics of people in images and video, *International Journal of Computer Vision (IJCV)*, Vol. 54, No. 1–3, pp. 183–209, 2003.
- [193] H. Sidenbladh, M. J. Black and D. Fleet. Stochastic tracking of 3D human figures using 2D image motion, *European Conference on Computer Vision (ECCV)*, Vol. 2, pp. 702–718, 2000.
- [194] L. Sigal and M. J. Black. HumanEva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion, *Technical Report: CS-06-08*, Brown University, 2006.
- [195] L. Sigal and M. J. Black. Predicting 3D People from 2D Pictures, *IV Conference on Articulated Motion and Deformable Objects (AMDO)*, 2006.
- [196] L. Sigal and M. J. Black. Measure Locally, Reason Globally: Occlusion-sensitive Articulated Pose Estimation, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 2041–2048, 2006.
- [197] L. Sigal, S. Bhatia, S. Roth, M. J. Black and M. Isard. Tracking Loose-limbed People, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 421–428, 2004.
- [198] L. Sigal, Y. Zhu, D. Comaniciu and M. J. Black. Tracking Complex Objects using Graphical Object Models, *1st International Workshop on Complex Motion*, 2004.
- [199] L. Sigal, M. Isard, B. H. Sigelman and M. J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation, *Neural Information Processing Systems (NIPS)*, Vol. 16, pp. 1539–1546, 2003.
- [200] R. Silva and R. Scheines. Bayesian learning of measurement and structural models, *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 825–832, 2006.
- [201] R. Silva, R. Scheines, C. Glymour and P. Spirtes. Learning the structure of linear latent variable models, *Journal of Machine Learning Research*, Vol. 7, pp. 191–246, February 2006.
- [202] B. W. Silverman. Density Estimation for Statistics and Data Analysis, *Chapman & Hall*, London, 1986.
- [203] S. Sinha, M. Pollefeys and L. McMillan. Camera Network Calibration from Dynamic Silhouettes, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 195–202, 2004.
- [204] J. Sivic, B. Russell, A. Efros, A. Zisserman and W. Freeman. Discovering object categories in image collections, *IEEE International Conference on Computer Vision (ICCV)*, 2005.
- [205] C. Sminchisescu, A. Kanaujia and D. Metaxas. Learning Joint Top-Down and Bottom-up Processes for 3D Visual Inference, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 1743–1752, 2006.

- [206] C. Sminchisescu, A. Kanaujia, Z. Li and D. Metaxas. Discriminative Density Propagation for 3D Human Motion Estimation, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 390–397, 2005.
- [207] C. Sminchisescu, A. Kanaujia, Z. Li and D. Metaxas. Learning to Reconstruct 3D Human Motion from Bayesian Mixtures of Experts: A Probabilistic Discriminative Approach, *Technical Report CSRG-502*, University of Toronto, 2004.
- [208] C. Sminchisescu and A. Jepson. Variational mixture smoothing for non-linear dynamical systems, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 608–615, 2004.
- [209] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 69–76, 2003.
- [210] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling, *International Journal of Robotics Research*, Vol. 22, No. 6, pp. 371–393, 2003.
- [211] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling, *European Conference on Computer Vision (ECCV)*, Vol. 1, pp. 769–783, 2002.
- [212] C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. *European Conference on Computer Vision (ECCV)*, Vol. 1, pp. 566–582, 2002.
- [213] P. Srinivasan and J. Shi. Bottom-up Recognition and Parsing of the Human Body, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [214] R. Srinivasan. Importance sampling - Applications in communications and detection, *Springer-Verlag*, Berlin, 2002.
- [215] J. Starck and A. Hilton. Model-based multiple view reconstruction of people, *IEEE International Conference on Computer Vision (ICCV)*, pp. 915–922, 2003.
- [216] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 246–252, 1999.
- [217] T. A. Stephenson. Conditional Gaussian Mixtures, *Technical Report IDIAP-RR 03-11*, 2003.
- [218] E. Sudderth. Graphical Models for Visual Object Recognition and Tracking, *PhD. Thesis*, Massachusetts Institute of Technology, May 2006.
- [219] E. Sudderth, M. Mandel, W. Freeman and A. Willsky. Distributed Occlusion Reasoning for Tracking with Non-parametric Belief Propagation, *Neural Information Processing Systems (NIPS)*, 2004.
- [220] E. Sudderth, A. Ihler, W. Freeman and A. Willsky. Nonparametric belief propagation, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 605–612, 2003.
- [221] J. Sun, H. Shum and N. Zheng. Stereo matching using belief propagation, *European Conference on Computer Vision (ECCV)*, pp. 510–524, 2002.
- [222] C. J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in a Single Image, *Computer Vision and Image Understanding*, Vol. 80, No. 3, pp. 349–363, 2000.

- [223] A. Torralba, K. P. Murphy and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 762–769, 2004.
- [224] A. Torralba, K. P. Murphy, W. T. Freeman and M. A. Rubin. Context-based vision system for place and object recognition, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 273–280, 2003.
- [225] L. Torresani, P. Hackney and C. Bregler. Learning Motion Style Synthesis from Perceptual Observations, *Neural Information Processing Systems (NIPS)*, 2006.
- [226] R. Urtasun, D. J. Fleet and P. Fua. 3D People Tracking with Gaussian Process Dynamical Models, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 238–245, 2006.
- [227] R. Urtasun, D. J. Fleet, A. Hertzmann and P. Fua. Priors for People Tracking from Small Training Sets, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1, pp. 403–410, 2005.
- [228] R. Urtasun and P. Fua. 3D Human Body Tracking Using Deterministic Temporal Motion Models, *European Conference on Computer Vision (ECCV)*, Vol. 3, pp. 92–106, 2004.
- [229] V. Vapnik. Statistical Learning Theory, *John Wiley and Sons, Inc.*, New York, 1998.
- [230] V. Vapnik. The Nature of Statistical Learning Theory, *Springer-Verlag*, New York, 1995.
- [231] S. Vedula, S. Baker and T. Kanade. Image-Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events, *ACM Transactions on Graphics*, Vol. 24, No. 2, pp. 240–261, April 2005.
- [232] S. Vedula, S. Baker, P. Rander, R. Collins and T. Kanade. Three-Dimensional Scene Flow, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 27, No. 3, pp. 475–480, March 2005.
- [233] J. Vermaak, A. Doucet and P. Perez. Maintaining Multi-Modality through Mixture Tracking, *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1110–1116, 2003.
- [234] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.
- [235] P. Viola, M. Jones and D. Snow. Detecting pedestrians using patterns of motion and appearance, *IEEE International Conference on Computer Vision (ICCV)*, pp. 734–741, 2003.
- [236] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 511–518, 2001.
- [237] S. Wachter and H. H. Nagel. Tracking Persons in Monocular Image Sequences, *Computer Vision and Image Understanding*, 74(3):174–192, 1999.
- [238] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Technical Report 649*, Department of Statistics, UC Berkeley, September 2003.
- [239] M. Wainwright, T. Jaakkola and A. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs, *Neural Information Processing Systems (NIPS)*, 2002.
- [240] J. Wang, D. Fleet and A. Hertzmann. Multifactor Gaussian process models for style-content separation, *International Conference on Machine Learning (ICML)*, Vol. 227, pp. 975–982, 2007.

- [241] P. Wang and J. M. Rehg. A Modular Approach to the Analysis and Evaluation of Particle Filters for Figure Tracking, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 790–797, 2006.
- [242] S. Waterhouse. Classification and Regression using Mixtures of Experts, *Ph.D. thesis*, Department of Engineering, Cambridge University, U.K, 1997.
- [243] M. Weber, M. Welling and P. Perona. Unsupervised Learning of Models for Recognition, *European Conference on Computer Vision (ECCV)*, pp. 18–32, 2000.
- [244] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology, *Neural Computation*, Vol. 13, pp. 2173–2200, 2001.
- [245] G. Wywill and T. L. Kunii. A functional model for Constructive Solid Geometry, *The Visual Computer*, Vol. 1, Pt. 1, pp. 3–14, July 1985.
- [246] W. Wiegnerinck. Variational approximations between mean field theory and the junction tree algorithm, *Conference on Uncertainty in Artificial Intelligence*, Vol. 16, pp. 626–633, 2000.
- [247] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland. Pfunder: Real-Time Tracking of the Human Body, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 19, No. 7, pp. 780–785, July 1997.
- [248] Y. Wu, G. Hua and T. Yu. Tracking Articulated Body by Dynamic Markov Network, *IEEE International Conference on Computer Vision (ICCV)*, pp. 1094–1101, 2003.
- [249] B. Xie, D. Comaniciu, V. Ramesh, M. Simon and T. Boult. Component fusion for face detection in the presence of heteroscedastic noise, *Annual Conf. of the German Society for Pattern Recognition (DAGM'03)*, pp. 434–441, 2003.
- [250] Y. Yacoob and L. S. Davis. Learned Models for Estimation of Rigid and Articulated Human Motion from Stationary or Moving Camera, *International Journal of Computer Vision (IJCV)*, Vol. 36, No. 1, pp. 5–30, 2000.
- [251] H. Yalcin, M. J. Black and R. Fablet. The dense estimation of motion and appearance in layers, *IEEE Workshop on Image and Video Registration*, 2004.
- [252] J. S. Yedidia, W. T. Freeman and Y. Weiss. Generalized belief propagation, *Neural Information Processing Systems (NIPS)*, pp. 689–695, 2000.
- [253] J. S. Yedidia, W. T. Freeman and Y. Weiss. Understanding belief propagation and its generalizations, *Technical Report*, TR2001-22, MERL, 2001.
- [254] J. S. Yedidia. An idiosyncratic journey beyond mean field theory, *Advanced Mean Field Methods*. MIT Press, 2001.
- [255] J. S. Yedidia, W. T. Freeman and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms, *IEEE Transactions on Information Theory*, Vol. 51(7), pp. 2282–2312, July 2005.
- [256] S. Yonemoto, D. Arita and R. Taniguchi. Real-time human motion analysis and IK-based human figure control, *Proceedings of the Workshop on Human Motion (HUMO)*, 2000.
- [257] J. Zhang, J. Luo, R. Collins and Y. Liu. Body Localization in Still Images Using Hierarchical Models and Hybrid Search, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 1536–1543, 2006.

- [258] N. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference, *Journal of Artificial Intelligence Research*, Vol. 5, pp. 301–328, 1996.
- [259] T. Zhao and R. Nevatia. Tracking Multiple Humans in Complex Situations, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 26, No. 9, pp. 1208–1221, September 2004.
- [260] T. Zhao and R. Nevatia. Bayesian Human Segmentation in Crowded Situations. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 459–466, 2003.
- [261] T. Zhao, T. Wang and H. Shum. Learning A Highly Structured Motion Model for 3D Human Tracking, *Asian Conference on Computer Vision (ACCV)*, 2002.
- [262] L. Zhu, Y. Chen and A. Yuille. Unsupervised Learning of a Probabilistic Grammar for Object Detection and Parsing, *Neural Information Processing Systems (NIPS)*, 2006.
- [263] L. Zhu and A. Yuille. A Hierarchical Compositional System for Rapid Object Detection, *Neural Information Processing Systems (NIPS)*, 2005.